KOMPENDIUM

CSS-Praxisbuch

Das Kompendium

Die Reihe für umfassendes Computerwissen

Seit mehr als 20 Jahren begleiten die KOMPENDIEN aus dem Markt+Technik Verlag die Entwicklung des PCs. Mit ihren bis heute über 500 erschienenen Titeln deckt die Reihe jeden Aspekt der täglichen Arbeit am Computer ab. Die Kompetenz der Autoren sowie die Praxisnähe und die Qualität der Fachinformationen machen die Reihe zu einem verlässlichen Partner für alle, ob Einsteiger, Fortgeschrittene oder erfahrene Anwender.

Das KOMPENDIUM ist praktisches Nachschlagewerk, Lehr- und Handbuch zugleich. Auf bis zu 1.000 Seiten wird jedes Thema erschöpfend behandelt. Ein detailliertes Inhaltsverzeichnis und ein umfangreicher Index erschließen das Material. Durch den gezielten Zugriff auf die gesuchte Information hilft das KOMPENDIUM auch in scheinbar aussichtslosen Fällen unkompliziert und schnell weiter.

Praxisnahe Beispiele und eine klare Sprache sorgen dafür, dass bei allem technischen Anspruch und aller Präzision die Verständlichkeit nicht auf der Strecke bleibt.

Mehr als 5 Millionen Leser profitierten bisher von der Kompetenz der KOMPENDIEN.

PEARSON Education

Markt+Technik

CSS-Praxisbuch

Der Weg zum pflegeleichten Web-Design

HELMA SPONA

KOMPENDIUM

Einführung | Arbeitsbuch | Nachschlagewerk

Bibliografische Information Der Deutschen Bibliothek

Die Deutsche Bibliothek verzeichnet diese Publikation in der Deutschen Nationalbibliografie; detaillierte bibliografische Daten sind im Internet über <http://dnb.ddb.de> abrufbar.

10 9 8 7 6 5 4 3 2 1

08 07 06

ISBN 3-8272-6892-3

Coverkonzept: independent Medien-Design,
Widenmayerstraße 16, 80538 München
Coverlayout: Thomas Arlt, tarlt@adesso21.net
Titelfoto: IFA-Bilderteam
Lektorat: Brigitte Alexandra Bauer-Schiewek, bbauer@pearson.de
Korrektorat: Petra Alm
Herstellung: Elisabeth Prümm, epruemm@pearson.de
Satz: Michael und Silke Maier, Ingolstadt (www.magus-publishing.de)
Druck und Verarbeitung: Bercker, Kevelaer
Printed in Germany

Im Überblick

Inhaltsverzeichnis

In diesem Buch enthalten ist eine Browser-Referenzkarte, die eine komplette Übersicht der Browser-Unterstützung der jeweiligen CSS-Eigenschaften und -Elemente bietet. Sollte sie fehlen, wenden Sie sich bitte an unseren Verlag unter `info@pearson.de` – wir sorgen umgehend für Ersatz.

Einleitung

Herzlich Willkommen beim Kompendium Cascading Style Sheets (CSS). Dieses Buch soll Ihnen dabei helfen, nicht nur in CSS einzusteigen, sondern vor allem den Einsatz von CSS hinsichtlich der Browserkompatibilität zu optimieren.

Konkretes Ziel ist neben der Vorstellung der einzelnen CSS-Eigenschaften vor allem die Ermittlung der Browser, die die Eigenschaft unterstützen bzw. fehlerhaft oder gar nicht darstellen. Nur so können Sie die Eigenschaften auswählen, die bereits jetzt praxistauglich sind. Welche das sind, hängt natürlich vom Zweck Ihres CSS-Einsatzes ab. Für eine firmeninterne Intranet-Anwendung, bei der nur ein Browser unterstützt werden muss, kann das ganz anders aussehen, als wenn Sie eine Webseite für das Internet erstellen, die von vielen verschiedenen Besuchern genutzt wird.

Buchaufbau

Um diese Ziele zu verwirklichen, besteht dieses Buch aus zwei großen Teilen und einem Anhang. Im ersten Teil, der die Kapitel 1 bis 3 umfasst, werden die wesentlichen Konzepte von CSS vorgestellt, die Vor- und Nachteile von CSS erläutert und relevante Begriffe aus CSS und HTML erläutert. Darüber hinaus erfahren Sie hier Details zu den verschiedenen Selektoren von CSS und der allgemeinen Syntax.

Falls Sie noch nicht viel über CSS und/oder HTML wissen, sollten Sie diese drei Kapitel auf jeden Fall gründlich lesen. Nur so können Sie auch den Referenzteil verstehen und die Beispiele nachvollziehen.

Der zweite Teil ist der Referenzteil. Er widmet sich den einzelnen Eigenschaften von CSS. Diese werden dazu nach Formatierungsbereichen geordnet, wie Textformatierungen, Größe und Positionierung oder Tabellen. Das führt zwar dazu, dass nicht alle Kapitel gleich lang sind, dafür haben Sie es aber erheblich einfacher, was die Suche nach bestimmen Eigenschaften betrifft.

Kapitelaufbau

Damit Sie schnell die praxistauglichen Befehle von den noch wenig bis gar nicht unterstützten CSS-Eigenschaften unterscheiden können, wurden die Kapitel in zwei bis drei Bereiche aufgeteilt.

Im ersten Bereich finden Sie die praxistauglichen Eigenschaften. Darunter habe ich alle CSS-Eigenschaften zusammengefasst, die von mehr als 80% der verwendeten Browsern unterstützt werden. Dem liegen die aktuellen Browser-Anteile gemäß WebHits[1] zum Zeitpunkt der Drucklegung zu Grunde.

Informationen zu den Marktanteilen der einzelnen Browser finden Sie in Kapitel 1, »CSS-Grundlagen«.

In Einzelfällen werden aber auch Eigenschaften als praxistauglich eingestuft, bei denen es nicht so tragisch ist, wenn der Browser die Eigenschaft nicht unterstützt. Das ist immer dann der Fall, wenn die Eigenschaft einen Effekt erzeugt, der nicht für die Bedienung der Webseite erforderlich ist.

Der zweite Teil des Kapitels enthält dann die Eigenschaften, die noch nicht praxistauglich sind.

In einigen Kapiteln finden Sie darüber hinaus noch einen dritten Teil mit einem kleinen und manchmal auch größeren Praxisteil, in dem die Eigenschaften und Selektoren des Kapitels noch einmal praxisnah erläutert werden. Das ist nicht in allen Kapitel möglich beziehungsweise notwendig, weil zum Teil auch bei den einzelnen Eigenschaften schon komplexere Beispiele gezeigt werden.

Suchen und Finden

Wichtig in einem solchen Referenzwerk ist natürlich auch, dass Sie schnell mal etwas nachschlagen können. Damit das möglichst einfach geht, wurden viele nützliche Indizes und Übersichten integriert.

Grundsätzlich gilt dabei, dass die CSS-Befehle im Referenzteil nach ihren deutschen Bezeichnungen sortiert sind. Wenn Sie also beispielsweise die CSS-Eigenschaft für den Innenabstand eines Elements ermitteln wollen, würden Sie in Kapitel »Größen, Abstände und Positionierung« suchen und dort unter »Innenabstand« nachschlagen.

Wenn Sie wissen, dass es die Eigenschaft `padding` gibt, aber nicht die deutsche Übersetzung kennen und vielleicht auch nicht genau wissen, in welchem Zusammenhang die Eigenschaft behandelt wird, können Sie im Index nachschlagen. Das Buch verfügt über zwei verschiedene Indizes. Einen Index mit den CSS-Eigenschaften und Selektoren und einen zweiten Gesamtindex. In beiden würden Sie die Eigenschaft finden.

1 http://www.webhits.de

Auch der Anhang eignet sich zum Nachschlagen. Neben Übersichten zur Browserkompatibilität, die in den gleichen Rubriken vorliegen wie die Referenzkapitel, also Textformatierungen, Größen, Abstände und Positionierung etc., finden Sie dort auch eine Gesamtübersicht aller CSS-Eigenschaften und -Selektoren, aufgeschlüsselt nach CSS-Version. Möchten Sie beispielsweise wissen, ob eine bestimmte Eigenschaft auch in der CSS-Mobile-Spezifikation enthalten ist, schlagen Sie entweder im Index nach, um im Referenzteil die Seite zu finden, auf der die Eigenschaft behandelt wird. Dort wird dann auch angegeben, zu welchen Spezifikationen die Eigenschaft gehört. Alternativ können Sie im Anhang in der Übersicht CSS-Mobile-Spezifikation nachsehen, ob dort die Eigenschaft aufgeführt wird.

Kapitelinhalte und Symbole

Die ersten drei Kapitel erläutern die Grundlagen von CSS. Sie sind daher besonders für CSS-Einsteiger sehr wichtig.

Behandelte CSS-Standards

In diesem Buch werden die CSS-Standards 1 bis 2.1, sowie TV und Mobile beschrieben. Von CSS 3.0 werden nur Teile behandelt, die bereits jetzt von einigen Browsern unterstützt werden, oder bei denen eine Unterstützung in naher Zukunft zu erwarten ist. Der CSS 3.0-Standard wird daher nicht vollständig beschrieben.

In den Tabellen zur Browserkompatibilität werden zu den einzelnen Eigenschaften auch deren mögliche Werte aufgelistet. Kursive Formatierungen kennzeichnen dabei den Standardwert der Eigenschaft, der verwendet wird, wenn Sie die Eigenschaft nicht explizit angeben. Finden Sie dort beispielsweise folgende Angabe, dann ist `none` der Standardwert der Eigenschaft:

Mögliche Werte: `capitalize`, `uppercase`, `lowercase`, *`none`*, `inherit`

Kapitel 1: CSS-Grundlagen

Die Basis von CSS ist immer eine Auszeichnungssprache, mit der ein zu formatierendes Dokument erstellt wird. Auf dieses Dokument wenden Sie mit CSS die Formatierungen an. Hier wird beispielhaft auf die am häufigsten verwendete Auszeichnungssprache im Internet, HTML 4, eingegangen.

Neben den Grundzügen von CSS erfahren Sie auch, welche Vor- und Nachteile physische und logische Auszeichnungen sowie Dokumentstrukturen haben und welche Möglichkeiten CSS bietet.

Darüber hinaus werden auch die Browserversionen erläutert. Sie erfahren hier, welche Browser eine gleiche Darstellung liefern und welche auf unterschiedlichem Code basieren. Nur mit diesem Hintergrundwissen haben Sie

die Möglichkeit, Ihre CSS-Stile mit möglichst wenigen Browsern so zu testen, dass Sie die Anzeige in anderen Browsern und auf anderen Betriebssystemen voraussagen können.

Kapitel 2: Stile definieren

In dem zweiten Kapitel geht es um die Syntax von CSS, also darum, wie Sie Stile benennen, was Stile überhaupt sind und welche Typen von Selektoren es gibt.

Auch wichtige Einheiten, wie z.B. Größeneinheiten, der Standardwert `inherit` sowie wichtige Konzepte von CSS, wie z.B. Vererbung, Überschreibung und die Grundlagen des Boxmodells, werden erläutert.

Kapitel 3: Browseroptimierung

Da leider immer noch nicht alle Browser alle Features, Selektoren und Eigenschaften von CSS unterstützen, ist es natürlich besonders wichtig, dass Sie Ihre Stylesheets so optimieren, dass jeder Browser die Seite akzeptabel und brauchbar anzeigt. Das notwendige Rüstzeug dazu liefert dieses Kapitel.

Sie erfahren, wie Sie Stile so definieren bzw. einbinden, dass ganz gezielt nur bestimmte Browser sie ausführen oder dass sie vor bestimmten Browsern verborgen werden.

Referenzkapitel

Die Kapitel 4 bis 11 stellen den Referenzteil dar. Sie erläutern, sortiert nach Rubriken, die einzelnen CSS-Eigenschaften. Zu Anfang eines jeden Kapitels werden die Konzepte und Grundlagen von CSS vorgestellt, die Sie für das Verständnis des Kapitels benötigen. Die können abhängig vom Inhalt der Kapitel mal länger oder auch ganz kurz ausfallen.

Kapitel 4: Textformatierungen

In diesem Kapitel geht es um die Formatierung von Texten. Dazu gehört neben der Textausrichtung (sowohl vertikal wie horizontal) auch die Farbgebung, die Schriftgröße und die besonderen Textauszeichnungen, wie Unterstreichungen, fette und kursive Formatierungen.

Kapitel 5: Größen, Abstände und Positionierung

Dies ist sicherlich das umfangreichste und wichtigste Kapitel des Referenzteils, in dem es ausschließlich um das Boxmodell von CSS geht, das die Basis aller Formatierungen bezüglich der Größe und Positionierung von Elementen darstellt. Neben dem Boxmodell, das ausführlich erläutert wird, lernen Sie hier die verschiedenen Positionierungs- und Anzeigearten kennen.

Kapitel 6: Listen und Aufzählungen

Dieses Kapitel befasst sich mit Listen und Aufzählungen. Dazu werden Eigenschaften zum Definieren und Ausrichten von Aufzählungszeichen genauso erläutert wie die Anweisungen aus CSS 2 zum Erzeugen von eigenen Zählern.

Kapitel 7: Bilder und Hintergründe

Hintergrundformatierungen sind neben den Schrifteigenschaften die häufigsten Formatierungen, die am meisten auffallen. Daher ist es natürlich vor allem hier wichtig, auf die Browserkompatibilität zu achten. In diesem Kapitel werden Ihnen daher die für die Hintergrundformatierungen erforderlichen CSS-Kenntnisse vermittelt.

Kapitel 8: Linien und Rahmen

Dieses Kapitel beschäftigt sich mit den Rahmeneigenschaften und Stilen sowie mit den Möglichkeiten, Elemente zu umranden, ohne damit die Boxgröße des Elements zu beeinflussen.

Kapitel 9: Tabellen und Spalten

Das Kapitel 9 »Tabellen und Spalten« beschäftigt sich mit der Tabellenformatierung. Dazu gehört die Darstellung der Zellumrandungen genauso wie die Ausrichtung der Tabellenbeschriftung und die Handhabung leerer Zellen.

Kapitel 10: Spezielle Medien: Druck- und Sprachausgabe

Hier geht es um die Druck- und Sprachausgabe. Beide Medien sind noch starken Einschränkungen hinsichtlich der Browserkompatibilität unterworfen. Vor allem die Sprachausgabe wird noch von keinem aktuellen Browser direkt unterstützt. Dennoch sollten Sie Informationen zur Sprachausgabe in Ihre Stylesheets integrieren, da Webseiten durchaus mit Hilfe von so genannten Screenreadern vorgelesen werden können, die natürlich auch Teile der CSS-Befehle zur Sprachausgabe unterstützen.

Kapitel 11: Sonstige Formatierungen, Pseudo-Elemente und -Klassen

Im letzten Kapitel geht es um einige wenige Eigenschaften, die sich in die vorherigen Kapitel nicht einordnen ließen. Der wesentliche Teil beschäftigt sich jedoch mit Pseudo-Elementen und Pseudo-Klassen. Sie ermöglichen es Ihnen, auch solche Elemente der Webseite zu formatieren, die nicht direkt aus dem (X)HTML-Code abgeleitet werden können.

Verwendete Symbole

Zur Erläuterung der Browser-Kompatibilität werden fünf verschiedene Symbole genutzt.

Tabelle E.1: Verwendete Symbole

○	Dieses Symbol bedeutet, dass der CSS-Standard bzw. die CSS-Eigenschaft zwar vollständig implementiert ist, aber dennoch so viele Fehler enthält, dass die Darstellung im Prinzip unbrauchbar ist.
⊙	Dieses Symbol zeigt an, dass der Browser den Standard/die Eigenschaft nur teilweise unterstützt und dass die unterstützten Teile zudem zahlreiche Fehler enthalten.
◉	Mit diesem Symbol werden Browser gekennzeichnet, die zwar nur Teile eines Standards oder der Eigenschaft beherrschen, diese Teile aber im Wesentlichen fehlerfrei.
●	Dieses Symbol zeigt an, dass der Browser den Standard bzw. die Eigenschaft nahezu vollständig beherrscht. Wird es verwendet, um nicht die Unterstützung einer einzelnen Eigenschaft anzuzeigen, sondern geht es um größere Teilkonzepte von CSS oder eine bestimmte CSS-Version, schließt das nicht aus, dass einzelne Teile/Eigenschaften davon fehlerhaft implementiert sind.
	Wird kein Symbol verwendet, bedeutet dies, dass der CSS-Standard bzw. die Eigenschaft oder der Selektor nicht bzw. so fehlerhaft unterstützt wird, dass eine Unterstützung nicht erkennbar ist.

Bei den Übersichten im Anhang werden teilweise Tabellen aus den Referenzkapiteln zusammengefasst. So gibt es zum Beispiel im Referenzteil zu jedem Wert der `display`*-Eigenschaft eine Kompatibilitätstabelle, im Anhang jedoch nur eine Zusammenfassung. Das kann zu scheinbaren Widersprüchen führen. In solchen Fällen wurden Eigenschaften als teilweise fehlerfrei unterstützt gekennzeichnet, wenn der Browser zumindest die wichtigsten Werte korrekt ausführt.*

Symbole in der Marginalspalte

Neben diesen kleinen Icons, die nur in tabellarischen Übersichten Anwendung finden, gibt es auch in der Marginalspalte Symbole, die in der Regel den Browser-Logos entsprechend. Sie kennzeichnen Erläuterungen zu einzelnen Browsern. Normalerweise handelt es sich dabei um nähere Beschreibungen zu Fehlern und wie sie sich vermeiden lassen oder zu abweichendem (nicht unbedingt fehlerhaftem) Verhalten der Browser, das Sie beim Erstellen Ihrer Stylesheets berücksichtigen sollten.

Tabelle E.2: Verwendete Browser-Icons

	Amaya
	Firefox für Windows und Mac
	iCab für Max OS X
	Internet Explorer für Windows und Mac
	Mozilla
	Netscape 6+, 7+
	Netscape Navigator 4.x und niedriger
	Der Opera-Browser
	Palm-Browser
	Safari für Mac

So wurde getestet

Für die Angaben der Browserkompatibilität wurde für jede behandelte CSS-Eigenschaft und jeden Selektor eine Testseite erstellt. Auf dieser Seite wird dann erläutert, wie sie bei korrekter Darstellung aussehen sollte. Anhand der Abweichungen können Sie dann sehen, ob ein Browser die Eigenschaft unterstützt.

Abbildung E.1: Die Beispielseiten erläutern Ihnen, wie die Formatierungen korrekt aussehen sollten.

Die Ergebnisse dieser Beispielseiten wurden dann in die Tabellen zur Browserkompatibilität übernommen und zum Teil noch durch Informationen ergänzt, die Erfahrungswerte darstellen und bestimmte Probleme bei der Kombination von Eigenschaften und Werten betreffen.

Für den Test wurden die folgenden Browser eingesetzt.

- der Palm-Browser auf einem Palm® Tungsten™ C
- Netscape Navigator 4.73 unter Windows XP
- Netscape 6 und Netscape 7.1 unter Windows XP
- Mozilla 1.8 Beta unter Windows XP
- Internet Explorer 4.1 unter Windows 98
- Internet Explorer 5.01 unter Windows Me

- Internet Explorer 5.5 unter Windows Me
- Internet Explorer 6 unter Windows XP Pro
- Internet Explorer 7 Beta unter Windows XP Pro
- Internet Explorer 5.0 unter Mac OS 8
- Internet Explorer 5.2 unter Mac OS X 10.3
- der Opera-Browser 6-8 unter Windows XP Pro
- FireFox unter Windows XP und Mac OS X 10.3
- Safari 2.0 unter Mac OS X 10.3
- iCab 2.9.x unter Mac OS X 10.3
- iCab 3.0 Beta unter Mac OS X 10.3

Bei den Angaben zu Konqueror wurde auf die Kompatibilität mit der Rendering-Engine KHTML gesetzt, die auch in Safari genutzt wird; außerdem wurden zusätzlich die Informationen auf der Herstellerseite mit einbezogen.

Aufgrund der vielen verschiedenen Betriebssysteme und Kombinationsmöglichkeiten mit den einzelnen Browsern konnte natürlich nicht jede mögliche Kombination getestet werden. Daher sind kleinere Abweichungen denkbar, wenn Sie eine andere Betriebssystem-Version verwenden oder auch eine nur geringfügig andere Versionen einsetzen.

Bitte bedenken Sie, dass es sich bei den Angaben zum Internet Explorer 7, iCab 3.0 und zu Mozilla 1.8 noch um BETA-Versionen handelt. Nicht unterstützte Features können daher in späteren Versionen noch hinzugefügt und auch Fehler noch beseitigt werden. Selbstverständlich ist es auch denkbar, dass sich dabei andere Fehler einschleichen.

Bei einer derart großen Anzahl von Browsern ist es leider nicht möglich, in jedem einzelnen Fall auf die neue Browserversion zu warten, um auf BETA-Versionen verzichten zu können. Würde man das Erscheinen der Release-Version von Mozilla 1.8 oder Internet Explorer 7 abwarten, wäre dann sicher schon eine Beta 9 des Opera-Browsers verfügbar und mit welchem Recht würde man dann nicht auf die fertige Opera-Version warten. Fängt man einmal damit an, würde ein solches Buch nie erscheinen. Aus diesem Grund wurden alle Browser in den Versionen verwendet, die zum Zeitpunkt der Drucklegung verfügbar waren.

Die Autorin

Ich, Helma Spona, bin seit über 10 Jahren als Autorin im Computerbereich tätig und befasse mich neben der Programmierung mit VBA, Visual Basic, PHP und anderen Programmiersprachen auch seit den Anfängen des Internets mit der Website-Gestaltung.

Nach einem Studium der Wirtschaftswissenschaften mit den Schwerpunkten Betriebsinformatik und Energiewirtschaft arbeite ich als freiberufliche Autorin und Programmiererin und betreibe eine EDV-Beratung am Niederrhein, die sich schwerpunktmäßig mit Webanwendungen und Webdesign beschäftigt. Ich habe bisher über 40 Bücher für verschiedene Verlage und zahlreiche Fachartikel veröffentlicht und bin zudem als Herausgeberin des Loseblattwerks »Web-Design Aktuell« des MEV-Verlags tätig. Nicht zuletzt deshalb bin ich bestens mit den Sorgen und Nöten von Webdesignern beim Einsatz von CSS vertraut.

Auch meine eigenen Webseiten

- `www.helma-spona.de`
- `www.s-v-g.net`
- `www.natur-fotos.biz`

nutzen aktuelle Webtechniken von SVG über PHP und MySQL bis zu CSS.

Fragen und Anmerkungen

Falls Sie Fragen und Anmerkungen zum Buch haben oder über den einen oder anderen Fehler zu einem Browser berichten möchten, können Sie sich gerne an mich wenden. Ich versuche jede Frage zum Buch innerhalb von zwei Werktagen zu beantworten.

Dazu steht Ihnen auf meiner Webseite `www.helma-spona.de` ein Kontaktformular zur Verfügung. Ich nehme darüber gerne Kritik und Anregungen oder auch Hinweise entgegen. Haben Sie eine konkrete Frage zum Buch, können Sie gerne auch die FAQ zum Buch nutzen, die Sie unter dem URL `www.helma-spona.de/buchdb` finden. Hier stehen gegebenenfalls auch Fehlerberichtigungen oder ergänzende Informationen und Links zum Buch.

Ich bitte allerdings um Verständnis dafür, dass ich keine Hausarbeiten, Diplom-, Seminararbeiten etc. bearbeite. Solche Anfragen erhalte ich immer wieder und muss das sowohl aus rechtlichen, zeitlichen wie moralischen Gründen ablehnen.

Wenn Sie mir Mitteilungen über Fehlverhalten oder unterstützte Features einzelner Browser machen, werde ich diese gerne sammeln, auf meiner Webseite veröffentlichen und bei einem Nachdruck oder einer Neuauflage berücksichtigen. Insbesondere Informationen über Browser, die mir zum Testen nicht zur Verfügung stehen, sind natürlich immer willkommen. Das gilt vor allem für Linux-Browser oder exotischere Betriebssysteme wie Solaris und BeOS.

Marken und Warenzeichen

In der Regel sind die Namen der Browser, wie beispielsweise Opera™, Safari™ und Internet Explorer, oder Teile davon wie Palm® eingetragene Marken, Warenzeichen oder Handelsmarken. Die Nennung dieser Namen erfolgt unter ausdrücklicher Anerkennung dieser Marken.

Gleiches gilt auch für die Firmennamen der Hersteller Microsoft®, Opera™ Software ASA und Apple®.

Teil 1 CSS-Konzepte im Überblick

1 CSS-Grundlagen

CSS zu lernen ist im Prinzip nicht sonderlich schwer. Es geht dabei im Wesentlichen darum, dass Sie wissen, wie und zu welchen Zwecken Sie CSS einsetzen und wie CSS-Anweisungen syntaktisch aufgebaut werden. Was daneben noch fehlt, sind die einzelnen CSS-Eigenschaften und ihre möglichen Werte. Die brauchen Sie jedoch nicht zu lernen, dafür finden Sie in diesem Buch einen ausführlichen Referenzteil, die Referenzkarte und eine Übersicht im Anhang.

1.1 Was ist CSS?

CSS ist die Abkürzung für Cascading Style Sheets. Wörtlich übersetzt bedeutet dies »kaskadierende Stilblätter«, was natürlich etwas unpassend ist. Besser lässt es sich mit »vererbbare Formatvorlage« oder »vererbbare Stile« übersetzen. Aber was genau steckt dahinter?

Bei CSS handelt es sich um eine Sammlung von Eigenschaften, Werten und Syntaxregeln, die dazu dienen, Webseiten zu formatieren. Die Webseite kann als XHTML-Seite oder HTML-Seite vorliegen. Für CSS spielt das eine untergeordnete Rolle. CSS ohne (X)HTML macht keinen Sinn, da CSS selbst nicht dargestellt werden kann. Sie müssen sich CSS mehr wie eine Farbpalette für Ihre Webseite vorstellen, die ohne die angemalten Flächen natürlich auch keinen Sinn macht.

Stylesheets

CSS-Code wird in der HTML-Seite direkt oder in einer externen Textdatei gespeichert und dann auf die Elemente der HTML-Seite angewendet. Die Gesamtheit der definierten Stile wird als Stylesheet bezeichnet.

Selektoren

Die definierten Stile in einem CSS-Stylesheet bestehen aus einem Namen und den Formatierungen. Die Stilnamen werden als Selektoren bezeichnet. Selektor deshalb, weil über den Namen des Stils auch das (X)HTML-Element (oder mehrere), dem die Formatierung zugewiesen wird, bestimmt, also ausgewählt, wird. CSS-Stile lassen sich mit den Absatz- und Zeichenformaten Ihrer Textverarbeitung vergleichen, denn sie legen fest, wie der damit formatierte Teil der Webseite aussieht. Sie können damit sowohl die Zeichenformatierungen wie auch Hintergrundbilder, Aufzählungen, Einrückungen, Abstände etc. definieren.

Wie funktioniert CSS?

Der CSS-Standard legt lediglich die CSS-Eigenschaften fest sowie die Syntax, mit der Selektoren definiert, kombiniert und zugewiesen werden. CSS-Code beschreibt also nur, wie eine Webseite aussehen soll. Der Rest ist Aufgabe der Browser oder des Programms, mit dem die Webseite dargestellt wird. Wenn ein Programm verwendet wird, das die Webseite vorliest, ein so genannter Screenreader, muss dieses die entsprechenden CSS-Befehle für die Sprachausgabe beherrschen, damit Sie die Reihenfolge steuern können, in der die Seite vorgelesen wird. Gleiches gilt für den Browser. Auch er muss in der Lage sein, die CSS-Stile zu lesen und die Webseite entsprechend anzuzeigen. CSS definiert also nur, wie eine Webseite aussehen *soll*. Ob die Ausgabe auch so erfolgt, hängt von dem Programm ab, das die Ausgabe erzeugt.

Der CSS-Standard im Überblick

CSS war einer der ersten W3C-Standards, die verabschiedet wurden. Mittlerweile liegt CSS in der Version 2.1 vor, die CSS-Version 3.0 ist in Arbeit. Erhebliche Unterschiede gibt es vor allem zwischen den CSS-Versionen 1 und 2 und wieder in der zukünftigen Version 3.0. Während in CSS 1.0 vor allem die reine Formatierung der Webseite im Vordergrund stand, wie die Schriftfarbe und -größe oder die Hintergrundformatierungen, geht es im CSS 2.0-Standard vornehmlich um die Positionierung der Elemente innerhalb der Seite sowie deren Überlappung und Größe. Der CSS-Befehlssatz, der diese Formatierungen ermöglicht, wird kurz mit dem Begriff Positionierung umschrieben, obwohl das nicht völlig zutreffend ist.

Mit CSS 3.0 werden vor allem neue Selektoren eingeführt, mit denen Sie auf bestimmte Tabellenzellen zugreifen oder nur Elemente mit bestimmten Eigenschaften formatieren können. Aber das ist noch Zukunftsmusik, weil nur wenige Browser den kommenden CSS 3.0-Standard bereits unterstützen – und das auch nur ansatzweise.

:-) TIPP

Neuerungen bezüglich des CSS-Standards und Hinweise auf neue Browser, Editoren etc., die im Zusammenhang mit CSS wichtig sind, finden Sie auf der W3C-Webseite: `http://www.w3c.org`. *Dort können Sie, wenn Sie möchten, auch die aktuellen CSS-Standards als PDF-Dateien oder in anderen Formaten herunterladen.*

CSS-Status

Aktuell ist CSS 2.0 der letzte offizielle W3C-Standard. Sowohl CSS 2.1 als auch CSS 3.0 befinden sich noch in der Entwicklung. Dabei gilt CSS 2.1 jedoch als nahezu abgeschlossen. Die endgültige Standardisierung von CSS 3.0 ist aber noch Zukunftsmusik.

Aus diesem Grund sollten Sie alle Erläuterungen zu CSS 3.0 mit äußerster Vorsicht genießen. Es kann sich noch eine Menge ändern.

Einsatzmöglichkeiten

Die Einsatzmöglichkeiten von CSS sind zwar relativ begrenzt, wenn Sie aber alle Möglichkeiten von CSS ausreizen, können Sie hervorragende Layouts für Ihre Webseiten realisieren, die einer mit Flash oder mit anderen proprietären Techniken erzeugten Webseite in nichts nachstehen muss.

Der große Vorteil von CSS ist, dass Sie gleiche Formatierungen für mehrere Webseiten nutzen können. Dennoch haben Sie die Möglichkeit, einzelne Formatierungen durch andere Einstellungen zu überschreiben. Genauso wurde das Farbleitsystem auf der Webseite in Abbildung 1.1 erzeugt. Die immer gleichen Formatierungen, wie Schriftarten und -größen, wurden in einer allgemeinen CSS-Datei definiert. Diese wurden dann auf den einzelnen Seiten um verschiedene Hintergrundfarben und Hyperlink-Farben ergänzt bzw. durch neue Werte ersetzt. Das lässt sich sogar mit den Grafiken in der linken oberen Ecke machen. Auch die sind ausschließlich per CSS definiert.

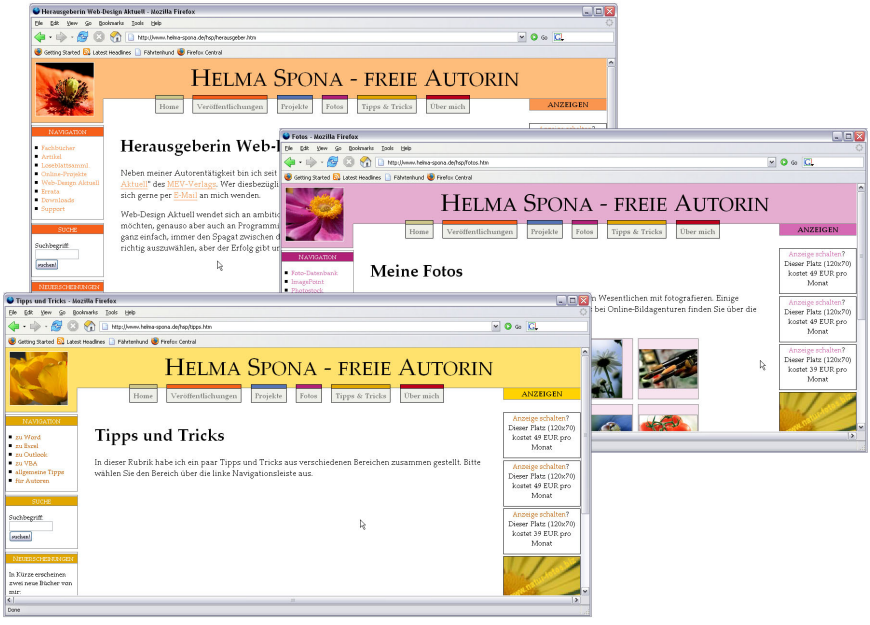


Abbildung 1.1: Das Farbleitsystem, das sich in unterschiedlichen Farben und Fotos äußert, wurde ausschließlich mittels CSS realisiert.

Aber nicht nur für diese Zwecke lässt sich CSS einsetzen. Genauso gut können Sie für eine Webseite unterschiedliche Formatierungen für die Anzeige am Bildschirm und die Ausgabe auf dem Drucker definieren. Schließlich sind nicht alle Elemente einer Webseite im Ausdruck notwendig. So können Sie beispielsweise nicht benötigte Elemente für den Druck ausblenden oder andere einblenden.

Ausgabemedien

Das beschränkt sich nicht auf die Druckausgabe. CSS 2.0 gestattet es sogar, Anweisungen zur Steuerung der Sprachausgabe oder Angaben für die Ausgabe auf einer Braillezeile für Blinde in das CSS-Stylesheet einzufügen. Für welches Ausgabemedium das Stylesheet bestimmt ist, legen Sie fest, wenn Sie es der Webseite zuweisen.

Details dazu finden Sie in Kapitel 3, »Browseroptimierung«.

CSS lässt sich aber nicht nur in Webseiten einsetzen. Sie können es auch nutzen, um beispielsweise SVG-Grafiken zu formatieren.

SVG ist eine auf XML basierende Markup-Sprache zur Erzeugung von Vektorgrafiken. Auch hier erfolgt die Formatierung der Grafikelemente mit Hilfe von CSS, wenngleich die CSS-Befehle vom CSS 2.0-Standard etwas abweichen. Das Prinzip bleibt jedoch das gleiche. Mehr zu SVG finden Sie bei Interesse auf der Webseite des W3C (`www.w3.org`*) oder auf der Seite* `www.s-v-g.net`.

Was CSS (zurzeit) noch nicht kann

CSS kann zwar eine ganze Menge, sogar weit mehr als die aktuellen Browser an CSS-Unterstützung bieten, dennoch gibt es einige Dinge, die wünschenswert wären, aber einfach nicht funktionieren. Beispielsweise ist es nicht möglich, in CSS-Stilen Ausdrücke zu verwenden, die bei Anzeige im Browser berechnet werden. Das würde es erheblich einfacher machen, Farben zu berechnen oder die Position von Elementen abhängig von bestimmten Parametern wie der Bildschirmauflösung zu machen.

Wünschenswert wäre es ebenfalls, wenn CSS den Zugriff auf ein bestimmtes Element, beispielsweise die vierte, siebte oder achte Tabellenzeile, ermöglichen würde. Das wird allerdings in CSS 3.0 möglich sein. Es bleibt in dieser Hinsicht abzuwarten, wie schnell die Browserhersteller die CSS 3.0-Befehle in ihre Browser implementieren. Und damit wären wir auch schon bei den Vor- und Nachteilen von CSS.

1.2 Vor- und Nachteile von CSS

Der größte Nachteil von CSS ist, dass ältere Browser CSS nur eingeschränkt oder gar nicht unterstützen. Das kann dazu führen, dass eine Webseite nicht mehr bedienbar ist und natürlich auch nicht mehr gut aussieht.

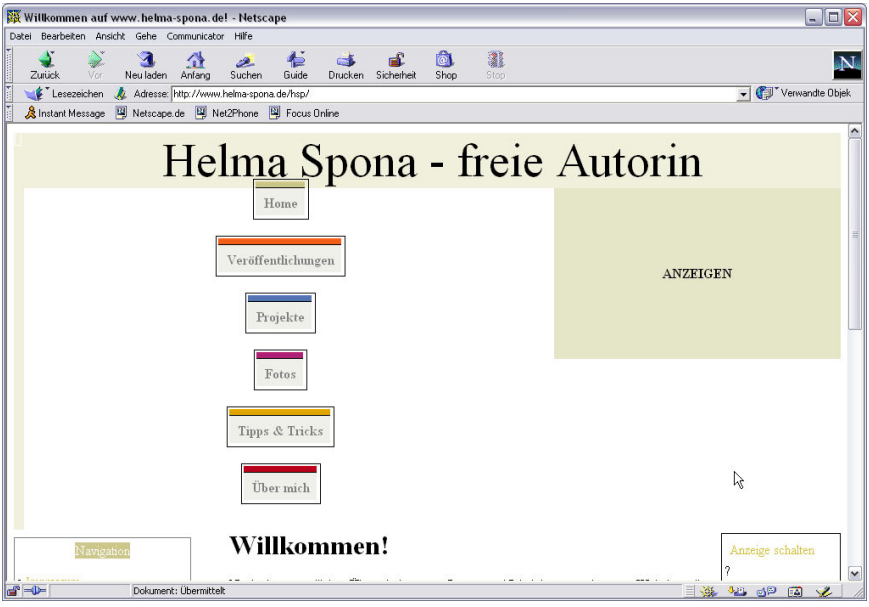


Abbildung 1.2: Die gleiche Webseite wie oben im Netscape Navigator 4.78, der die verwendeten CSS-Befehle nur mangelhaft umsetzt

Es gibt aber auch da eine gute Nachricht. Sie können den CSS-Code fast immer so gestalten, dass auch ältere Browser die Webseite passabel anzeigen. Dazu sorgen Sie einfach dafür, dass CSS-Befehle vor den Browsern verborgen werden, die diese nicht ausführen können. Wie das geht, wird ausführlich in Kapitel 3, »Browseroptimierung«, beschrieben. Für fast alle Browser gibt es nämlich so genannte Browserweichen.

Weitere Nachteile von CSS gibt es nicht. Die Vorteile überwiegen daher auf jeden Fall. Der größte von allen: Die Wartbarkeit einer mit CSS formatierten Webseite ist deutlich einfacher als bei herkömmlicher Formatierung mittels HTML-Tags. Das liegt daran, dass Sie mit einem einzigen CSS-Stil beispielsweise alle Absätze der Website formatieren können. Wenn Sie später mal die Farbe der Schrift oder die Schriftart ändern möchten, müssen Sie das auch nur an einer Stelle machen, nämlich im CSS-Stylesheet. Verwenden Sie hingegen zur Formatierung immer noch die HTML-Tags, wie beispielsweise `font`, müssten Sie die Änderung in jedem einzelnen Absatz vornehmen. Das kann dauern ...

Abbildung 1.3: Nicht perfekt, aber lesbar; die Seite nach Optimierung für den Netscape Navigator

Zudem wird durch den Einsatz von CSS die Struktur der (X)HTML-Seiten wesentlich einfacher. Das liegt daran, dass Sie in der (X)HTML-Datei lediglich die Struktur der Seite festlegen. Das Layout und die Formatierung bestimmt hingegen der CSS-Code. Dies ist eines der wichtigsten Prinzipien, das Sie berücksichtigen sollten, wenn Sie barrierefreie Seiten erstellen möchten, die einfach zu warten sein sollen und zudem gut aussehen müssen.

INFO

Als barrierefrei gilt eine Webseite dann, wenn sie so gestaltet ist, dass alle Ausgabeprogramme, die (X)HTML beherrschen, die Inhalte der Seite darstellen können. Barrierefrei bedeutet also keinesfalls, dass die Seite in allen Browsern gleich aussehen muss, sondern nur, dass sie auch behinderten Besuchern zugänglich ist. Diese Besucher nutzen oft besondere Programme oder Hardware, wie Screenreader oder Braillezeilen, die die Inhalte vorlesen (Screenreader) oder in Blindenschrift darstellen (Braillezeilen). Auf diesen Ausgabegeräten spielt die Formatierung dann keine Rolle. Für sie ist nur wichtig, dass der HTML-Code strukturiert und korrekt ist. Für die Formatierung können sie dann per CSS sorgen. Das ist auch der Grund, warum im Allgemeinen Webseiten, die ausschließlich per CSS formatiert wurden, als barrierefrei gelten. Richtig ist das allerdings nicht ganz. Denn auch eine HTML-Seite, die gar keine Angaben zur Formatierung dafür aber validen (=standardkonformen) HTML-Code enthält, ist barrierefrei.

XHTML

Für XHTML-Seiten ist CSS ohnehin die einzige Möglichkeit der Formatierung, da im XHTML-Standard keine Elemente für die Formatierung der Seite zur Verfügung stehen. XHTML stellt wie HTML 4.1 Strict nur logische Textauszeichnungen zur Verfügung.

Browserkompatibilität

Wie bereits erwähnt, zählt die teilweise noch schlechte Browserkompatibilität zu den Nachteilen des CSS-Einsatzes. Allerdings hat sich die Browserunterstützung in den letzten Jahren erheblich verbessert. Das liegt daran, dass viele neue Browser und neue Browserversionen erschienen sind, deren CSS-Unterstützung den CSS 2.0-Standard bereits weitgehend abdeckt. Problematisch ist allerdings immer noch, dass einige der Browser bestimmte CSS-Formatierungen fehlerhaft anwenden. Aber auch da hat sich in letzter Zeit einiges getan. Sogar im Internet Explorer 7 wurden einige Fehler des Internet Explorers 5 und 6 ausgemerzt. Ansonsten sind Fortschritte bezüglich der CSS-Unterstützung des Internet Explorers 7 allerdings ausgeblieben. Sicherlich können auf die diesen Tests zugrunde liegende BETA noch einige Änderungen folgen, aber große Sprünge wird man von der endgültigen Version des Internet Explorers dennoch nicht erwarten dürfen.

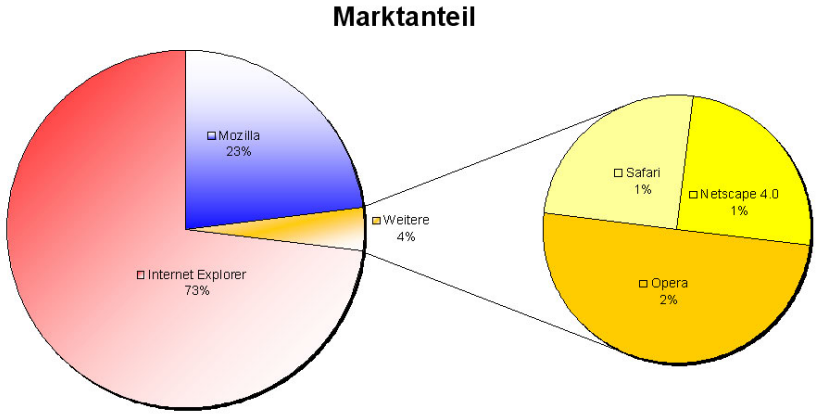
Vor allem die Markteinführung von Mozilla Firefox hat einiges zur besseren CSS-Unterstützung beigetragen, da sowohl Mozilla als auch Firefox große Teile des CSS-Standards unterstützen (sogar schon ein paar Elemente aus CSS 3.0).

Marktanteile

Problematische Browser sind hingegen immer noch der Netscape Navigator 4.x sowie ältere Internet Explorer-Versionen wie der Internet Explorer 3 und 4.x. Gerade beim Internet Explorer 4 ist zwar die Unterstützung von CSS 1.0 sehr gut, der CSS 2.0-Standard wird jedoch teilweise sehr fehlerhaft ausgeführt, was zu Abweichungen gegenüber der Darstellung in moderneren Browsern führt. Wenn Sie sich aber die Marktanteile der einzelnen Browser vor Augen halten, haben gerade diese Browser einen so verschwindend geringen Marktanteil, dass Sie sie durchaus vernachlässigen können.

Betrachten Sie die Marktanteile der Browser in Abbildung 1.4, erkennen Sie, dass den größten Marktanteil immer noch der Internet Explorer hat. Von diesen 73% fallen jedoch 71% auf den Internet Explorer 6, der eine sehr gute CSS-Unterstützung bietet, auch wenn sie nicht ganz so gut ist, wie die der Mozilla-Browser und der Opera-Browser. Nimmt man den Opera-Browser hinzu, machen die Browser Internet Explorer 6, Mozilla (incl. Firefox und Netscape 6.x/7.x) und der Opera-Browser zusammen 96% aus. Das bedeutet, wenn Sie sich beim Einsatz von CSS an diesen Browsern orientieren, sehen 96% der Benutzer eine perfekte Webseite und das mit minimalem Aufwand.

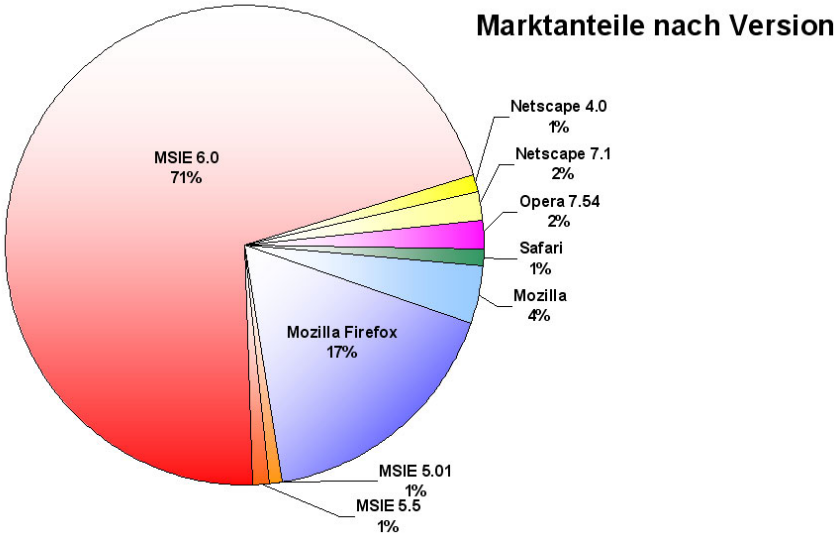
Abbildung 1.4: Aktuelle Marktanteile der Browser nach Hersteller (Stand 24.05.2005, Daten: WebHit[a]s)



a. http://www.webhits.de

Interessant ist aber auch die Aufteilung der einzelnen Browser nach Versionen sowie die Betriebssysteme, die verwendet werden. Sie können daran sehr schön erkennen, dass es gar nicht mehr viele Browser gibt, die CSS 2.0 nicht bzw. mangelhaft unterstützen. Auch bei den Betriebssystemen herrscht Windows eindeutig vor. Daran können Sie sehen, dass spezielle Browser für Linux und Unix, wie Konqueror, nur in verschwindend geringer Anzahl eingesetzt werden. Schließlich sind auch in dem Marktanteil von Mozilla Versionen für Linux/Unix erfasst.

Abbildung 1.5: Die verwendeten Browserversionen (Stand 24.05.2005, Daten: WebHits)



Die verwendeten Betriebssysteme sind auch deshalb interessant, weil Sie daran sehen können, ob Sie neben CSS auch andere Techniken auf Ihrer Webseite einsetzen können, die unter Umständen ein bestimmtes Betriebssystem voraussetzen. Das gilt vor allem für den Einsatz von Plug-Ins wie Flash-Player oder SVG-Viewer. Für CSS ist das Betriebssystem insofern interessant, als es Aufschluss darüber gibt, ob Sie bestimmte spezielle Browserversionen gesondert unterstützen müssen. Gemeint ist damit beispielsweise der Internet Explorer für Mac. Er unterstützt CSS in einigen Bereichen weitaus besser als die Windows-Version. Dennoch gibt es einige kleine Fehler in der CSS-Implementierung. Solange er einen gewissen Marktanteil hat und noch nicht ganz von Safari und Firefox verdrängt ist, sollten Sie darauf Rücksicht nehmen. Es stellt sich allerdings die Frage, welchen Marktanteil der Internet Explorer für Macintosh noch hat. Da Safari ausschließlich für Mac OS X zur Verfügung steht und bereits 1% Marktanteil hat, dies aber gleichzeitig auch der Marktanteil des Mac-Betriebssystems (einschließlich Mac OS 8.x) ist, könnte man daraus schließen, dass alle Macintosh-Benutzer Safari als Browser verwenden. Damit hätten Browser wie der Internet Explorer für Macintosh, Camino oder iCab sowie Mozilla einen Marktanteil von 0% auf Macintosh-Rechnern. Das kann also nicht sein, zumal das bedeuten würde, dass Benutzer von Mac OS 8.x nicht im Internet surfen.

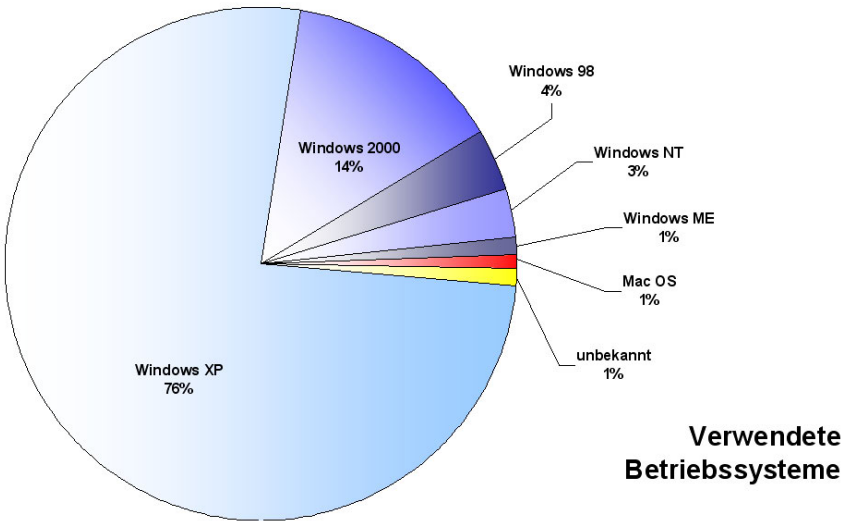


Abbildung 1.6: Die verwendeten Betriebssysteme (Stand 24.05.2005, Daten: WebHits)

!! STOP

Alle diese Statistiken müssen Sie daher relativieren. Es treten kleine Ungenauigkeiten durch Rundungen auf und jede Statistik ist nur so gut wie die zugrunde liegenden Daten. Die WebHits-Daten werden beispielsweise von den Besuchern des URLs `www.webhits.de` *erhoben. Die Verteilung von Betriebssystemen und Browsern auf Ihrer Webseite kann gänzlich anders aussehen. Wenn Sie beispielsweise auf Ihrer Webseite Tipps für Mac-Benutzer anbieten und diese Seite auch entsprechend in den Suchergebnissen der*

großen Suchmaschinen gelistet ist, werden sich nur sehr wenige Windows-Benutzer auf Ihre Seite verirren. Warum sollten die auch nach Macintosh-Tipps suchen. Da WebHits jedoch sehr viele Besucher hat und durch das Thema der Seite (Webstatistiken, Seitenzähler etc.) browser- und betriebssystemneutral ist, können Sie die hier gewonnenen Daten durchaus als realistisch (bezogen auf alle Internet-Nutzer) ansehen. Ob das dann aber auch auf Ihre Seite zutrifft, lässt sich damit noch nicht sagen.

Bedeutung der Marktanteile

Für die Webentwicklung bedeutet dies vor allem, dass Sie Ihren CSS-Code an den marktführenden Browsern ausrichten sollten. Das heißt keinesfalls, dass Sie nur die CSS-Befehle nutzen können, die alle Browser unterstützen. Sie sollten aber dafür sorgen, dass die Browser, die eine bestimmte Formatierung nicht unterstützen, dennoch eine akzeptable Ersatzformatierung anzeigen.

Damit Ihnen das gelingt, enthält der Referenzteil zu jeder CSS-Eigenschaft eine ausführliche Tabelle, in der Sie die Browserkompatibilität auf den ersten Blick sehen können.

Darüber hinaus finden Sie in Kapitel 3, »Browseroptimierung«, Möglichkeiten beschrieben, wie Sie bestimmte Stile und Formatierungen vor ungeeigneten Browsern verbergen können.

Browser und ihre Fähigkeiten im Überblick

Nach den bisher eher allgemeinen Ausführungen zur Browserkompatibilität geht es nun in die Details.

Überblick

Wichtig für das Verständnis der Browserkompatibilität ist zu wissen, dass es nicht so viele verschiedene Browser gibt, wie dies den Anschein hat. Neben dem Internet Explorer, der für Macintosh und Windows zur Verfügung steht, gilt es zwischen Netscape 4.x und früher sowie Mozilla 0.x+, dem Opera-Browser, Safari/Konqueror und iCab zu unterscheiden.

Betriebssystemunterschiede

Nur beim Internet Explorer spielt das Betriebssystem überhaupt eine Rolle, da die Versionen für Windows und Macintosh auf unterschiedlichem Code beruhen und daher auch die Seiten unterschiedlich darstellen. Für Mozilla und dem Opera-Browser gilt das nicht. Hier wird für jedes Betriebssystem die gleiche Rendering-Engine verwendet. Wenn Sie Ihre Seite mit dem Opera-Browser für Windows getestet haben, können Sie also ziemlich sicher sein, dass diese Seite im Opera-Browser für Mac oder im Opera-Browser für Linux genauso dargestellt wird, vorausgesetzt die Version ist die gleiche.

Als Rendering-Engine wird der Programmteil eines Browsers bezeichnet, der für die Auswertung des (X)HTML- und CSS-Codes sorgt und diesen entsprechend den gültigen Standards in eine grafische Ausgabe umsetzt. Allein die Rendering-Engine des Browsers bestimmt also die Qualität der Ausgabe.

Alle Browser, die die Gecko-Rendering-Engine von Mozilla verwenden, basieren somit auf Mozilla und stellen die Seiten auch gleich dar. Dazu gehören die Netscape-Browser ab der Version 6.x, Camino (nur für Mac OS X) und Firefox (für Windows, Mac OS X, Linux). Wichtig ist hier nur, auf welcher Version von Mozilla die Browser beruhen.

Safari/Konqueror und iCab sind hingegen von Mozilla und Microsoft unabhängige Entwicklungen, die jedoch alle nur für ein Betriebssystem zur Verfügung stehen. Bei Konqueror ist dies Linux, bei Safari und iCab ist das Mac OS X. Konqueror und Safari teilen sich jedoch die gleiche Rendering-Engine namens KHTML.

Zur Darstellung der CSS-Kompatibilität verwendet die Tabelle 1.1 einige Symbole, die einer Erklärung bedürfen. Sie werden so weitgehend auch für alle anderen Tabellen und späteren Erläuterungen verwendet.

Dieses Symbol bedeutet, dass der CSS-Standard zwar vollständig implementiert ist, aber dennoch so viele Fehler enthält, dass die Darstellung im Prinzip unbrauchbar ist. ○

Dieses Symbol zeigt an, dass der Browser den Standard nur teilweise unterstützt und dass die unterstützten Teile des Standards zudem zahlreiche Fehler enthalten. ⊙

Mit diesem Symbol werden Browser gekennzeichnet, die zwar nur Teile eines Standards beherrschen, diese Teile aber im Wesentlichen fehlerfrei. ◘

Dieses Symbol zeigt an, dass der Browser den Standard nahezu vollständig beherrscht. Das schließt selbstverständlich nicht aus, dass der Browser den einen oder anderen Fehler macht. ●

Leere Tabellenzellen geben an, dass der Standard gar nicht berücksichtigt wird, und ein »+« hinter der Version zeigt an, dass damit die genannte Version und höhere Versionen gemeint sind. Analog dazu gibt ein Browsername mit »-« hinter der Versionsnummer an, dass neben der angegebenen Version auch Vorversionen gemeint sind. Wenn nicht anders angegeben, beziehen sich die Angaben auf alle Betriebssystemversionen, für die der Browser verfügbar ist.

Tabelle 1.1: Unterstützte CSS-Standards nach Browserversionen

Browser	CSS 1	CSS 2	CSS 2.1	CSS 3
IE 3	⊙			
IE 4	○	⊙		
IE 5.0 (Mac)	●	●		
IE 5.1+ (Mac)	●	●	◎	
IE 5.0/5.01 (Windows)	●	⊙		
IE 5.5 (Windows)	●	●		
IE 6	●	●[1]	◎	
IE 7	●	●[1]	◎	
NN 4.x	○	⊙		
Mozilla 0.x/Netscape 6	●	○[1]		
Mozilla 1.x/Netscape 7	●	○[1]		
Mozilla 1.7/Camino 0.8	●	●[1]		
Mozilla 1.7.8/Netscape 7.1	●	●[1]	◎	◎
Mozilla 1.8	●	●[1]	◎	◎
Firefox 1.0	●	●[1]	◎	
Opera 4	●	●[1]		
Opera 5	●	●[1]		
Opera 6	●	●[1]		
Opera 7	●	●[1]	●	
Opera 8	●	●	●	
Safari 1.x	●	●[1]	◎	◎
Safari 2.0	●	●[1]	◎	◎
Konqueror 3.x	●	●[1]	◎	◎
iCab 2.x	●	◎		
iCab 3.0+	●	●[2]	●[2]	
Palm Web Browser 2.x	◎[2]	◎[2]		

[1] Sprachausgaben werden nicht unterstützt

[2] auch Teile von HTML werden nicht unterstützt, die CSS-Unterstützung selbst ist aber gar nicht so schlecht für einen PDA-Browser

Browser ohne CSS-Unterstützung

Noch heute werden Browser benutzt, die kein CSS unterstützen, auch wenn deren Marktanteil so gering ist, dass sie in den Browserstatistiken nicht mehr auftauchen. Zu diesen Browsern gehören NCSA Mosaik, der Netscape Navigator 3 und früher, sowie der Internet Explorer 2 und früher. Aber auch Textbrowser wie Lynx, die vorwiegend von körperbehinderten Internetnutzern verwendet werden, sind immer noch im Einsatz. Lynx zeigt generell nur die Texte einer Seite an, Bilder und auch Formatierungen jeglicher Art werden nicht interpretiert. Daher ist es für solche Browser ganz wichtig, dass Sie Ihre Webseite korrekt strukturieren.

Mehr dazu erfahren Sie weiter unten im Abschnitt »Trennung von Layout und Inhalt«.

Internet Explorer 3 und Netscape Navigator 4.x

Der Internet Explorer 3 unterstützt den CSS 1.0-Standard, das allerdings nur sehr rudimentär. Dargestellt werden vor allem Formatierungen der Schrift sowie Hintergrundfarben. Das ist aber auch im Wesentlichen schon alles, was dieser Browser korrekt beherrscht. Weite Teile des CSS 1.0-Standards werden nur fehlerhaft oder gar nicht interpretiert. Etwas, aber nicht viel besser sieht es beim Netscape Navigator 4.x aus. Er unterstützt zwar bereits weite Teile von CSS 1.0, diese aber teilweise fehlerhaft. Zudem werden Teile von CSS 2.0 unterstützt, wenn auch noch fehlerhafter und lückenhafter als bei CSS 1.0. Probleme hat der Netscape Navigator 4.x vor allem bei der Vererbung von CSS-Eigenschaften.

Mehr zur Vererbung finden Sie in Kapitel 2 »Stile definieren«. Dort wird im Detail auf die Probleme des Netscape Navigators 4.x eingegangen.

Zudem benötigt der Netscape Navigator 4.x zwingend aktiviertes JavaScript, damit CSS ausgeführt wird. Das liegt an der internen Implementierung von CSS über JSS. JSS ist eine Netscape-spezifische Vorform von CSS, nämlich JavaScript Style Sheets.

Internet Explorer 4

Der Internet Explorer 4 ist hinsichtlich CSS 1.0 bereits recht praxistauglich, stellt er doch den CSS 1.0-Standard weitgehend dar. Allerdings macht auch er hier noch einige Fehler. Beim CSS 2.0-Standard sieht es noch schlechter aus. Hier treten Fehler vor allem bei der Positionierung von Elementen auf und der CSS 2.0-Standard wird auch nur teilweise unterstützt.

Internet Explorer 5.x für Windows und Macintosh

Der Internet Explorer 5 und höher unterstützt den CSS 1.0-Standard sowohl in der Mac- wie der Windows-Version weitgehend fehlerfrei und vollständig. Unterschiede gibt es hier vor allem bei CSS 2.0 und CSS 2.1. Hier hat die Mac-Version die Nase vorn. Während CSS 2.0 schon von der Version 5.0 (Mac) weitgehend fehlerfrei unterstützt wird, ist die CSS 2.0-Unterstützung unter Windows erst in der Version 5.5 brauchbar. Auf dem Mac wird ab der Version 5.1 auch schon ein Teil der CSS 2.1-Spezifikation ausgeführt.

Für den Macintosh ist die Version 5.2 nach Aussage von Microsoft die letzte Version. Die Entwicklung der Macintosh-Version wird eingestellt. Daher steht auch der Internet Explorer 6 und höher nur für Windows zur Verfügung.

Der Opera 4.x-Browser

Der Opera-Browser in der Version 4.0 beherrscht den CSS 1.0-Standard vollständig und CSS 2.0 nahezu vollständig, allerdings ohne die Sprachausgabe.

Internet Explorer 6 und 7

Die Version 6 des Internet Explorers ist hinsichtlich der CSS-Unterstützung erheblich besser geworden als seine Vorgänger. Nun wird der CSS 2.0-Standard nahezu vollständig (mit Ausnahme der Sprachausgabe) unterstützt und auch Teile von CSS 2.1 sind bereits implementiert. In der Version 7 des Internet Explorers sind vor allem Fehler in der CSS-Implementierung bezüglich der relativen Positionierung von Elementen zu beheben. Große Neuerungen gibt es allerdings nicht und die erste BETA-Version des Internet Explorers weist auch noch viele Fehler der Vorversion auf.

Die Opera-Browser in den Versionen 5-8

Die Opera-Browser in den Versionen 5 und 6 führen CSS 1.0- und CSS 2.0-Code weitgehend vollständig und fehlerfrei aus, die Versionen 7.0 und 8.0 unterstützen zudem CSS 2.1 nahezu vollständig und fehlerfrei sowie CSS 3.0 zum Teil. Die Opera-Versionen unterstützen jedoch alle keine Sprachausgabe.

Netscape 6/7 und Mozilla 0.x und höher

Netscape 6 und Mozilla 0.x verwenden die gleiche Codebasis und führen daher beide CSS 1.0 vollständig und weitgehend fehlerfrei aus. Bei CSS 2.0 ist die Unterstützung zwar nahezu vollständig, aber teilweise fehlerhaft.

Besser ist die Unterstützung bei Netscape 7 und Mozilla 1.0 und höher. Hier wird auch der CSS 2.0-Standard besser unterstützt, das heißt es wurden einige Fehler behoben.

Mozilla 1.7+, Firefox 1.0+ und Camino 0.8+

Alle drei Browser beherrschen CSS 1.0 und CSS 2.0 nahezu fehlerfrei und unterstützen auch bereits Teile von CSS 2.1 und 3.0. Dies gilt insbesondere für alle Browser, die auf Mozilla 1.8+ basieren.

Safari, iCab und Konqueror

Diese drei Browser haben zwar einen verschwindend geringen Marktanteil, aber sie haben Potenzial, insbesondere Konqueror, der Standardbrowser der Linux KDE, und Safari, der eine Entwicklung von Apple ist und mittlerweile mit Mac OS X standardmäßig ausgeliefert wird.

Mit Mac OS X (Release 10.4.1) wird kein Internet Explorer für Mac mehr ausgeliefert. Dieser lässt sich zwar nachträglich bei Microsoft downloaden und installieren, da aber nun Safari 2.0 Standardbrowser ist, wird der Marktanteil von Safari zu- und entsprechend der des Internet Explorers für Macintosh abnehmen.

Für Safari 1.x+ und Konqueror 3.x+ gilt, dass beide Browser den CSS 1.0-Standard vollständig und annähernd fehlerfrei unterstützen. CSS 2.0 wird hingegen noch nicht vollständig und ohne die Sprachausgabe unterstützt. Beide Browser unterstützen aber schon CSS 2.1 und 3.0 in Teilen.

Bei iCab 2.9.x ist die CSS-Unterstützung noch sehr mangelhaft. Hier wird lediglich CSS 1.0 weitgehend vollständig dargestellt. CSS 2.0 zeigt vor allem bei den Befehlen für die absolute und relative Positionierung große Mängel. Dies ändert sich jedoch mit der Version 3.0 von iCab. In der Version 3.0 werden fast alle CSS 2.1-Befehle unterstützt und das nahezu fehlerfrei. Problematisch sind hier vor allem noch die Befehle für die Druckausgabe. Zum Teil erzeugt der Browser unleserliche Ausgaben, was allerdings auch ein Problem der BETA-Version sein kann oder unter Umständen an einer schlechten Kommunikation mit dem Druckertreiber liegt.

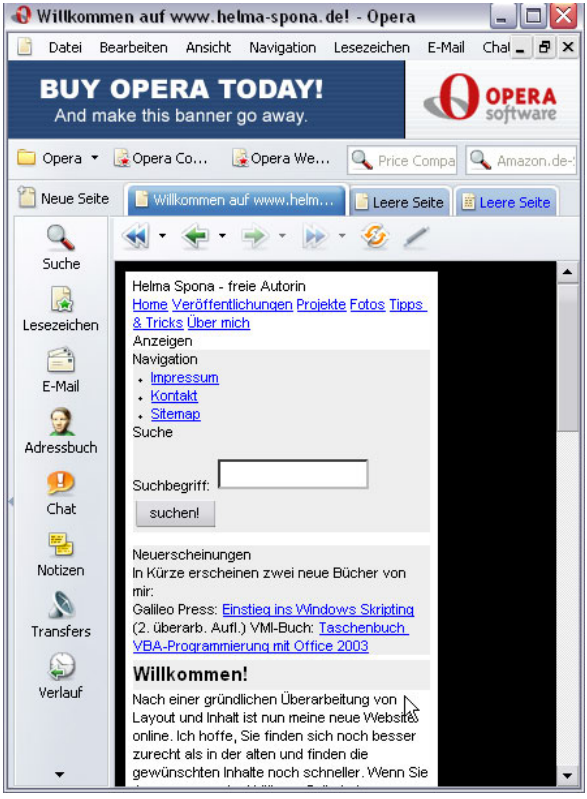
WAP und PDA-Browser

Neben den zuvor genannten Browsern für Computer gibt es noch die Gruppe der WAP- und PDA-Browser, die für Handys und Handhelds entwickelt wurden. Sie unterstützen oft nur Teile von HTML (wenn überhaupt) bzw. WML (der Markup-Sprache für WAP-Seiten). Dennoch spielen sie im Alltag eine immer größere Rolle, da insbesondere PDAs und Windows-CE-Geräte mit Internet-Fähigkeiten mittlerweile erschwinglich werden. Daher geht der Referenzteil des Buches auch beispielhaft auf diese Browser ein. Als Referenz wird hier der Palm Web Browser in der Version 2.0.x verwendet.

:-) TIPP

Einige aktuelle Handys benutzen auch den Opera-Mobile-Browser, der wie das PC-Pendant sehr gute CSS-Fähigkeiten aufweist. Wenn Sie wissen möchten, wie Ihre Webseite auf einem solchen Handy dargestellt wird, wählen Sie in Ihrem Opera-Browser (für Windows) VIEW / SMALL SCREEN bzw. ANSICHT / KLEINBILDSCHIRM aus. Der Opera-Browser zeigt dann die Seite an, wie sie auf dem Handy (mit dem Opera-Browser) angezeigt würde. Wenn Sie keine Stylesheets für Handhelds definiert haben, nutzt der Opera-Browser seine Technik SSR (Small Screen Rendering), um die Seite handytauglich zu formatieren. Dabei werden die Formatierungen auf ein Minimum reduziert.

Abbildung 1.7: Die »Handy-Vorschau« des Opera-Browsers



:-) TIPP

Wenn Sie Ihre Webseite auch mit einem PDA-Browser testen möchten, aber selbst nicht über einen PDA mit Browser bzw. Internetzugang verfügen, können Sie auch einen PDA-Browser-Simulator verwenden, den Sie auf der Seite `http://software.palmandmore.de` *von Palmandmore finden.*

Genauso gibt es auch Simulatoren von verschiedenen Handy-Herstellern, die die WAP-Emulation in ihren Handys simulieren.

Natürlich gibt es noch zahlreiche kleine Browser, die jedoch wegen der geringen Marktanteile vernachlässigt werden können. Wichtig ist einfach, dass Sie korrekten (X)HTML- und CSS-Code erzeugen. Dann sollte ein standardkonformer Browser die Seite ansprechend und korrekt darstellen.

1.3 Trennung von Layout und Inhalt

Das wichtigste Argument für den Einsatz von CSS ist die Möglichkeit, Inhalt und Layout bzw. Formatierung der Webseite voneinander zu trennen. Das erhöht die Wartungsfreundlichkeit, unterstützt die Barrierefreiheit der Seite und ergibt wesentlich übersichtlicheren, einfacheren und kürzeren HTML-Code. Das hat nicht nur für Sie als Entwickler Vorteile, sondern auch für den Browser.

Viele Browserabstürze resultieren nämlich aus fehlerhaftem oder zu komplexem Code der Seite. Das gilt vor allem für den Netscape Navigator 4-, der enorme Schwierigkeiten mit verschachtelten Tabellen und Framesets hat. Je einfacher Sie den Code gestalten, desto schneller und sicherer kann der Browser den Code darstellen und umso weniger Daten müssen an den Browser übertragen werden. Das reduziert also nicht nur die Übertragungszeit der Seite, sondern spart auch Kosten und Speicherplatz.

An einem einfachen Beispiel lernen Sie nachfolgend die HTML- und CSS-Grundlagen kennen, die in den folgenden Kapiteln weiter ausgebaut werden.

Wenn Sie über HTML-Kenntnisse verfügen, können Sie den Abschnitt »(X)HTML-Crashkurs« auslassen. Sie benötigen ihn nur dann, wenn Ihnen grundlegende Kenntnisse zur Struktur und den Elementen von (X)HTML-Seiten fehlen, weil Sie bisher einen WYSIWYG-Editor genutzt und sich weniger mit dem im Hintergrund erzeugten Code beschäftigt haben.

WYSIWYG ist die Abkürzung für »What You See Is What You Get«, was übersetzt werden kann mit: »Was du siehst, bekommst du«. Gemeint sind damit Editoren, bei denen Sie bereits zum Zeitpunkt des Entwurfs ein Ergebnis sehen, das weitgehend der Darstellung im Browser entspricht. Solche HTML-Editoren verfügen in der Regel über eine grafische Benutzeroberfläche, mit der Sie den Inhalt und Aufbau der Seite per Mausklick gestalten können. HTML-Kenntnisse sind bei diesen Editoren nicht zwingend erforderlich. Zu diesen Editoren gehören unter anderem Adobe® GoLive™, Macromedia Dreamweaver® und NetObjects Fusion.

(X)HTML-Crashkurs

HTML ist von Haus aus eine Beschreibungssprache, d.h., sie beschreibt den Aufbau (und ursprünglich auch die Formatierung) einer Webseite. Derzeit ist der HTML-Standard 4.1 aktuell, wenngleich auch Version 4.0 noch gängig ist und durchaus eingesetzt werden kann.

XHTML

XHTML ist eine Fortentwicklung von HTML, die jedoch im Gegensatz zu HTML auf XML (eXtensible Markup Language) basiert. Allerdings nutzt XHTML im Wesentlichen die gleichen Elemente wie HTML, und auch die Struktur eines XHTML-Dokuments unterscheidet sich nicht wesentlich von der eines HTML-Dokuments. Die vorhandenen Unterschiede sind bezüglich des Einsatzes von Cascading Style Sheets zu vernachlässigen.

HTML-Dateien sind Textdateien. Daher reicht es aus, wenn Sie einen einfachen Texteditor zum Bearbeiten des HTML-Codes verwenden. Sie können aber natürlich auch im Quellcode Ihres gewohnten HTML-Editors arbeiten.

Eine HTML-Datei erstellen

CD

Das Beispiel finden Sie als Datei crashkurs.htm *auf der Buch-CD im Verzeichnis* BSP/K01.

Erzeugen Sie mit Ihrem Editor eine neue HTML-Datei und speichern Sie diese mit der Dateinamenserweiterung *.html* oder *.htm* ab. Aktivieren Sie anschließend die Quellcodeansicht.

INFO

Falls Sie einen einfachen Texteditor wie den Windows-Editor verwenden, gibt es natürlich keine spezielle Quellcodeansicht. Mehr als Quellcode können Sie dann nicht anzeigen lassen. Dafür müssen Sie allerdings auch den nachfolgend erläuterten Code selbst eingeben, weil der Editor ihn natürlich nicht automatisch mit der Datei erzeugt.

Die Grundstruktur

Der Quellcode der erzeugten Datei sollte nun in etwa wie folgt lauten. Die erste Zeile, die mit `<!DOCTYPE` beginnt, legt den Dokumenttyp fest. Er hat unter anderem ganz erhebliche Auswirkungen darauf, wie der HTML-Code der Seite vom Browser dargestellt wird und welche Elemente im Code zulässig sind. Da die DocType-Angabe optional ist, gibt es auch Editoren, die diese Zeile nicht einfügen. In diesem Fall würde diese erste Zeile einfach ersatzlos entfallen.

In XHTML-Dateien ist diese Zeile jedoch Pflicht. Ferner steht dort als erste Zeile, also noch vor der DocType-Angabe, die Zeile `<?xml version="1.0"?>`*, die die XML-Version der XHTML-Datei und optional die Kodierung bestimmt. Diese Zeile wird als* XML-Prolog *bezeichnet.*

Mehr dazu erfahren Sie etwas weiter unten im Abschnitt »Der DocType der HTML-Datei«.

Mit `<html>` beginnt der eigentliche Inhalt der Seite. Dabei handelt es sich um einen Anfangs-Tag (auch öffnender Tag genannt), der zusammen mit dem obligatorischen End-Tag `</html>` (auch Abschluss-Tag genannt) ein Tag-Paar bildet. Tag-Paare sowie einzelne Tags, die nicht paarweise auftreten, werden als HTML-Elemente bezeichnet. In diesem Fall heißt das Element also `html` und besteht aus dem Anfangs-Tag `<html>` und dem End-Tag `</html>`. Innerhalb des Elements, also zwischen Anfangs- und End-Tag, steht der Inhalt der HTML-Seite.

Alle Tags haben einen einheitlichen Aufbau. Sie werden durch spitze Klammern `<` *und* `>` *begrenzt. Dazwischen steht der Tag-Name. Bei End-Tags steht zwischen der einleitenden spitzen Klammer und dem Tag-Namen noch ein Schrägstrich. Auch wenn das* `html`*-Element immer Inhalte hat, gibt es durchaus auch leere Elemente. Sie sehen beispielsweise das Element* `<p></p>` *im Listing. Dabei handelt es sich um einen Absatz ohne Inhalt. Solche leeren Elemente können Sie in XHTML-Dokumenten auch abkürzen, indem Sie nach dem Namen des öffnenden Tags ein Leerzeichen und anschließend einen Schrägstrich schreiben. Dann kann der separate End-Tag entfallen. Analog zu* `<p></p>` *in HTML könnten Sie in XHTML also* `<p />` *schreiben.*


```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01//EN">
<html>
    <head>
        <title>Crashkurs HTML</title>
    </head>
    <body>
        <p></p>
    </body>
</html>
```

Listing 1.1: Aufbau einer HTML-Seite

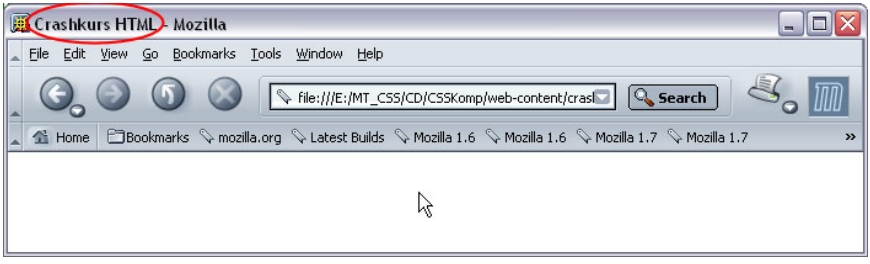
Innerhalb des `html`-Elements steht der Inhalt der Seite. Der besteht wiederum aus zwei Teilen. Der erste enthält die Daten des Dateikopfes, die mit wenigen Ausnahmen für den Betrachter der Seite nicht sichtbar sind. Dieser Teil steht innerhalb des `head`-Elements. Der zweite Teil ist das `body`-Element und dessen Inhalt. Dies ist der eigentliche, sichtbare Inhalt der Seite.

INFO

Der HTML- bzw. XHTML-Standard legt genau fest, welche Elemente im `head`*-Element und welche im* `body`*-Element stehen dürfen und müssen. Daher können Sie keineswegs innerhalb des* `head`*-Elements Absätze einfügen. Möchten Sie versteckten Text in der Seite definieren, können Sie dazu wahlweise* `meta`*-Elemente im* `head`*-Element einfügen oder auch normale Absätze mit Hilfe von* `p`*-Elementen innerhalb des* `body`*-Elements. Diese könnten Sie dann bei Bedarf mit Hilfe von CSS unsichtbar formatieren oder ausblenden.*

Innerhalb des `head`-Elements sollte mindestens das `title`-Element stehen. Es legt den Text für den Fenstertitel des Browserfensters fest.

Abbildung 1.8: Das Ergebnis des title-Elements in Mozilla



Das `body`-Element enthält in der Regel mindestens eine Überschrift und einen Absatz mit Text. Absätze definieren Sie mit dem `p`-Element, Überschriften hingegen mit den Elementen `h1`, `h2`, `h3`, `h4`, `h5` und `h6`. Das Element `h1` definiert eine Überschrift erster Ebene, `h2` eine Überschrift zweiter Ebene etc.. Alle Browser stellen diese Überschriften entsprechend der Hierarchietiefe in unterschiedlichen Schriftgrößen dar.

Der folgende Codeabschnitt definiert beispielsweise für die Seite eine Hauptüberschrift, zwei Absätze und eine Überschrift zweiter Ordnung. Das Ergebnis dieses Codes sieht dann wie in Abbildung 1.9 aus.

Listing 1.2: Den Inhalt der Seite festlegen

```
<body>
    <h1>Dies ist eine Überschrift erster Ordnung!
    Sie wird daher sehr groß angezeigt.</h1>
    <p>Untergeordnete Überschriften werden hingegen
    kleiner dargestellt, wie die folgende.</p>
    <h2>Überschrift zweiter Ordnung</h2>
    <p>Dies ist wieder ein Absatz mit Fließtext.</p>
</body>
```

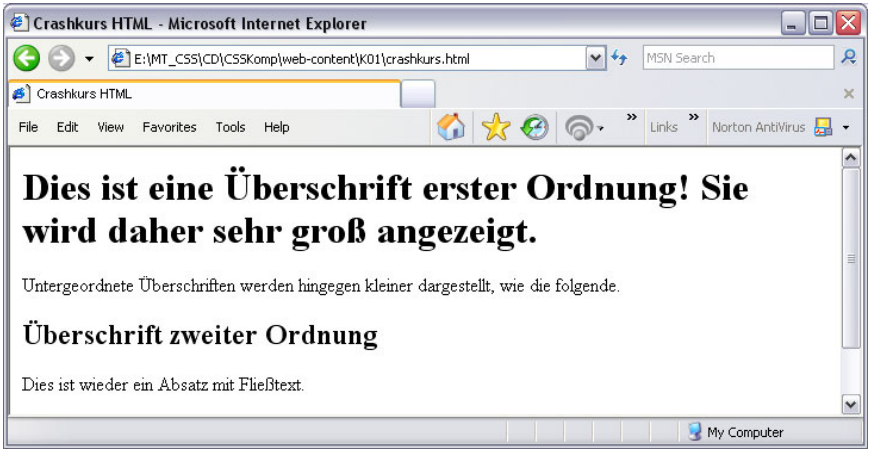


Abbildung 1.9: Darstellung der h1-, h2- und p-Elemente im Internet Explorer

HTML-Tags verschachteln

Wie Sie an dem vorstehenden kurzen Code schon sehen können, werden HTML-Tags verschachtelt. Das heißt, innerhalb eines Elements werden andere, untergeordnete Elemente definiert und darin können wiederum Elemente vorhanden sein. Die Verschachtelungstiefe ist dabei quasi unbeschränkt. Allerdings dürfen nicht alle Elemente auch untergeordnete Elemente enthalten und wenn, dann nicht beliebige, sondern oft nur bestimmte Elemente. Welche das sind, legt der HTML- bzw. XHTML-Standard fest.

Wichtig ist aber in jedem Fall, dass Sie die Elemente korrekt verschachteln. Dazu gibt es folgende einfache Regel.

Immer wenn Sie in einem Element ein untergeordnetes Element mit dem Anfangs-Tag öffnen, müssen Sie es auch innerhalb dieses Elements wieder mit dem End-Tag schließen.

!! STOP

Das bedeutet also ganz konkret, dass folgender Code aus mehreren Gründen fehlerhaft ist. Zwar wurde das `span`-Element korrekt innerhalb des `p`-Elements platziert, da es innerhalb des Absatzes geöffnet und auch wieder geschlossen wird, der Absatz selbst wird aber nicht innerhalb des `body`-Elements geschlossen, sondern außerhalb. Das ist der erste Fehler.

Der zweite Fehler besteht darin, dass der End-Tag des Absatzes innerhalb des `html`-Elements und nicht innerhalb des `body`-Elements geschlossen wird. Im `html`-Element sind nämlich ausschließlich die Elemente `head` und `body` zulässig. Jedes andere Element stellt einen Fehler dar.

Listing 1.3: Fehlerhaft verschachtelter Code

```
<html>
    <head>
    </head>
    <body>
        <p>Dies ist ein Absatz der ein
        <span>untergeordnetes Element </span>
        enthält.
    </body>
    </p>
</html>
```

Korrekt müsste der Code also wie folgt lauten. Damit wären beide Fehler behoben.

Listing 1.4: So ist es richtig!

```
<html>
    <head>
    </head>
    <body>
        <p>Dies ist ein Absatz der ein
        <span>untergeordnetes Element </span>
        enthält.
        </p>
    </body>
</html>
```

Das Element `span` *dient vornehmlich dazu, CSS-Formatierungen auf einen Teil eines Absatzes oder anderen Elements anzuwenden. Ohne Formatierungsangaben (CSS oder HTML) wird es im Browser nicht gesondert formatiert.*

Notwendige und optionale Attribute

HTML-Elemente können nicht nur Inhalte haben, sondern auch Attribute. Diese Attribute dienen dazu, die Elemente zu beschreiben, und helfen daher dem Browser, das Element korrekt darzustellen.

Attribute werden immer im Anfangs-Tag des Elements definiert, niemals im End-Tag. Sie bestehen in der Regel aus `Name=Wert`-Kombinationen. In HTML gibt es jedoch einige wenige Attribute, deren Name ohne Wert angegeben wird.

In XHTML ist das etwas anders. Hier müssen alle Attribute als `Name=Wert`*-Kombinationen angegeben werden.*

Generell gilt, dass alle Attributwerte in Anführungszeichen oder Hochkommata eingefasst werden. Die Syntax eines solchen Attributs lautet damit:

```
<element attributname="Wert"></element>
```

oder

```
<element attributname='Wert'></element>
```

Nicht jedes HTML-Element verfügt über Attribute. Für manche gibt der HTML-Standard mehrere Attribute vor, für andere gar keine. Darüber hinaus gibt es optionale Attribute und so genannte Pflichtattribute. Pflichtattribute müssen Sie angeben, um fehlerfreien HTML-Code zu erstellen, optionale Attribute können Sie auch weglassen.

Welche Attribute für welche HTML-Elemente obligatorisch oder optional sind, regelt wiederum der HTML-Standard.

Selbstverständlich kann ein Element auch mehrere Attribute enthalten. In diesem Fall trennen Sie die Attribut-Wert-Kombinationen durch ein Leerzeichen voneinander. Folgendes Beispiel demonstriert dies. Es definiert einen Hyperlink mit Hilfe des `a`-Elements.

```
<p>Links definieren Sie mit dem a-Element. Es verf&uuml;gt &uuml;ber mehrere
Attribute. Die wichtigsten sind href, name und title.
<a title="Zur&uuml;ck zur &Uuml;bersicht" href="../index.html">Home</a></p>
```

Listing 1.5: Attribute verwenden

Das `a`-Element legt das Ziel des Hyperlinks über das Attribut `href` fest. Hier verweist der Link auf die Startseite der Website, die sich im übergeordneten Verzeichnis befindet. Optional können Sie ein zweites Attribut angeben, nämlich `title`. Es bestimmt einen Text, den die meisten Browser als Tooltip für den Hyperlink anzeigen.

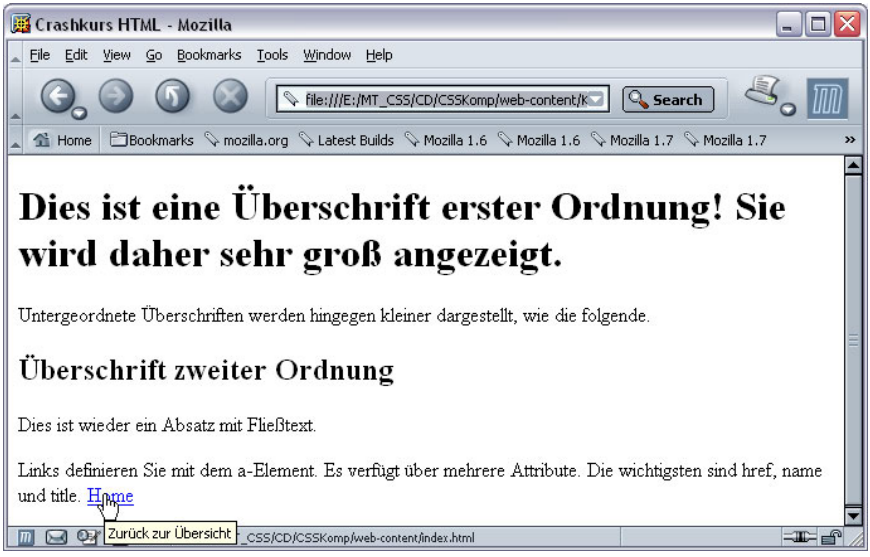


Abbildung 1.10: Der definierte Hyperlink mit Tooltip in Mozilla

Der DocType der HTML-Datei

Die DocType-Definition am Anfang einer (X)HTML-Datei legt fest, welche Version von HTML bzw. XHTML verwendet wird. Viele neuere, standardkonforme Browser richten sich danach und stellen die Dokumente abhängig von der angegebenen HTML- und XHTML-Version auch unterschiedlich dar, wenn der jeweilige Standard das erfordert.

Einen wirklich standardkonformen Browser gibt es leider immer noch nicht. Es gibt aber Browserhersteller, die sich die Standardkonformität auf die Fahne geschrieben haben. Deren Browser halten sich in der Regel recht eng an den Standard. Fehler sind natürlich dennoch nicht auszuschließen. Zu diesen Browsern gehören unter anderem Mozilla und der Opera-Browser und natürlich der W3C-eigene Editor Amaya.

Damit eine HTML- oder XHTML-Datei gemäß HTML 4.x bzw. XHTML 1.x-Standard gültig ist, müssen Sie eine DocType-Angabe machen. Sie ist jedoch nicht erforderlich, damit die Browser sie anzeigen können. Das liegt daran, dass es die DocType-Angabe in früheren HTML-Versionen noch nicht gab und alle Browser abwärtskompatibel sind.

Da die DocType-Angabe den verwendeten (X)HTML-Standard definiert, ist es auch erforderlich, dass sich der HTML-Code der Datei selbst daran hält. Ansonsten haben Sie wieder keine gültige (d.h. valide) (X)HTML-Seite.

Jede DocType-Angabe für HTML-Seiten hat einen ganz bestimmten Aufbau. Sie beginnt mit `<!DOCTYPE HTML PUBLIC`. Das sagt aus, dass es sich um eine öffentlich verfügbare DTD (Document Type Definition) handelt. Danach folgt die Spezifizierung der HTML-Version, die in Anführungszeichen eingefasst wird. Getrennt durch eine oder mehrere Leerzeichen folgt dann der URL, unter der die DTD zu finden ist.

Der URL ermöglicht es dem Browser oder jedem anderen Programm, das die HTML-Datei verarbeitet, die Syntax der Seite gemäß der angegebenen DTD zu prüfen.

Das können Sie sich zunutze machen, wenn Sie einen Editor verwenden, der die Validierung (Syntaxprüfung) der Datei gemäß der angegebenen DTD ermöglicht. Das sind beispielsweise Adobe® GoLive™ 6 und höher oder Macromedia Dreamweaver® MX 2004. Anhand der angegebenen DTD können Sie dann vom Editor die Syntax prüfen lassen, um Fehler im HTML-Code zu finden.

Aktuell ist HTML 4.01 der neuste HTML-Standard. Wenn es keinen Grund gibt, auf andere HTML-Versionen auszuweichen, sollten Sie daher immer diesen Standard verwenden. Dazu stehen für einfache HTML-Seiten (ohne Framesets) folgende DTDs zur Verfügung:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
 "http://www.w3.org/TR/html4/strict.dtd">
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://
www.w3.org/TR/html4/loose.dtd">
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Frameset//EN" "http://
   www.w3.org/TR/html4/frameset.dtd">
```

Listing 1.6: Mögliche DocType-Angaben für HTML

`Strict` (streng) bedeutet, dass keine Elemente und Attribute zulässig sind, die beispielsweise von HTML 4.01 Transitional noch aus Gründen der Abwärtskompatibilität geduldet werden. Während die DTD Transitional beispielsweise noch das `font`-Element (aus HTML 3.2) erlaubt, ist dies bei der DTD für HTML 4.01 Strict nicht mehr erlaubt. Die Frameset-DTD ist ausschließlich für Webseiten zu verwenden, die ein Frameset definieren. Sie sollten das jedoch besser vermeiden, weil Framesets nicht barrierefrei sind und verschiedene Nachteile für den Benutzer mit sich bringen.

Darüber hinaus ist auch eine Kurzform der DocType-Angabe zulässig, die einfach den URL der DTD weglässt. Die Angaben würden dann also entsprechend lauten:


```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN">
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" >
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Frameset//EN" >
```

Listing 1.7: DocType-Angaben für HTML 4.01

Dies ist aber nicht einfach nur eine Schreibersparnis, sondern wirkt sich auch auf die Darstellung im Browser aus.

Für XHTML-Dateien gibt es entsprechend folgende DocType-Angaben:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "http://
   www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://
www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Frameset//EN" "http://
   www.w3.org/TR/xhtml1/DTD/xhtml1-frameset.dtd">
```

Listing 1.8: DocType-Angaben für XHTML

Auf weitere Auswirkungen der angegebenen DTD kommt noch einmal Abschnitt »Anzeige-Modi« weiter unten zurück. Darüber hinaus gibt es natürlich auch DTDs für ältere HTML-Versionen, die Sie jedoch besser nicht mehr verwenden. Der Vollständigkeit halber finden Sie sie jedoch im Anhang aufgelistet.

Webseiten strukturieren und formatieren

Eine Webseite bekommt ihre Struktur dadurch, dass Sie die HTML-Elemente gemäß ihrem Zweck einsetzen. Das bedeutet, Sie verwenden ein `h1`-Element wirklich nur, um eine Überschrift erster Ordnung festzulegen und nicht um den Text einfach größer anzeigen zu lassen, ohne dass es sich dabei um eine Überschrift handelt.

Auf diese Weise können die darstellenden Programme Wichtiges von Unwichtigem unterscheiden. Vor allem für Screenreader und Braillezeilen ist dies wichtig, weil der Benutzer so schnell über die Überschriften den Inhalt der Seite überfliegen kann, ohne alles von vorne bis hinten »lesen« zu müssen.

Struktur bedeutet auch, dass Sie die vorhandenen Möglichkeiten von (X)HTML nutzen. Möchten Sie beispielsweise Auflistungen erstellen, missbrauchen Sie dazu keine Absätze (`p`-Element), sondern nutzen Sie die Listenelemente `ul` und `ol`. Möchten Sie beispielsweise Quellcode auf Ihrer Seite darstellen, fassen Sie den in das Element `code` ein, anstatt des `p`-Element zu formatieren. Gefallen Ihnen die Standardformatierungen für das Element nicht, können Sie es immer noch per CSS formatieren und zwar viel gezielter, als wenn Sie ein `p`-Element missbrauchen.

Logische Auszeichnungen

Um Code zu strukturieren, stellt HTML so genannte logische Textauszeichnungen zur Verfügung. Dazu gehört beispielsweise das `h1`-Element. Logisch deshalb, weil es nicht festlegt, wie ein Browser die Überschrift ausgibt, sondern nur, dass es sich bei dem Inhalt um eine Überschrift handelt. Zu diesen logischen Auszeichnungen gehören auch das `p`-Element, das `code`-Element zur Ausgabe von Quellcode und das `li`-Element zur Definition von Listeneinträgen.

Darüber hinaus kennt HTML auch physische Textauszeichnungen. Sie bestimmen, wie der Browser das Element anzeigen soll. Zu diesen Auszeichnungen gehört beispielsweise das Element `b`, dessen Inhalt fett (bold) dargestellt werden soll. Physische Auszeichnungen sagen nichts über die Struktur der HTML-Seite aus.

Folgendes Beispiel soll dies verdeutlichen. Nehmen Sie an, Sie möchten eine Webseite erstellen, bei der es um Programmierung, CSS oder HTML geht. In diesem Fall müssen Sie in der Regel auch Code in der Seite ausgeben.

Den Code finden Sie in der Datei Struktur.html *im Verzeichnis* BSP/K01.

Ohne logische Textauszeichnungen und CSS könnte das beispielsweise wie folgt aussehen: Sie würden den kompletten Inhalt als Absätze mit Hilfe des `p`-Elements definieren. Damit dann die Überschrift anders formatiert wird als der Code und die normalen Absätze, würden Sie das `font`-Element sowie weitere physische Auszeichnungen verwenden. Der dazu notwendige Code ist im Listing fett dargestellt, damit Sie sehen, wie viel überflüssigen Code Sie dadurch erzeugen.

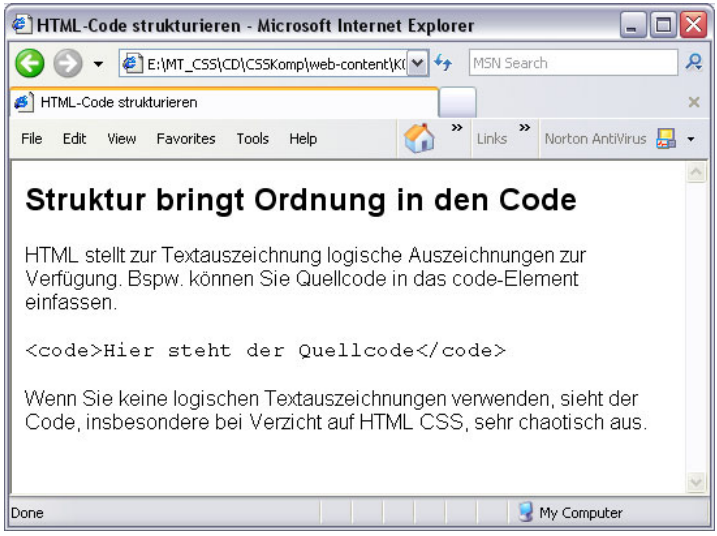


Abbildung 1.11: Die formatierte Seite im Internet Explorer

```
<body>
    <p><font size="5" face="Helvetica,
    Geneva, Arial, SunSans-Regular, sans-
    serif"><b>Struktur bringt Ordnung in den
    Code</b></font></p>
    <p><font face="Helvetica, Geneva, Arial,
    SunSans-Regular, sans-serif">HTML stellt
    zur Textauszeichnung logische Auszeichnungen
    zur Verf&uuml;gung. Bspw. k&ouml;nnen Sie
    Quellcode in das code-Element einfassen.
    </font></p>
    <p><font face="monospace">&lt;code&gt;Hier
    steht der Quellcode&lt;/code&gt;</font></p>
    <p><font face="Helvetica, Geneva, Arial,
    SunSans-Regular, sans-serif">Wenn Sie
    keine logischen Textauszeichnungen
    verwenden, sieht der Code, insbesondere bei
    Verzicht auf HTML CSS, sehr chaotisch aus.
    </font></p>
</body>
```

Listing 1.9: Formatierung mit physischen Textauszeichnungen

Wie Sie sehen, müssen Sie auf diese Weise für jeden einzelnen Absatz die Schriftart und gegebenenfalls die Schriftgröße festlegen. Stellen Sie sich vor, Sie wollen später die Schriftart mal ändern. Dann müssen Sie das für jeden Absatz, jede Überschrift etc. machen.

Viel gewonnen ist schon, wenn Sie logische Textauszeichnungen verwenden. Das heißt, für die Überschrift nutzen Sie das `h1`-Element und für den Code das `code`-Element.

Die optimierte Version des Beispiels finden Sie in der Datei struktur_opt.html *im Verzeichnis* BSP/K01.

Bei logischen Auszeichnungen können beispielsweise alle `font`- und `b`-Elemente entfallen. Das Element `b` deshalb, weil Überschriften erster Ordnung automatisch fett formatiert werden, die `font`-Elemente, weil Sie die Schriftformatierungen besser per CSS definieren können. Der Code ist dann schon sehr viel kürzer als vorher.

Listing 1.10: Optimierter Code für den Einsatz von CSS

```
<body>
    <h1>Struktur bringt Ordnung in den
    Code</h1>
    <p>HTML stellt zur Textauszeichnung logische Auszeichnungen zur
    Verf&uuml;gung. Bspw. k&ouml;nnen Sie Quellcode in das code-
    Element einfassen.</p>
    <p><code>&lt;code&gt;Hier steht der Quellcode&lt;/code&gt;
    </code></p>
    <p>Wenn Sie keine logischen Textauszeichnungen
    verwenden, sieht der Code, insbesondere bei
    Verzicht auf HTML CSS, sehr chaotisch aus.</p>
</body>
```

Das Ergebnis im Browser kommt dem Ziel schon recht nahe. Der Code wird tatsächlich in einer Nichtproportionalschrift dargestellt und die Überschrift größer und fett angezeigt.

Was nun noch fehlt, ist die Anpassung der Schriftart. Und genau hier kommt CSS ins Spiel.

Formatieren mit CSS

CSS-Code können Sie auf dreierlei Weise nutzen. Sie erstellen innerhalb der HTML-Seite Stile, die Sie bestimmten oder allen Elementen der Website zuweisen. Alternativ können Sie diese Stile auch in eine externe Datei auslagern. Die dritte Möglichkeit besteht darin, die CSS-Formatierung innerhalb des `style`-Attributs des zu formatierenden Elements einzufügen. In diesem Fall geben Sie alle CSS-Formatierungen als Wert für das `style`-Attribut an.

Abbildung 1.12: Das Zwischenergebnis im Internet Explorer

Generell kann jedes Element innerhalb des `body`-Elements wie auch das `body`-Element selbst ein `style`-Attribut haben. Notieren Sie beispielsweise im `style`-Attribut des `body`-Elements die CSS-Anweisung `font-family:sans-serif`:

```
<body style="font-family:sans-serif">
...
</body>
```

Listing 1.11: Formatieren mit dem style-Attribut

Das bewirkt, dass alle Elemente innerhalb des `body`-Elements, die nicht selbst über eine Einstellung für die Schriftart verfügen, nun in einer serifenlosen Schrift formatiert werden, wobei `sans-serif` eine so genannte generische Schriftart ist. Das bedeutet, der Browser, der die Seite darstellt, entscheidet darüber, welche Schrift verwendet wird. Dabei gibt der Name der generischen Schriftart (hier `sans-serif`) an, dass eine serifenlose Schrift genutzt wird. Das sind beispielsweise Schriften wie Arial, Helvetica etc.

Alle CSS-Formatierungen werden in Form von `Name:Wert`-Paaren notiert, die durch Semikolons voneinander getrennt werden. Sie könnten also neben der Eigenschaft `font-family`, die die Schriftfamilie definiert, auch weitere Eigenschaften, wie beispielsweise die Schriftfarbe festlegen. Diese Eigenschaft mit ihrem Wert fügen Sie dann einfach hinten dran, beispielsweise so:

```
style="font-family:sans-serif; color:red"
```

In diesem Fall würde die Schrift rot dargestellt.

Wenn Sie viele verschiedene Formatierungen festlegen möchten, ist die Sache mit dem `style`-Attribut aber nicht mehr die richtige Lösung. Sie müssten dann nämlich ganz viele `style`-Attribute definieren und diese

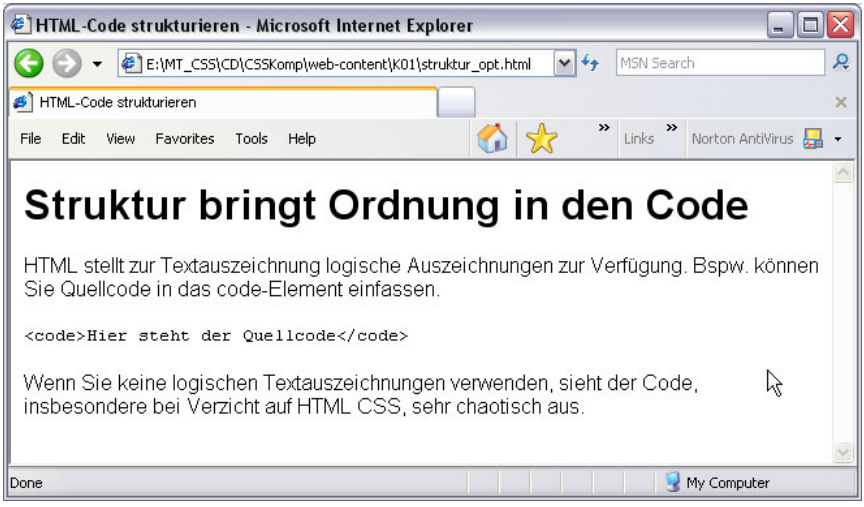


Abbildung 1.13: Das Ergebnis. Mit nur wenigen Zeichen CSS-Code wurde die Schriftart umgestellt und das für alle Absätze und Überschriften der Seite.

natürlich auch einzeln ändern, wenn Sie die Formatierung ändern möchten. Das funktioniert dann auch nicht viel besser als bei den `font`-Elementen. In diesem Fall bietet es sich an, Stile zu definieren. Die fügen Sie innerhalb der Seite ein, indem Sie im `head`-Element ein `style`-Element erstellen. Innerhalb dieses Elements definieren Sie dann die Stile.

Es gibt mehrere verschiedene Typen von Stilen. Die wichtigsten sind Elementstile, oft auch Element-Selektoren genannt. Dies ist allerdings nicht ganz richtig. Nur der Teil des Stils vor den geschweiften Klammern ist ein Selektor. Folglich besteht ein Elementstil immer aus dem Element-Selektor und den Formatierungen. Vielfach werden in der Literatur jedoch die Begriffe Stil und Selektor gleichgesetzt.

Elementstile werden automatisch dem HTML-Element zugewiesen, für das sie definiert wurden. Dieses Element legen Sie über den Namen des Element-Selektors fest. Möchten Sie beispielsweise einen Stil für das `body`-Element der Seite definieren, heißt der Selektor `body`. Dem Namen folgt ein geschweiftes Klammerpaar, innerhalb dessen Sie die CSS-Eigenschaften für den Stil angeben.

:-) TIPP

Innerhalb einer Stildefinition dürfen Sie auch Zeilenumbrüche einfügen, um den CSS-Code übersichtlicher zu gestalten. Üblicherweise wird immer genau eine Eigenschaft pro Zeile geschrieben.

```
    <head>
        <title>HTML-Code strukturieren</title>
        <style type="text/css">
        <!--
            body {
                font-family:sans-serif;
                color:red
            }
        -->
        </style>
    </head>
    <body>
...
    </body>
```

Listing 1.12: Definieren von CSS-Stilen

Wenn Sie vorstehenden Code in die Datei eingefügt haben, können Sie auch das `style`-Attribut wieder löschen. Diese Definition wird dann automatisch auf das `body`-Element und alle von diesem erbenden Elemente angewendet.

:-) TIPP

Sie sollten den Inhalt des `style`*-Elements immer in HTML-Kommentarzeichen* `<!--` *und* `-->` *einfassen. Das beugt Problemen in Browsern vor, die kein CSS beherrschen, da sie den Inhalt des* `style`*-Elements dann als Kommentar auffassen und nicht darstellen.*

Selbstverständlich können Sie in einem solchen `style`-Element auch mehrere Stile definieren. Die geben Sie dann einfach untereinander an. Mit dem folgenden, ergänzten Listing können Sie beispielsweise dafür sorgen, dass die Inhalte des `code`-Elements in grauer Schrift dargestellt werden.

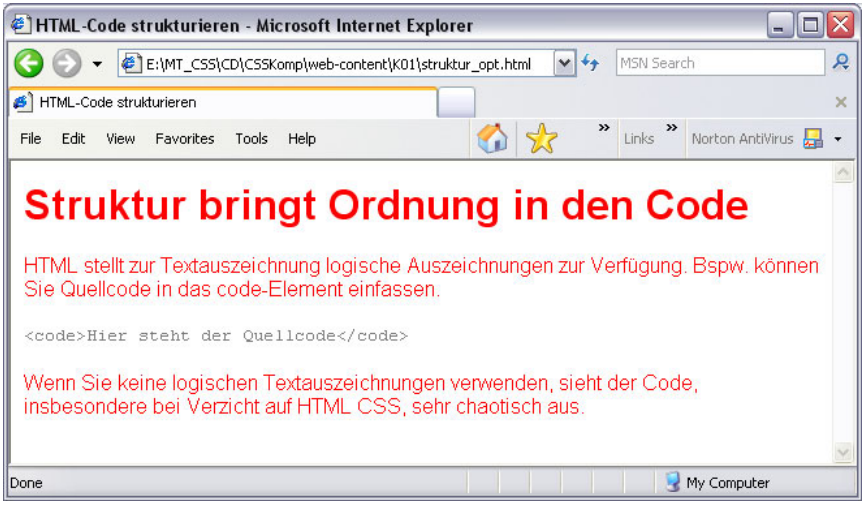


Abbildung 1.14: Das Ergebnis der Formatierung mittels Element-Selektoren

Listing 1.13: Mehrere Stile definieren

```
<style type="text/css">
    <!--
        body {
            font-family:sans-serif;
            color:red
        }
        code {
            color:gray
        }
    -->
    </style>
```

Damit haben Sie einen kurzen Überblick über den Nutzen und die Anwendung von CSS bekommen. Das ist aber natürlich nicht alles. CSS ist äußerst komplex und flexibel einsetzbar. Mehr zu Stilen, deren Anwendung und Nutzung beschreibt Kapitel 2, »Stile definieren«.

1.4 CSS in der Praxis

Gerade durch die Komplexität und den Umfang der CSS-Standards ist die Anwendung in der Praxis nicht ganz so einfach, wie es zunächst scheint. Das große Problem, bei allem Fortschritt in den letzten Jahren, ist der Test Ihrer Webseite in vielen verschiedenen Browsern. Da die Browser CSS immer noch nicht gleich gut unterstützen und zudem auch Fehler machen, müssen Sie wohl oder übel durch entsprechende Tests und Browserweichen sicherstellen, dass zumindest die neueren Versionen von Opera-Browser, Internet Explorer, Mozilla/Netscape und Safari die Seiten akzeptabel darstellen, und dazu ist es unerlässlich, dass Sie die Seite testen.

CSS- und HTML-Code im Browser testen

Nicht nur der CSS-Code einer HTML-Seite wirkt sich darauf aus, ob die Browser sie korrekt darstellen können, sondern auch der verwendete HTML-Code. Damit standardkonforme Browser eine Seite optimal anzeigen können, sollten Sie sicherstellen, dass Sie sowohl validen CSS- wie validen (X)HTML-Code erzeugen.

Valide ist der Code dann, wenn er dem jeweiligen W3C-Standard genügt. Valider HTML-Code muss also dem Standard entsprechen, der mit der DocType-Angabe definiert wurde, valider CSS-Code entsprechend der CSS 1.0 oder 2.x-Spezifikation. Um zu prüfen, ob der Code valide ist, können Sie so genannte Validatoren einsetzen. Diese prüfen den Code gemäß der verwendeten Standards und markieren Fehler bzw. führen sogar Verbesserungen durch.

Mehr zur Validierung des Codes erfahren Sie etwas weiter unten im Abschnitt »Validität von CSS und HTML prüfen«.

REF

Aber auch bei validem Code können Browser die Seite unterschiedlich anzeigen. Sie sollten daher Ihre Website immer mit mindestens den folgenden Browsern prüfen.

- Internet Explorer 4-7 für Windows
- dem Opera-Browser 7 oder 8 (bzw. die neueste Version und die Vorversion)
- Mozilla/Netscape oder Firefox in der aktuellen und einer älteren Version

Falls Sie die Möglichkeit haben und über einen Mac verfügen, sollten Sie ebenfalls noch Safari und den Internet Explorer für Mac verwenden. Für Linux bietet sich ein Test in Konqueror an.

Auch Amaya, der W3C-Browser, bietet sich zum Testen an. Für die Praxis ist er allerdings nicht relevant, da ihn kaum jemand zum Surfen verwendet. Er hat jedoch den Vorteil, dass er sich eng an alle W3C-Standards hält und nicht standardkonformen Code einfach ignoriert. So zeigt er beispielsweise bei Verwendung der DocType-Angabe für HTML 4.01 keine Formatierungen an, die Sie über die `font`*-Elemente vornehmen. Daher wird die erste Variante des Beispiels in Amaya gänzlich ohne Formatierungen angezeigt. Amaya ist allerdings auch kein Browser im herkömmlichen Sinn, sondern stellt neben der Anzeigefunktion für Webseiten auch Editierfunktionen zur Verfügung, sowohl für CSS als auch für HTML und XML.*

:-) TIPP

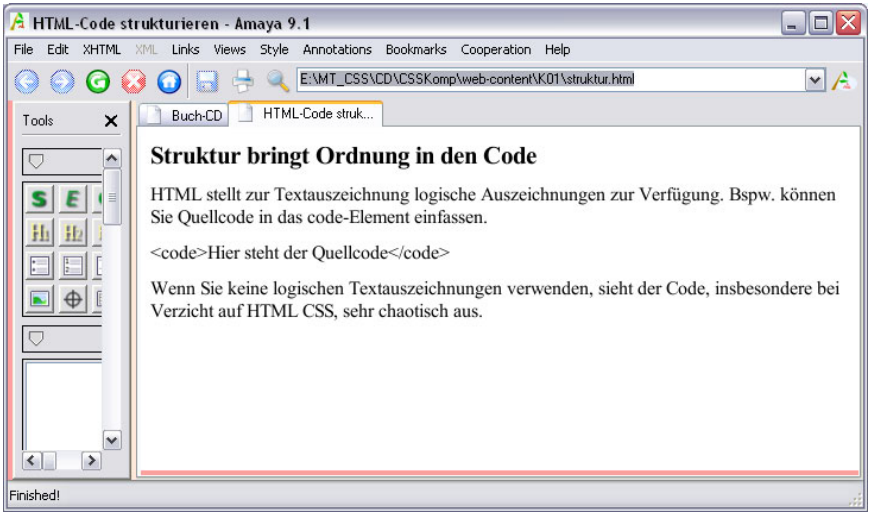


Abbildung 1.15: Darstellung der ersten Variante in Amaya. Nicht standardkonforme Elemente werden einfach ignoriert.

Sie bekommen Amaya kostenlos zum Download auf der W3C-Website unter dem URL `http://www.w3.org/Amaya/`.

Fast alle Browser lassen sich eigentlich problemlos in mehreren Versionen auf einem Rechner installieren, einzig der Internet Explorer macht da Schwierigkeiten. Sie können ihn immer nur in einer Version installieren. Zum Testen in mehreren Versionen benötigen Sie also zwingend mehrere Rechner oder zumindest mehrere Windows-Installationen auf einem System.

Es gibt allerdings auch hier eine Alternative. Alle neueren Internet Explorer und die meisten anderen Browser verfügen über so genannte Anzeige-Modi, die Sie über eine entsprechende DocType-Angabe aktivieren können.

Anzeige-Modi

Wie bereits mehrfach erwähnt, hat die DTD nicht nur Auswirkungen auf die zur Verfügung stehenden HTML-Elemente und deren Attribute, sondern auch darauf, wie Browser den HTML-Code und vor allem den enthaltenen CSS-Code darstellen. Das bezieht sich unter anderem auch auf fehlerhafte Darstellungen älterer Browser.

Alle großen Browser (der Opera-Browser, Internet Explorer und Mozilla) sowie Safari stellen dazu mindestens zwei verschiedene Anzeige-Modi zur Verfügung, den so genannten Quirks-Modus und den Standard-Modus. Diese können Sie über entsprechende DocType-Angaben ein- bzw. ausschalten.

Anzeige-Modus ermitteln

In welchem Modus ein Browser die Seite ausführt, können Sie mit Hilfe von JavaScript ermitteln. Das setzt aber natürlich voraus, dass der Browser JavaScript ausführt, d.h. dass JavaScript aktiviert ist und der Browser JavaScript kann.

Sie finden das Beispiel als anzeigemodus.html *auf der Buch-CD im Verzeichnis* BSP/K01.

Unter diesen Voraussetzungen fügen Sie einfach folgenden Code in das `body`-Element der Seite ein. JavaScript gibt dann den entsprechenden Modus aus.

Listing 1.14: Anzeige-Modus mit Hilfe von JavaScript ermitteln

```
<body >
    <script type="text/javascript">
    <!--
        document.write ('<p>Anzeigemodus: ' +
                        document.compatMode + '</p>');
    //-->
    </script>
</body>
```

Browser, die keinen Anzeige-Modus unterstützen, aber JavaScript ausführen, geben als Wert für die Eigenschaft `compatMode` *den Wert »undefined« aus.*

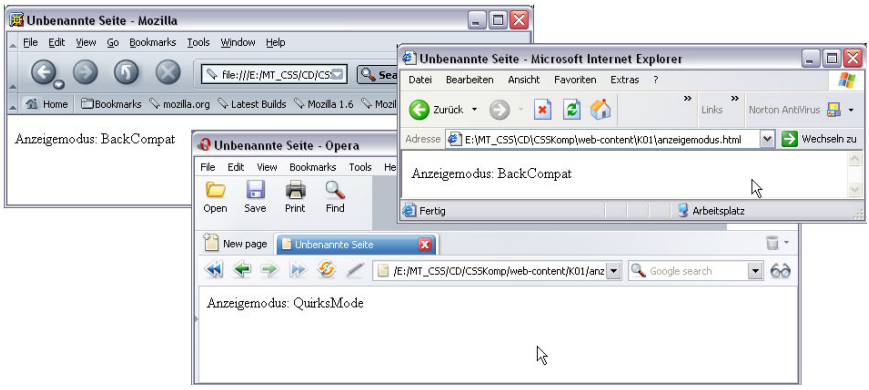


Abbildung 1.16: Die erzeugte Ausgabe in Mozilla, dem Opera-Browser und Internet Explorer

Im Quirks-Modus verhält sich der Browser wie eine ältere Version, die bestimmte Fehler im Quellcode ignoriert und veraltete HTML-Tags und -Attribute kennt und darstellt. Der Quirks-Modus wird von JavaScript (abhängig vom Browser) mit `QuirksMode` oder `BackCompat` ausgegeben.

Im Standard-Modus (Standards-Compliant Mode) verhält sich der Browser weitgehend gemäß den W3C-Standards, sowohl was HTML betrifft als auch was CSS betrifft. Der Standard-Modus wird von JavaScript mit der Zeichenkette `CSS1compat` ausgegeben.

In Mozilla 1.0.1 und 1.1 (keine höheren Versionen mehr) kennen die Mozilla-Browser einen dritten Modus, nämlich den Almost Standard Mode. Der Unterschied zum Standard-Modus ist nicht sehr groß und beschränkt sich auf die unterschiedliche Berechnung von Zeilenhöhen in Tabellen.

Modus aktivieren

Welchen Modus Sie aktivieren, hängt von der verwendeten DocType-Angabe und eventuell einem vorhandenen XML-Prolog ab. Mit der passenden DTD können Sie also ganz gezielt einen bestimmten Modus aktivieren. Dabei hängt der Modus nicht nur von der gewählten XHTML oder HTML-Version ab, sondern auch davon, ob ein XML-Prolog oder ein URI angegeben wurde.

Die folgende Tabelle gibt Ihnen einen Überblick über das Verhalten der entsprechenden Browser. Dabei stellt »S« den Standard-Modus, »Q« den Quirks-Modus und »A« den Almost Standard Mode dar.

Tabelle 1.2: Zusammenhang zwischen DocType-Angabe und Anzeige-Modus

DocType-Angabe			Browser						
HTML/XML-Version	**URI**	**XML-Prolog**	**IE 5.x (Mac)**	**IE 6+ (Win) Opera 7.0 - 7.03**	**Mozilla 0.6+ N 6.0+**	**Mozilla 1.1 und 1.01 N 7**	**Opera 7.1+**	**Safari 0.82+**	**Mozilla >1.1 N 7.1+**
Kein			Q	Q	Q	Q	Q	Q	Q
Unbekannt			Q	S	S	S	S	S	S
Unbekannt	●		Q	S	S	S	S	S	S
HTML 2.0			Q	Q	Q	Q	Q	Q	Q
HTML 3.2			Q	Q	Q	Q	Q	Q	Q
HTML 3.2	●		Q	Q	Q	Q	Q	Q	Q
HTML 4.0 Strict			S	S	S	S	S	S	S
HTML 4.0 Strict	●		S	S	S	S	S	S	S
HTML 4.0 Transitional			Q	Q	Q	Q	Q	Q	Q
HTML 4.0 Transitional	●		S	S	Q	Q	S	Q	Q
HTML 4.0 Frameset			Q	Q	Q	Q	Q	Q	Q
HTML 4.0 Frameset	●		S	S	Q	Q	S	Q	Q
HTML 4.01 Strict			S	S	S	S	S	S	S
HTML 4.01 Strict	●		S	S	S	S	S	S	S
HTML 4.01 Transitional			Q	Q	Q	Q	Q	Q	Q
HTML 4.01 Transitional	●		S	S	S	A	S	Q	S
HTML 4.01 Frameset			Q	Q	Q	Q	Q	Q	Q

DocType-Angabe			Browser						
HTML/XML-Version	URI	XML-Prolog	IE 5.x (Mac)	IE 6+ (Win) Opera 7.0 - 7.03	Mozilla 0.6+ N 6.0+	Mozilla 1.1 und 1.01 N 7	Opera 7.1+	Safari 0.82+	Mozilla >1.1 N 7.1+
HTML 4.01 Frameset	●		S	S	S	A	S	Q	S
XHTML 1.0 Strict	●		S	S	S	S	S	S	S
XHTML 1.0 Transitional	●		S	S	S	A	S	Q	S
XHTML 1.0 Frameset	●		S	S	S	A	S	Q	S
XHTML 1.0 Strict	●	●	S	Q	S	S	S	S	S
XHTML 1.0 Transitional	●	●	S	Q	S	A	S	Q	S
XHTML 1.0 Frameset	●	●	S	Q	S	A	S	Q	S
XHTML 1.1			S	S	S	S	S	S	S
XHTML 1.1		●	S	Q[a]	S	S	S	S	S

Tabelle 1.2: Zusammenhang zwischen DocType-Angabe und Anzeige-Modus (Forts.)

a. Genauso verhalten sich die angegebenen Browser auch, wenn vor einer DocType-Angabe in einer HTML-Seite ein Kommentar steht.

:-) TIPP

Optimal wäre es, wenn Sie den Browser dazu bewegen, möglichst den Standard-Modus zu verwenden. Dann können Sie relativ sicher sein, dass alle neueren Browser die Seite (fast) gleich darstellen. Der Standard-Modus ist auf jeden Fall gewährleistet, wenn Sie

- *XHTML 1.1 ohne XML-Prolog*
- *XHTML 1.0 Strict mit URL und ohne XML-Prolog*
- *HTML 4.0 Strict mit/ohne URL*
- *HTML 4.01 Strict mit/ohne URL*

angeben.

Unterschiede in der Darstellung

Vielleicht fragen Sie sich jetzt, was der Anzeige-Modus mit dem Test der Seiten in verschiedenen Internet Explorer-Versionen zu tun hat? Nun, das ist ganz einfach. Im Internet Explorer ist vor allem bei der Windows-Version 5.x und früher das fehlerhafte Boxmodell für Anzeigefehler verantwortlich. Indem Sie also die Seite im Quirks-Modus testen, können Sie prüfen, wie in diesen Browsern die Seite in etwa aussehen würde, da der IE 6 im Quirks-Modus auch das fehlerhafte Boxmodell simuliert. Die auf Mozilla basierenden Browser und die Opera-Browser sind im Quirks-Modus wesentlich fehlertoleranter und führen auch fehlerhaften CSS-Code noch aus.

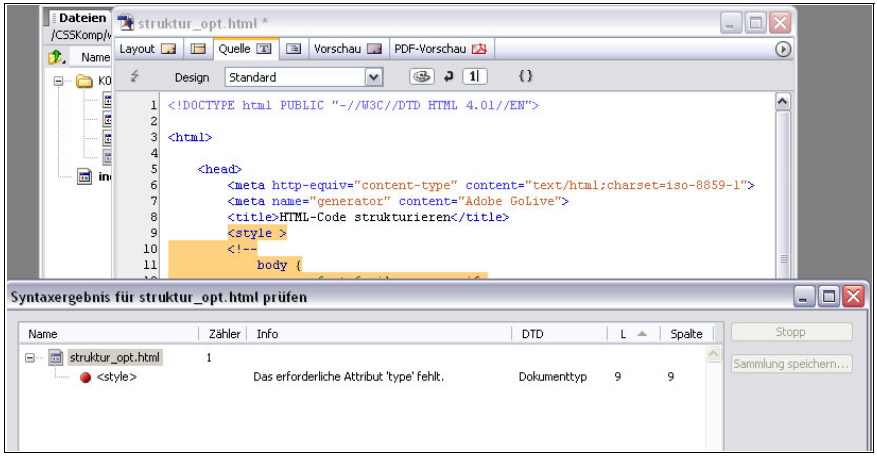
:-) TIPP

Wenn Sie validen CSS-Code erstellen möchten, fallen Fehler daher am ehesten auf, wenn Sie den Standard-Modus nutzen.

Validität von CSS und HTML prüfen

Die Gültigkeit von HTML-Code zu prüfen, ist für viele große HTML-Editoren kein Problem mehr. Sowohl Adobe® GoLive™ (ab Version 6.0) als auch Macromedia Dreamweaver MX 2004 beherrschen die Syntaxprüfung gemäß der angegebenen DTD. Vor allem bei GoLive werden Sie sehr schön auch auf fehlerhafte Attributwerte hingewiesen und die Fehlerstelle wird im Code farbig unterlegt.

Abbildung 1.17: Anzeige von Syntaxfehlern in GoLive CS



GoLive ist sogar in der Lage, dem Programm unbekannte DTDs mit dem angegebenen URL aus dem Internet zu laden und auch diese zu prüfen.

INFO

W3C-Validator

Neben HTML-Editoren, die die Validierung der (X)HTML-Seiten beherrschen, gibt es aber zahlreiche Programme und Online-Tools. Das wichtigste ist sicherlich der W3C-Validator.

Den W3C-Validator können Sie am besten online nutzen. Sie finden ihn unter dem URL `http://validator.w3.org/`*. Sie können dort eine bereits online verfügbare Seite prüfen lassen, indem Sie deren URL eingeben. Alternativ können Sie jedoch auch eine Datei hochladen, die dann geprüft wird.*

REF

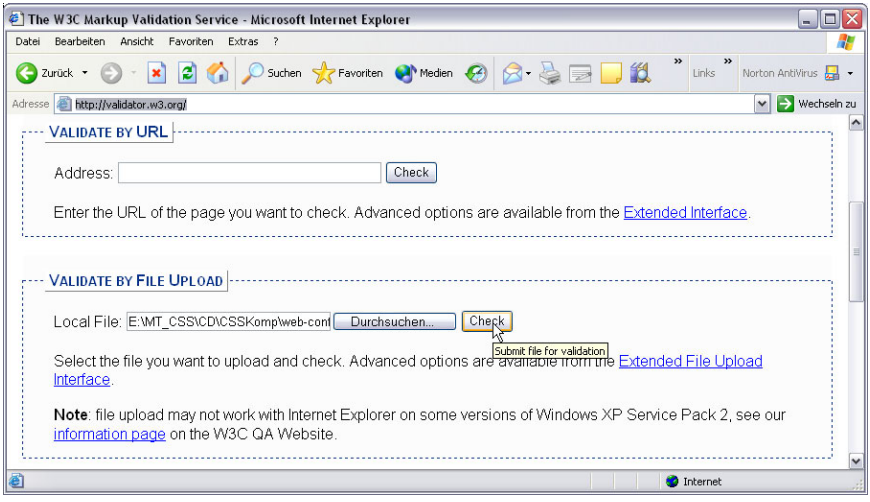


Abbildung 1.18: Der HTML-Validator des W3C

Hat der Validator die Prüfung durchgeführt, werden entweder die Fehler aufgelistet, so dass Sie sie beseitigen können, oder Ihnen wird gemeldet, dass die Seite valide ist.

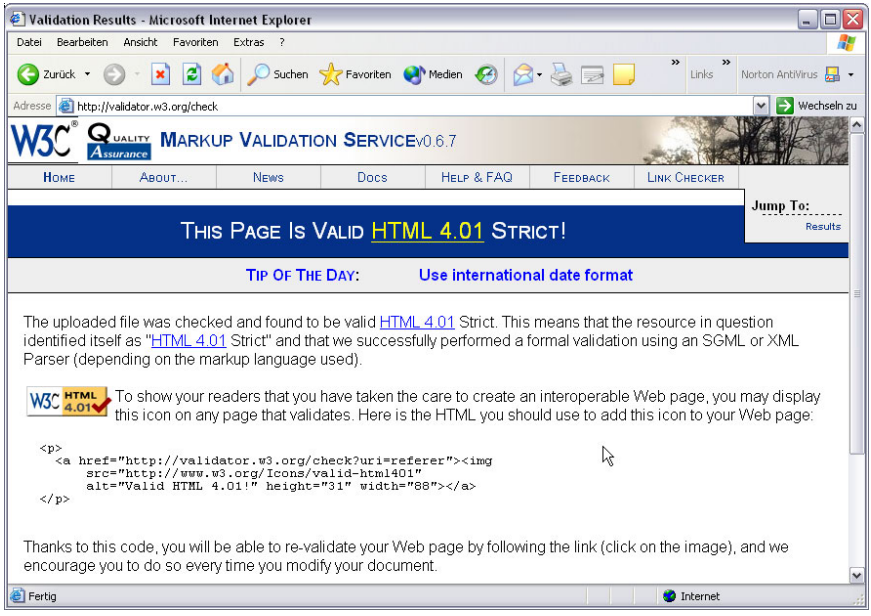


Abbildung 1.19: So meldet der Validator die Validität der Seite.

Tidy

Tidy ist ein kleines Programm, das es Ihnen nicht nur ermöglicht, Ihre Webseiten auf korrekte Syntax zu prüfen, sondern das bestimmte Fehler wie falsch verschachtelte Tags auch gleich behebt.

Sie können Tidy von Sourceforge.Net herunterladen. Der Link dazu lautet: `http://tidy.sourceforge.net`.

Tidy steht für alle wichtigen Betriebssysteme von Windows über Mac OS, Linux, Unix bis hin zu OS/2 zur Verfügung und arbeitet mit verschiedenen Editoren wie beispielsweise emacs und vim zusammen. Für Windows und Mac stehen auch Versionen mit grafischer Benutzeroberfläche zur Verfügung, die die Bedienung etwas vereinfacht. Eine davon ist TidyUI, die denkbar einfach zu bedienen ist.

Wenn Sie eine Seite mit TidyUI prüfen und korrigieren möchten, starten Sie zunächst einfach das Programm und laden dann die Datei über FILE / OPEN / HTML. TidyUI prüft die Datei sofort und zeigt die korrigierte Datei in einer separaten Registerkarte TIDY an. Wie und was korrigiert werden soll, können Sie auf der linken Seite über die Register einstellen. Standardmäßig werden die gröbsten Fehler automatisch behoben und beispielsweise fehlende Pflichtattribute ergänzt.

Wenn Sie die korrigierte Datei speichern möchten, gehen Sie zu dem Menü FILE / SAVE AS / HTML und wählen dann Verzeichnis und Dateiname aus.

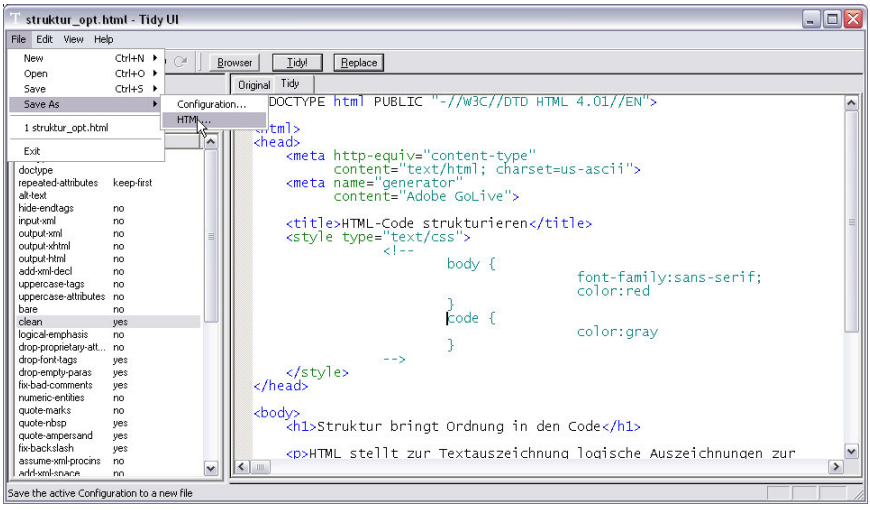


Abbildung 1.20: Die korrigierte Datei können Sie dann speichern.

CSS-Code validieren

Bei CSS sieht die Sache schon etwas anders aus. Programme zum Offline-Testen von CSS-Code gibt es nicht. Einzig den CSS-Validator des W3C können Sie online nutzen, um CSS-Code zu validieren.

Sie finden den CSS-Validator auf `http://jigsaw.w3.org/css-validator`.

CSS-Editoren

Fehler im CSS-Code vermeiden Sie am einfachsten dadurch, dass Sie zum Erstellen des CSS-Codes geeignete Editoren verwenden. Sowohl die großen Website-Editoren Adobe® GoLive™ CS als auch Macromedia Dreamweaver verfügen über einen integrierten CSS-Editor, der sehr guten Code erzeugt. Allerdings haben Sie bei beiden nicht immer die volle Kontrolle über den Code. Es kommt gelegentlich vor, dass die Editoren bei Änderungen an den CSS-Eigenschaften auch die Reihenfolge der Definition umstellen oder beispielsweise Kurzschreibweisen verwenden, die bezüglich der Browserkompatibilität nicht optimal, wenn auch standardkonform sind.

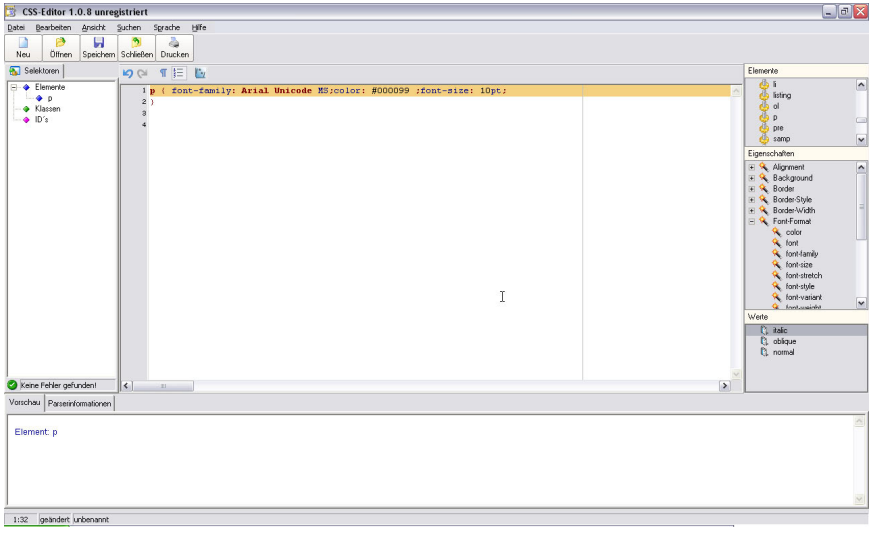
Darüber hinaus können Sie aber auch separate CSS-Editoren verwenden. Davon gibt es mittlerweile eine ganze Reihe. Vier sollen hier nachfolgend stellvertretend vorgestellt werden. Teilweise finden Sie die Programme auch auf der beiliegenden Buch-CD.

CSS-Editor 1.0

Den Editor bekommen Sie auf der Webseite des Herstellers unter dem URL `http://www.css-maker.de/css-maker/index.html` *oder alternativ unter* `http://www.shareware.de`.

Der CSS-Editor ist ein einfacher, dafür aber sehr übersichtlicher Editor. Auf der rechten Seite finden Sie Listen mit HTML-Elementen, CSS-Eigenschaften und ihren möglichen Werten. Um beispielsweise einen Elementstil zu erstellen, wählen Sie erst das Element per Doppelklick aus, dann die CSS-Eigenschaft ebenfalls per Doppelklick. Zum Schluss wählen Sie den verfügbaren Wert aus.

Abbildung 1.21: Der Editor CSS-Editor 1.08



Der Stil wird dann erstellt bzw. ergänzt und der ausgewählte Wert in den Stil geschrieben. Unterhalb des Editorfensters, der das Stylesheet in der Quellcodeansicht anzeigt, ist ein Vorschaufenster, das die gewählten Formatierungen anzeigt.

Etwas ungünstig ist hier die Realisation der Eingabe für numerische Werte. Beispielsweise können Sie zwar die Schriftgröße (`font-size`-Eigenschaft) in einen Stil einfügen, aber Sie können nur die vordefinierten Werte wie `small`, `medium` und `large` auswählen, nicht jedoch numerische Werte und Einheiten eingeben. Sie müssten also zunächst einen vorgegebenen Wert auswählen, können den aber danach in der Codeansicht des Stylesheets in den gewünschten Wert ändern.

Bei anderen Eigenschaften mit numerischen Werten werden einzelne Werte zur Auswahl angeboten, nicht jedoch bei allen Eigenschaften. So stehen beispielsweise für die Eigenschaft `margin`, die den Außenabstand eines Elements bestimmt, nur Werte mit den Einheiten `mm` und `px` zur Verfügung.

Wenn Sie einen vorhandenen Stil ändern, müssen Sie darauf achten, den Cursor an die richtige Stelle zu setzen, da die neu hinzugefügte Eigenschaft an der Cursor-Position eingefügt wird. Automatische Korrekturen finden nicht statt. Möchten Sie einer Eigenschaft einen neuen Wert zuweisen, müssen Sie die Eigenschaft und deren Wert zuvor markieren. Ansonsten wird die Eigenschaft ein zweites Mal eingefügt.

StyleAssistant

Bei StyleAssistant handelt es sich um ein kleines Tool von Thomas Meinike, das Sie auf `http://www.styleassistant.de` *herunterladen können.*

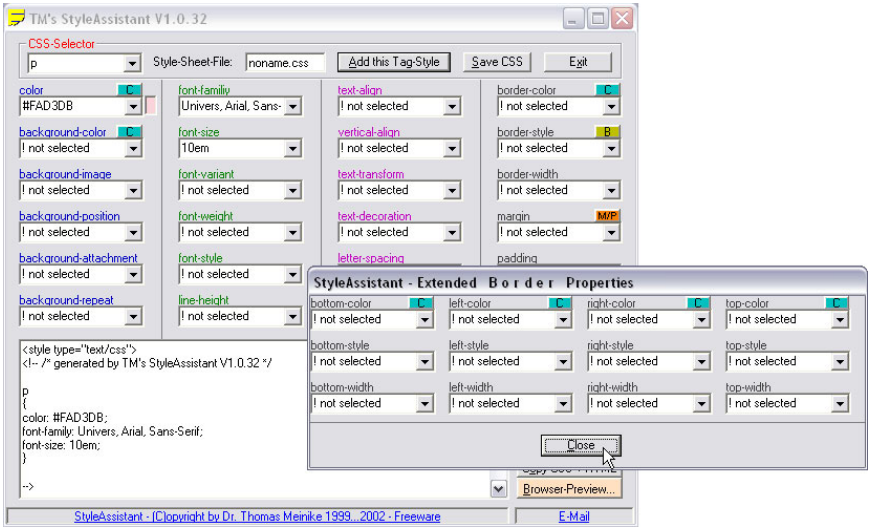
Es ermöglicht die einfache Erzeugung von CSS-Stilen aller Art, die Sie entweder in eine Datei schreiben oder in die Zwischenablage kopieren und von dort beispielsweise in Ihrem HTML-Editor einfügen können.

Die Auswahl der Eigenschaften und Werte erfolgt komfortabel über Kombinationslistenfelder, in die Sie jedoch auch Werte manuell eingeben können. Wenn Sie einen Stil für ein Element erzeugen möchten, gehen Sie dazu folgendermaßen vor:

1. Wählen Sie das Element aus der Liste CSS-SELECTOR aus.
2. Geben Sie den Namen der CSS-Datei in das Feld STYLE-SHEET-FILE ein, wenn Sie die Stile in einer CSS-Datei speichern möchten.
3. Wählen Sie nun die CSS-Eigenschaften und deren Werte über die Listenfelder aus.
4. Klicken Sie auf ADD THIS TAG-STYLE, um den Stil in das Textfeld am Ende des Dialogfelds zu schreiben.
5. Wenn Sie möchten, können Sie den Inhalt des Textfeldes nun in eine CSS-Datei schreiben. Klicken Sie dazu auf SAVE CSS. Alternativ können Sie auch den oder die erzeugten Stile im Textfeld markieren und mit der Schaltfläche COPY TO CLIPBOARD in die Zwischenablage kopieren.

Eine Browservorschau der Stile ist ebenfalls möglich, indem Sie auf BROWSER-PREVIEW klicken.

Abbildung 1.22: Erstellen von CSS-Stilen mit StyleAssistant



StyleAssistant hält sich beim Erzeugen des CSS-Codes sehr eng an den CSS-Standard und ermöglicht eine einfache und effiziente Auswahl und Eingabe der Werte. Nachteilig ist allerdings, dass einmal erstellte Stile nicht mehr ausgewählt und geändert werden können. Sie müssen sie dann mit allen Einstellungen neu erstellen. Zudem wird eine vorhandene CSS-Datei (nach Rückfrage) überschrieben, wenn Sie erneut den Inhalt des Textfeldes als CSS-Datei speichern, die Datei wird nicht ergänzt oder geändert. Ansonsten ist der StyleAssistant aber ein sehr einfach zu bedienendes Tool.

TopStyle Lite

Zu den leistungsfähigsten Tools gehört sicherlich TopStyle Lite. Anders als die zuvor vorgestellten Programme ist es keine als Freeware entwickelte Software, sondern eine Lite-Version von TopStyle Pro, die von einem ganzen Entwicklerteam erstellt wurde.

TopStyle Lite und TopStyle Pro bekommen Sie auf der Webseite des Herstellers. Der URL lautet: `http://www.bradsoft.com`.

Die Benutzeroberfläche kann man als aufgeräumt bis spartanisch beschreiben. Das ist aber nur der erste Eindruck. Dahinter verbirgt sich ein Funktionsumfang, der kaum Wünsche übrig lässt. Sie können sich beispielsweise die verfügbaren CSS-Eigenschaften nach CSS-Standard oder Browserkompatibilität anzeigen lassen.

Folgende Schritte sind erforderlich, wenn Sie eine CSS-Datei erstellen und darin einen Elementstil erzeugen möchten:

1. Wählen Sie FILE / NEW aus, um eine neue Datei zu erzeugen.
2. Klicken Sie den Menüeintrag EDIT / NEW SELECTOR an oder benutzen Sie das entsprechende Icon in der Symbolleiste.

3. Geben Sie nun den Namen des Selektors in das Feld CURRENT SELECTOR ein oder wählen Sie ein HTML-Element aus der Liste HTML-ELEMENT aus. Klicken Sie anschließend auf die Pfeilschaltfläche, um den Selektor hinzuzufügen.

4. Schließen Sie das Dialogfeld mit OK.
5. Setzen Sie nun die Eigenschaften für den Stil, indem Sie deren Werte im Style-Inspektor eingeben oder aus den Listenfeldern auswählen.
6. Wählen Sie FILE / SAVE aus, um die Datei zu speichern.

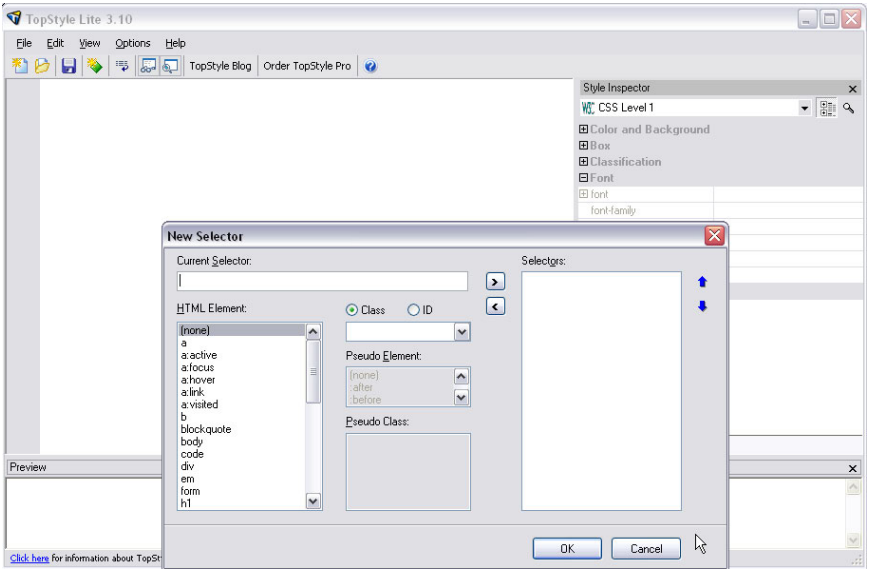


Abbildung 1.23: TopStyle Lite

Falls Sie später einen einmal erstellten Stil wieder ändern möchten, müssen Sie nur den Cursor in den Stil setzen. Im Style-Inspektor auf der rechten Seite werden dann die Eigenschaften des Stils angezeigt und Sie können sie bearbeiten. Auch die Auswahl von Farben und numerischen Werten erfolgt hier wesentlich effizienter als bei den zuvor vorgestellten Tools.

CSSED

CSSED steht als einziges Tool für viele verschiedene Betriebssysteme zur Verfügung, neben Windows auch für Mac OS X und Linux. Sie finden es auf der Seite `http://www.zwahlendesign.ch/en/node/40`.

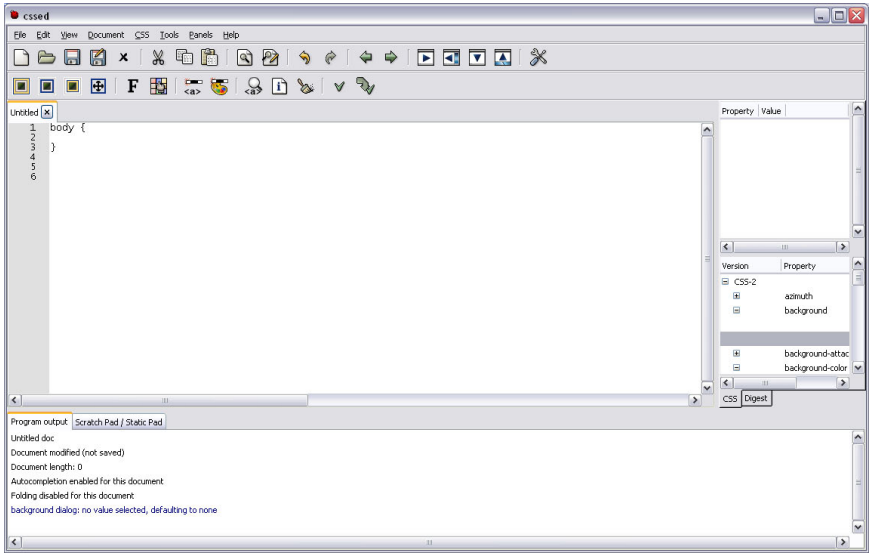
Die Installation gestaltet sich hier etwas schwierig. Sie finden auf der Seite des Herstellers jedoch entsprechende Hinweise. Unter Windows müssen Sie zunächst die GTK-Umgebung installieren, die Sie ebenfalls von der Herstellerseite herunterladen können.

Wichtig ist allerdings, dass Sie nach der Installation Windows neu starten, sonst wird eine Datei nicht gefunden, die den Start des Programms verhindert.

Die Benutzeroberfläche ist im Prinzip gut durchdacht. Auf der rechten Seite finden Sie eine Liste der CSS-Eigenschaften, über die Sie Einstellungen für die Stile vornehmen können.

Mit dem Eintrag CSS / SELECTOR-WIZARD können Sie einen Selektor erstellen und dessen CSS-Eigenschaften dann über das CSS-Eigenschaftenfenster festlegen.

Abbildung 1.24: CSSED-Programmfenster unter Windows



Eigentlich macht das Programm einen guten Eindruck. Allerdings trat bei mir ein Problem mit der Anzeige der CSS-Eigenschaften auf. Die untergeordneten Listeneinträge waren nicht sichtbar (siehe Abbildung 1.24). Das führt dazu, dass man nicht weiß, um welche Eigenschaft es sich handelt, und dass auch die einmal eingegebenen Werte nicht angezeigt werden. Es muss jedoch nicht sein, dass das Problem auch in der Linux- und Mac OS-Version vorhanden ist.

2 Stile definieren

Damit Sie den Referenzteil auch effektiv nutzen können, ist es wichtig, dass Sie grundsätzlich wissen, wie Stile definiert werden. Darum und um weitere wichtige Konzepte von CSS, wie Vererbung, das Boxmodell und die verschiedenen Stilarten, geht es in diesem Kapitel.

2.1 Verschiedene Stilarten und deren Bedeutung

Selektoren bestimmen, für welche Elemente der HTML-Seite die CSS-Formatierungen gelten sollen. Sie legen also fest, was formatiert werden soll. Üblicherweise werden einfache Selektoren verwendet. Dazu gehören Element-Selektoren, Klassen-Selektoren und ID-Selektoren.

Element-Selektoren werden in der CSS-Dokumentation als Typ-Selektoren bezeichnet. Nachfolgend wird jedoch die Bezeichnung Element-Selektor beibehalten, weil diese auch von vielen Editoren benutzt wird und deshalb eine weitere Verbreitung sowohl in Literatur als auch Software gefunden hat.

Operatoren

Operatoren dienen dazu, aus den einfachen Selektoren komplexere zusammenzusetzen. Sie können damit ganz speziell bestimmte Kombinationen und Hierarchien von HTML-Elementen ansprechen.

Pseudo-Elemente

Bei Pseudo-Klassen und Pseudo-Elementen handelt es sich im Prinzip um Elementstile, nur werden diese nicht für existierende HTML-Elemente erstellt, sondern für Elemente, die der Browser erst bei Anzeige der Seite selbst generiert. Ein gutes Beispiel dafür, sind die Pseudo-Klassen `:link` und `:visited`. Erst der Browser kann hier feststellen, ob ein Link schon mal besucht wurde (`:visited`) oder nicht (`:link`). Ein HTML-Element für besuchte Links gibt es nicht. Erst die Kombination aus Selektoren, Operatoren und Pseudo-Elementen/-Klassen ermöglicht eine perfekte und gleichzeitig einfache Gestaltung mittels CSS. Zunächst sollen daher die wichtigsten Selektoren vorgestellt werden.

Eine Stildefinition hat immer den gleichen Aufbau:

```
Selektor { eigenschaft:wert; ... }
```

Dem Namen des Selektors folgt ein geschweiftes Klammerpaar. Innerhalb der Klammern definieren Sie die CSS-Eigenschaften, die die Formatierungen des Stils bestimmen.

INFO

Alles das, was vor der ersten geschweiften Klammer steht, wird gemäß CSS-Dokumentation als Selektor bezeichnet. Der Inhalt einschließlich der umgebenden geschweiften Klammern ist der Deklarationsblock. Selektoren dürfen immer nur zusammen mit einem Deklarationsblock auftreten. Falsch wäre also die Deklaration:

```
body
```

Richtig hingegen:

```
body {}
```

Selektoren und Deklarationsblock ergeben zusammen eine CSS-Regel oder einen CSS-Stil.

Element-Selektoren

Element-Selektoren, auch Elementstile genannt, sind am einfachsten zu verwenden. Sie legen die Formatierung für ein bestimmtes (X)HTML-Element fest. Ihr Name entspricht immer dem Namen des (X)HTML-Elements. Wenn Sie also die Formatierung für die mit dem `p`-Element erzeugten Absätze festlegen möchten, heißt der Stil `p` und Sie definieren ihn folglich mit `p {}`. Genauso würden Sie einen Stil für das `b`-Element mit `b {}` definieren.

INFO

Element-Selektoren brauchen Sie auch nicht explizit einem Element der Seite zuweisen, wie dies bei Klassen- und ID-Selektoren erforderlich ist. Haben Sie einen Selektor `p` *erstellt, wird der automatisch auf alle* `p`*-Elemente der Seite angewendet. Genau deshalb sind Element-Selektoren ausgesprochen nützlich, wenn Sie schnell eine Seite umgestalten möchten.*

:-) TIPP

Jedes Stylesheet sollte mindestens die folgenden Elementstile enthalten, um Überschriften (`h1 - h6`*), Absätze (*`p`*) und die Seite selbst (*`body`*) zu formatieren.*

Listing 2.1: Wichtige Elementstile für Stylesheets

```
body {}
p {}
h1 {}
h2 {}
h3 {}
h4 {}
h5 {}
h6 {}
```

Angaben, wie z.B. gleiche Schriftfamilien, müssen Sie dann aber nicht für jeden Stil erstellen. Durch die Vererbung, die in CSS vorgesehen ist, werden bestimmte Eigenschaften, die Sie beispielsweise für das `body`*-Element definieren, an untergeordnete Elemente wie Überschriften und Absätze vererbt.*

Mehr zum Thema Vererbung finden Sie etwas weiter unten im Abschnitt »Vererbung von Stilen«.

Klassen-Selektoren

Klassen-Selektoren werden auch CSS-Klassen oder Klassen-Stile genannt. Ein Klassen-Selektor kann jedem beliebigen Element der Webseite zugewiesen werden, für das ein `class`-Attribut verfügbar ist. Über dieses `class`-Attribut wird der Klassen-Selektor nämlich dem HTML-Element zugewiesen.

Haben Sie beispielsweise eine CSS-Klasse mit dem Namen `fehler` definiert, wird diese wie folgt einem Absatz zugewiesen:

```
<p class="fehler">Hier kann eine Fehlermeldung stehen </p>
```

Listing 2.2: Zuweisen eines Klassen-Selektors zu einem Absatz

Namensgebung für CSS-Klassen

Namen von CSS-Klassen sind prinzipiell frei wählbar, da sie unabhängig von den Elementnamen sind. Für die Bildung der Namen gelten die allgemeinen Regeln für CSS-Bezeichner.

In CSS 2.0 dürfen Bezeichner alle Zeichen von A bis Z (bzw. a bis z) sowie die Ziffern 0 bis 9, den Bindestrich »-« und den Unterstrich »_« enthalten. Darüber hinaus sind bestimmte Sonderzeichen zulässig. Zu diesen Sonderzeichen gehören:

- *die Zeichen 161 und höher des ISO 10646-Zeichensatzes,*
- *geschützte Zeichen, die jedoch escaped werden müssen (d.h. Sie müssen diesen Zeichen das Escape-Zeichen »\« voranstellen),*
- *alle Zeichen des ISO 10646-Zeichensatzes, wenn Sie sie als numerischen Code angeben. Auch der numerische Code muss mit einem Escape-Zeichen beginnen.*

Bezeichner dürfen nicht mit einem Trennstrich oder einer Ziffer beginnen. Daraus lässt sich im Umkehrschluss ableiten, dass Bezeichner durchaus mit einem Unterstrich beginnen dürfen.

Dennoch sollten Sie vermeiden, Stile zu definieren, die mit einem Unterstrich beginnen. Sowohl der Internet Explorer 6 als auch der Netscape Navigator 4.7- und der Palm Web Browser stellen Stile mit solchen Namen nicht dar, obwohl sie durchaus standardkonform sind. Genauso sollten Sie

auf Sonderzeichen verzichten, auch wenn Sie diese escapen oder mit ihrem numerischen Code darstellen. Damit haben verschiedene Browser ebenfalls Schwierigkeiten.

Mehr dazu finden Sie in Kapitel 3, »Browseroptimierung«. Einige dieser durchaus zulässigen Sonderzeichen werden nämlich verwendet, um bestimmte Stile vor Browsern zu verbergen oder nur für bestimmte Browser zu definieren.

Die Beispiel-CSS-Datei mit den folgenden CSS-Codes finden Sie auf der Buch-CD im Verzeichnis BSP/K02 *unter dem Namen* vorlage.css. *Die CSS-Datei können Sie mit folgendem Code im* `head`*-Bereich einer (X)HTML-Datei mit der Webseite verknüpfen:*

```
<link href="vorlage.css" rel="stylesheet">
```

Details zur Verknüpfung von Stylesheet-Dateien mit Webseiten und andere Möglichkeiten, CSS-Code innerhalb einer Webseite verfügbar zu machen, finden Sie in Kapitel 3, »Browseroptimierung«.

Wenn Sie einen Klassen-Selektor definieren möchten, stellen Sie dem Namen der CSS-Klasse einen Punkt voran. Dieses Zeichen kennzeichnet den Stil als Klassenstil. Die Definition für die CSS-Klasse `fehler` sieht also wie folgt aus:

```
.fehler { color:red }
```

ID-Selektoren

Bei den ID-Selektoren handelt es sich um Selektoren, die über das `id`-Attribut eines (X)HTML-Elements zugewiesen werden. Da das `id`-Attribut innerhalb der Seite eindeutig zu sein hat, bedeutet das, dass ein ID-Selektor (korrektes (X)HTML vorausgesetzt) nur auf genau ein Element der Seite angewendet wird.

Auch bei den ID-Selektoren können Sie den Namen frei wählen, sollten jedoch die Bestimmungen zu den Werten für das `id`-Attribut ihrer HTML- bzw. XHTML-Versionen berücksichtigen. Für die Namensgebung gilt (bezogen auf CSS) jedoch das Gleiche wie für die Klassen-Selektoren.

Im Unterschied dazu stellen Sie dem Namen bei ID-Selektoren jedoch nicht den Punkt voran, sondern ein Doppelkreuz »#«. Eine Definition für einen ID-Selektor mit dem Namen `banner` sieht also wie folgt aus:

```
#banner { font-size:3em }
```

Der Palm Web Browser 2.0.1 unterstützt ID-Selektoren. In dem für den Test verwendeten Beispiel werden jedoch Formatierungen benutzt, die der Browser nicht unterstützt, wie die rechtsbündige Textausrichtung. Daher zeigt der Browser die Seite nicht korrekt an.

Die Eigenschaft `font-size` *legt die Schriftgröße fest. Bei dem Wert* `3em` *handelt es sich um eine relative Größe in der Einheit* `em`*. Der Wert* `3em` *besagt, dass die Schrift in der dreifachen Größe angezeigt wird. Was die Ausgangsgröße ist, die eben verdreifacht wird, hängt von den übrigen Stilen im Stylesheet und von der Verschachtelung des HTML-Codes ab, dem die Formatierung zugewiesen wird. Deutlicher wird das, wenn Sie die Abschnitte »Vererbung von Stilen« und »Zahlen und Maßeinheiten« weiter unten gelesen haben.*

Die Zuweisung von ID-Stilen erfolgt wie gesagt über das `id`-Attribut. Wenn Sie also den Stil einem Absatz zuweisen möchten, legen Sie als Wert für dessen `id`-Attribut den Namen des ID-Stils fest:

```
<p id="banner">Seitenbanner</p>
```

Pseudo-Klassen und Pseudo-Elemente

Neben diesen Selektorarten gibt es Pseudo-Klassen und Pseudo-Elemente, für die Sie Stile erstellen können. Pseudo-Klassen und Pseudo-Elemente gestatten die Formatierung abhängig von Informationen, die nicht im Quellcode zu finden sind, sondern die nur der darstellende Browser ermitteln kann.

Zu den Pseudo-Elementen gehört beispielsweise die Angabe `:first-line`, die auf die erste Zeile eines Absatzes verweist. Unabhängig davon, dass es kein (X)HTML-Element gibt, das diese Zeile definiert, kann auch erst der Browser, abhängig von der Schriftgröße, der Fenstergröße und anderen Parametern, bestimmen, wie viele Zeichen zur ersten Zeile des Absatzes gehören. Mit Hilfe solcher Pseudo-Elemente können Sie auch Informationen hinzufügen, beispielsweise abhängig von der Sprache eines Absatzes.

Im Gegensatz zu Pseudo-Elementen sind Pseudo-Klassen meist dynamisch in dem Sinne, dass (X)HTML-Elemente ihre zugeordnete Pseudo-Klasse wechseln können (beispielsweise Hyperlinks). Während ein normaler, noch nicht besuchter Hyperlink der Pseudo-Klasse `link` zugeordnet ist, erhält er nach einmaligem Anklicken die Pseudo-Klasse `visited`.

Wenn Sie beispielsweise die Formatierung für besuchte Hyperlinks festlegen möchten, nutzen Sie dazu die Pseudo-Klasse `visited`, die Sie auf den Element-Selektor für das `a`-Element anwenden. Da alle Pseudo-Elemente und Pseudo-Klassen vom Rest des Selektors durch einen Doppelpunkt getrennt werden, lautet die Definition also:

```
a:visited { text-decoration:none; color:gray }
```

In diesem Fall wird für besuchte Links die automatische Unterstreichung deaktiviert, indem die Eigenschaft `text-decoration` auf `none` gesetzt wird. Außerdem wird die Schriftfarbe auf grau gesetzt.

Mehr Details zu Pseudo-Elementen und Pseudo-Klassen finden Sie im Referenzteil in Kapitel 11, »Sonstige Formatierungen, Pseudo-Elemente und -Klassen«.

Abbildung 2.1: Die Darstellung des besuchten Links in Firefox

Operatoren nutzen

Es gibt eine Reihe von Zeichen in CSS, die eine besondere Bedeutung haben. Diese Zeichen dienen dazu, verschiedene einfache Selektoren zu verknüpfen, so dass komplexere Selektoren entstehen. Diese besonderen Zeichen werden nachfolgend als Operatoren bezeichnet.

Selektoren gruppieren

Häufig kommt es vor, dass Sie für mehrere Stile die gleichen Formatierungen festlegen möchten. Dazu müssen Sie nicht alle Stile einzeln angeben, sondern brauchen nur die Selektoren zu gruppieren. CSS stellt dazu das Komma als Trennzeichen zur Verfügung.

CSS-Selektor:	Gruppierung (»,«)	*Beschreibung:*
CSS-Version:	1.0, 2.0, 2.1, 3.0, Mobile, TV	Gruppiert mehrere einfache Selektoren.

Palm	NN	Mozilla		Internet Explorer							Opera			Safari		iCab	Kq	FF
2.x	4.x	1.x	1.8	4.x	5.0x (Win)	5.0 (Mac)	5.1+ (Mac)	5.5 (Win)	6	7	6	7	8	1.x	2.0	2.9+	3.x	1.0
●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●

Möchten Sie beispielsweise sowohl für eine CSS-Klasse, die auf Absätze mit Fehlermeldungen angewendet werden soll (`.fehler`), als auch für wichtige Hervorhebungen (`.wichtig`) eine rote Schrift bestimmen, sähe eine einfache Definition dazu wie folgt aus:

```
.fehler  { color:red }
.wichtig { color:red }
```

Listing 2.3: Zwei CSS-Klassen mit gleicher Formatierung

Viel kürzer können Sie es haben, wenn Sie die CSS-Stile gruppieren. Dazu geben Sie die Stile nacheinander als Selektor an und trennen sie durch ein Komma.

```
.fehler, .wichtig { color:red }
```

Nachkommen-Selektor

Der Nachkommen-Selektor bezeichnet Elemente der (X)HTML-Seite, die in einer bestimmten Hierarchie ineinander verschachtelt sind. Als Operator für den Nachkommen-Selektor dient das Leerzeichen.

CSS-Selektor:	Nachkommen-Selektor (» «)	*Beschreibung:*
CSS-Version:	1.0, 2.0, 2.1, 3.0, Mobile, TV	Wählt ein Element `E` aus, das sich innerhalb eines anderen Elements `S` befindet: `S E`.

Palm	NN	Mozilla		Internet Explorer							Opera			Safari		iCab	Kq	FF
2.x	4.x	1.x	1.8	4.x	5.0x (Win)	5.0 (Mac)	5.1+ (Mac)	5.5 (Win)	6	7	6	7	8	1.x	2.0	2.9+	3.x	1.0
●	○	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●

Möchten Sie also nur span-Elemente formatieren, die sich innerhalb eines p-Elements befinden, könnte der entsprechende Selektor wie folgt aussehen:

```
p span { color:lime }
```

Bei dieser Definition wird die Formatierung auf alle span-Elemente angewendet, die sich innerhalb eines p-Elements befinden. Dabei spielt es keine Rolle, ob sie dem p-Element direkt untergeordnet sind oder ob vielleicht innerhalb des p-Elements noch ein div-Element vorhanden ist, in dem sich das span-Element befindet.

Bei dem folgenden HTML-Code würden nur die ersten beiden (hier fett dargestellten) span-Elemente mit dem Stil formatiert. Das dritte würde nicht formatiert werden, weil es sich innerhalb eines div-Elements befindet.

Abbildung 2.2: Die hellgrünen Elemente wurden durch den Stil mit dem Nachkommen-Selektor formatiert.

Listing 2.4: Der HTML-Code zum Testen des Nachkommen-Selektors

```
<body>
      <p id="banner">Schrift groß? Dann werden
      ID-Stile angewendet!</p>
      <p class="_fehler">Wenn dieser Text nicht
      rot angezeigt wird, wendet der Browser keine
      Stile an, deren Name mit einem Unterstrich
      beginnt.</p>
      <p class="fehler">Hier könnte die
      Fehlermeldung stehen!</p>
      <p>Der<code><span> innerhalb des span-
      Elements </span> enthaltene Text wird grün
      dargestellt, wenn der Browser den
      Nachkommen-Selektor unterstützt.</code></p>
```

```
    <p>Der <a href="../index.html">Link</a> in
    diesem Absatz wird grau dargestellt, wenn er
    bereits einmal angeklickt wurde und der
    Browser die Pseudo-Klasse <span>:visited
    </span> beherrscht.</p>
    <div><span>Dieser Text wird nicht durch den
    Nachkommen-Selektor formatiert.</span></div>
</body>
```

Selbstverständlich funktioniert der Selektor nicht nur mit Elementstilen, sondern auch mit Klassen- und ID-Selektoren. Möchten Sie beispielsweise nur die `span`*-Elemente formatieren, die sich innerhalb des Elements mit der ID* `banner` *befinden, würde der Selektor entsprechend* `#banner span` *lauten.*

Kind-Selektor

Im Gegensatz zum Nachkommen-Selektor wählt der Kind-Selektor nur die Elemente aus, die direkt dem angegebenen Element untergeordnet sind. Als Operator für den Kind-Selektor wird das Größer-Zeichen `>` verwendet.

CSS-Selektor:	Kind-Selektor (»>«)	*Beschreibung:*
CSS-Version:	2.0, 2.1, 3.0, Mobile, TV	Wählt ein Element `E` aus, das Kind-Element zu einem anderen `S` ist: `S > E`.

Palm	NN	Mozilla		Internet Explorer							Opera			Safari		iCab	Kq	FF
2.x	4.x	1.x	1.8	4.x	5.0x (Win)	5.0 (Mac)	5.1+ (Mac)	5.5 (Win)	6	7	6	7	8	1.x	2.0	2.9+	3.x	1.0
●		●	●			●	●				●	●	●	●	●	●[1]	●	●

[1] ab iCab 3.0

Bei dem Kind-Selektor `p > span {text-decoration:underline}` wird die Formatierung nur auf die `span`-Elemente angewendet, die direkt innerhalb eines `p`-Elements stehen. Im Beispiel-Code aus Listing 2.4 wird daher nur das zweite `span`-Element formatiert und unterstrichen dargestellt.

Direkter Geschwister-Selektor bzw. Nachbar-Selektor

Der direkte Geschwister- oder Nachbar-Selektor wählt Elemente aus, die auf gleicher Ebene direkt davor ein definiertes Element haben. Beispielsweise sind im folgenden Code:

```
<body>
   <p id="banner">Werbezug</p>
   <p>Absatz, der mit Linie abgetrennt werden soll.</p>
```

```
    <p>Absatz, der nicht mit Linie abgetrennt werden soll.</p>
</body>
```

der Absatz mit der ID `banner` und das danach folgende `p`-Element gleichrangig dem `body`-Element untergeordnet. Mit Hilfe des Nachbar-Selektors können Sie daher dafür sorgen, dass das `p`-Element, das dem Absatz mit der ID `banner` folgt, mit einem oberen Abstand von 10 Punkten formatiert wird. Zusätzlich wird der Absatz mit einer durchgezogenen roten Linie am oberen Rand versehen. Dazu definieren Sie den Stil:

Listing 2.5: Beispiel für den direkten Geschwister-Selektor

```
#banner + p {
    padding-top:10pt;
    border-top:1px solid red
}
```

CSS-Selektor:	Direkter Geschwister-Selektor (»+«)	*Beschreibung:* Wählt ein Element `E` aus, das gleichrangig dem Element `S` folgt: `S + E`.
CSS-Version:	2.0, 2.1, 3.0	

Palm	NN	Mozilla		Internet Explorer							Opera			Safari		iCab	Kq	FF
2.x	4.x	1.x	1.8	4.x	5.0x (Win)	5.0 (Mac)	5.1+ (Mac)	5.5 (Win)	6	7	6	7	8	1.x	2.0	2.9+	3.x	1.0
●		●	●			●	●				●	●	●	●	●	●[1]	●	●

[1] ab iCab 3.0

Indirekter Geschwister-Selektor

CSS 3.0 bietet noch einen weiteren Geschwister-Selektor. Er wählt Elemente aus, die auf ein – nicht notwendigerweise direkt – vorangehendes Geschwisterelement folgen. Bei folgendem HTML-Code bedeutet dies, dass ein Selektor

```
h2 ~ div { background-color:gray }
```

dazu führt, dass das `div`-Element unter dem `h2`-Element mit einem grauen Hintergrund versehen wird.

Listing 2.6: Aufbau des HTML-Fragments

```
<h2>Zwischenüberschrift der Ebene 2</h2>
<h3>Zwischenüberschrift der Ebene 3</h3>
<div>Dieser Text soll einen grauen Hintergrund haben.</div>
```

Allerdings ist all das nur graue Theorie. In der Praxis können Sie diesen Selektor noch nicht einsetzen, da es derzeit keinen Browser gibt, der ihn unterstützt.

Abbildung 2.3: Der Opera-Browser stellt alle Stile korrekt dar. Sichtbar ist somit auch die Linie, die vom Geschwister-Selektor erzeugt wird.

Universal-Selektor

CSS kennt einen universellen Platzhalter, Universal-Selektor genannt. Das ist das Sternchen »*«. Es ersetzt ein beliebiges Element. Der Selektor `p * a` würde also sowohl auf die Elemente `<p><span><a href=""></span></p>` wie auch auf `<p><li><a href=""></li></p>` angewendet.

CSS-Selektor:	Universal-Selektor (»*«)	*Beschreibung:*
CSS-Version:	2.0, 2.1, 3.0, Mobile, TV	Wählt ein Element `E` aus, das irgendwo innerhalb des übergeordneten Elements `S` enthalten ist. Dabei muss aber zwischen `S` und `E` ein Element vorhanden sein: `S * E`.

Palm	NN	Mozilla		Internet Explorer							Opera			Safari		iCab	Kq	FF
2.x	4.x	1.x	1.8	4.x	5.0x (Win)	5.0 (Mac)	5.1+ (Mac)	5.5 (Win)	6	7	6	7	8	1.x	2.0	2.9+	3.x	1.0
●	○	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●[1]	●	●

[1] ab iCab 3.0 wird der Universal-Selektor vollständig unterstützt

Da der Universal-Selektor aber ein beliebiges Element darstellt, muss an der Stelle, an der es innerhalb des Selektors auftaucht, auch ein Element vorhanden sein. Der Stil `p * a` würde daher nicht auf Links angewendet werden, die unmittelbar innerhalb eines `p`-Elements definiert sind.

:-) TIPP

Steht der Universal-Selektor am Anfang des Selektors können Sie ihn weglassen. Die Stile `*p` *und* `p` *sind also gleichbedeutend. Das ist auch der Grund, warum die Angabe* `.achtung` *den Klassenstil auf jedes beliebige Element anwendet, das mit* `class="achtung"` *definiert ist. Dieser Selektor ist nämlich gleichbedeutend mit* `* .achtung`.

Enthält Ihr Code beispielsweise die folgende Definition und möchten Sie nur die Formatierung für `i`-Elemente innerhalb eines `div`-Elements mit der Klasse `achtung` festlegen, bei denen zwischen dem `i`-Element und dem `div`-Element mindestens noch ein Element (hier `span`) vorhanden ist, funktioniert das nur mit dem Universal-Selektor.

Listing 2.7: Beispiel zum Universal-Selektor

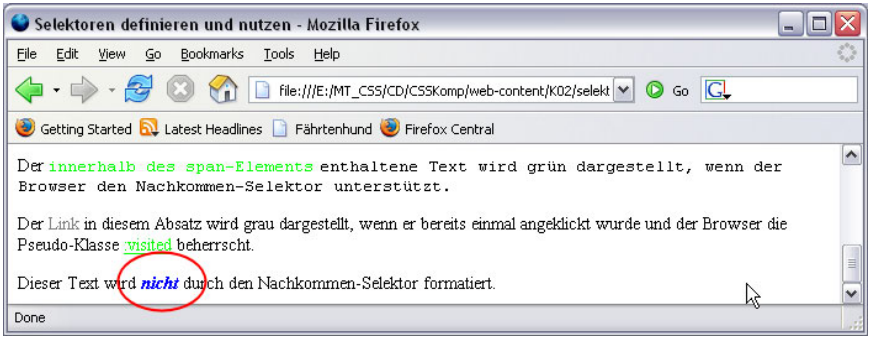
```
<div class="achtung">
    <span>Dieser Text wird <i>nicht</i> durch
    den Nachkommen-Selektor formatiert.</span>
</div>
```

Mit folgendem Code

```
.achtung * i { font-weight:bold; color:blue }
```

wird dem Inhalt des `i`-Elements zusätzlich zu der durch das Element bestimmten, kursiven Auszeichnung auch noch die Farbe Blau (`color:blue`) und die Textauszeichnung Fett (`font-weight:bold`) zugewiesen.

Abbildung 2.4: Firefox zeigt die Formatierung korrekt an.



Der Netscape Navigator 4.x und iCab (bis 2.9.x) zeigen die Formatierung nicht an, obwohl beide den Universal-Selektor zumindest teilweise beherrschen. Teilweise bedeutet, dass beispielsweise der Universal-Selektor am Anfang des Selektors akzeptiert wird, wenn das Zeichen * weggelassen wird.

Attribut-Selektoren

Mit CSS 2.0 wurden auch Attribut-Selektoren eingeführt. Sie ermöglichen Ihnen, Elemente der (X)HTML-Seite über deren Attribute oder Attributwerte auszuwählen.

Sie finden den Code für die folgenden Beispiele in der Datei BSP/K02/attributselektoren.html *sowie in der CSS-Datei* BSP/K02/vorlage.css.

Leider ist der Einsatz der Attribut-Selektoren noch nicht praxistauglich, wenn Sie auf Kompatibilität mit dem Internet Explorer 6 und 7 Wert legen, da diese die Attribut-Selektoren nicht unterstützen. Alle anderen großen Browser – von Netscape 7 über Mozilla/Firefox bis hin zum Opera-Browser und Konqueror – können mit Attribut-Selektoren umgehen.

Abfragen, ob ein Attribut vorhanden ist

Möchten Sie einen Stil für ein Element erstellen, das ein bestimmtes Attribut hat, geben Sie einfach den Attributnamen in eckigen Klammern hinter dem Selektor an.

CSS-Selektor:	Attribut-Selektor (»`[attribut]`«)	*Beschreibung:*
CSS-Version:	2.0, 2.1, 3.0	Wählt ein Element `E` aus, das das angegebene Attribut `A` hat: `E[A]`.

Palm	NN	Mozilla		Internet Explorer							Opera			Safari		iCab	Kq	FF
2.x	4.x	1.x	1.8	4.x	5.0x (Win)	5.0 (Mac)	5.1+ (Mac)	5.5 (Win)	6	7	6	7	8	1.x	2.0	2.9+	3.x	1.0
		●	●								●	●	●	●	●	●[1]	●	●

[1] ab Version 3.0

Listing 2.8: Der HTML-Code zum Testen der Attribut-Selektoren

```
<body>
    <h1 lang="de">Herzlich Willkommen!</h1>
    <h1 lang="en">Welcome!</h1>
    <p lang="en">You don't speak german? <a
    href="english.html">Please visit our english
    site!</a>
    </p>
</body>
```

Bei vorstehendem HTML-Code erreichen Sie mit dem folgenden Stil, dass alle Elemente der Seite, die über das `lang`-Attribut verfügen, rechtsbündig ausgerichtet (`text-align:right`) werden.

```
*[lang] { text-align:right }
```

Safari stellt diesen Attribut-Selektor in Kombination mit dem Universal-Selektor nicht dar, wenn zwischen dem Universal-Selektor und dem Attribut-Selektor ein Leerzeichen steht. Alle anderen Browser, die Attribut-Selektoren überhaupt unterstützen, stören sich nicht daran. Da jedoch der Universal-Selektor am Anfang des Selektornamens auch entfallen kann, können Sie den Stil auch mit:

```
[lang] { text-align:right }
```

definieren. Den stellt auch Safari korrekt dar.

iCab 2.9.x unterstützt den Attribut-Selektor nur zum Teil. In Kombination mit dem Universal-Selektor (egal in welcher Form) wird der Stil nicht angewendet. Wenn Sie den Attribut-Selektor jedoch auf einen Element-Selektor anwenden, funktioniert es. Auch in iCab 3.0 funktioniert der Selektor.

Abbildung 2.5: Alle Absätze, die über ein lang-Attribut verfügen, werden jetzt rechtsbündig ausgerichtet.

INFO

Sie können den Attribut-Selektor natürlich nicht nur zusammen mit dem Universal-Selektor anwenden, sondern genauso mit Element-, Klassen- und ID-Selektoren. Wenn Sie beispielsweise nicht möchten, dass auch normale Absätze rechtsbündig ausgerichtet werden, weil sie ein `lang`*-Attribut haben, können Sie mit dem Stil* `h1[lang] {text-align:right}` *den gleichen Effekt nur für die Überschriften der ersten Ebene erzielen. Das stellt dann sogar iCab 2.9.x korrekt dar.*

Abbildung 2.6: Die Darstellung des Stils in iCab 2.9.x

Formate abhängig vom Attributwert

Nicht immer ist es aber sinnvoll, die Formatierung nur vom Vorhandensein des Attributs abhängig zu machen. Gerade beim `lang`-Attribut, mit dem Sie die Sprache eines Elements festlegen, wäre es sinnvoll, Sie würden die Formatierungen von dessen Wert abhängig machen.

CSS-Selektor:	Attribut-Selektor (»`[attribut=wert]`«)	*Beschreibung:*
CSS-Version:	2.0, 2.1, 3.0	Wählt ein Element `E` aus, das das angegebene Attribut `A` mit dem Wert `X` hat: `E[A=X]`.

Palm	NN	Mozilla		Internet Explorer							Opera			Safari		iCab	Kq	FF
2.x	4.x	1.x	1.8	4.x	5.0x (Win)	5.0 (Mac)	5.1+ (Mac)	5.5 (Win)	6	7	6	7	8	1.x	2.0	2.9+	3.x	1.0
		●	●								●	●	●	●	●	●[1]	●	●

[1] fehlerfrei erst ab iCab 3.0

Auf diese Weise können Sie beispielsweise dafür sorgen, dass die mit `lang="en"` definierte Überschrift kleiner und blasser und unterhalb der deutschen Überschrift angezeigt wird. Sie müssen dazu nur folgenden Stil zusätzlich zu dem zuvor erstellten Attribut-Selektor erstellen:

```
h1[lang]    { text-align:right }
h1[lang=en] { font-size:120%; position:relative;
              top:-20pt; color:gray }
```

Listing 2.9: Notwendige Stile zur rechtsbündigen Ausrichtung der Überschriften und zusätzlich zur Formatierung der englischsprachigen Überschriften

Korrekt sollte das Ergebnis dann aussehen, wie es Safari anzeigt:

iCab 2.9.x wendet den Stil an, ohne jedoch tatsächlich den Wert des Attributs zu berücksichtigen. Das führt dazu, dass nicht nur die Überschrift mit dem Wert »en« im Attribut `lang` formatiert wird, sondern auch die andere Überschrift. In iCab 3.0 funktioniert der Selektor aber einwandfrei.

Auch Amaya 9.1 hat wider Erwarten Probleme mit diesem Attribut-Selektor. Statt die zweite Überschrift rechtsbündig zu formatieren, wie dies der Stil `h1[lang] {text-align:right}` festlegt, wird zwar der zweite Attribut-Selektor angewendet, die Formatierungen des ersten jedoch nicht.

Abbildung 2.7: Safari zeigt die Formatierungen korrekt an

Abbildung 2.8: Fehlerhafte Anwendung des Attribut-Selektors in iCab

Text ist als Wort im Attributwert enthalten

Sie können aber auch Elemente selektieren und formatieren, die im Wert eines bestimmten Attributs eine bestimmte Zeichenfolge als separates Wort enthalten. Ein Wort ist definiert als eine Zeichenfolge, die von Leerzeichen umgeben ist oder durch das Ende oder den Anfang der Zeichenkette begrenzt wird.

CSS-Selektor:	Attribut-Selektor (»`[attribut~=wort]`«)	*Beschreibung:*
CSS-Version:	2.0, 2.1, 3.0	Wählt ein Element `E` aus, das das angegebene Attribut `A` hat, in dessen Wert das Wort `X` vorkommt: `E[A~=X]`.

Palm	NN	Mozilla		Internet Explorer							Opera			Safari		iCab	Kq	FF
2.x	4.x	1.x	1.8	4.x	5.0x (Win)	5.0 (Mac)	5.1+ (Mac)	5.5 (Win)	6	7	6	7	8	1.x	2.0	2.9+	3.x	1.0
●		●	●								●	●	●	●	●	●[1]	●	●

[1] ab iCab 3.0

Für diesen Attribut-Selektor verwenden Sie dann nicht den Vergleichsoperator = sondern ~=. Mit dem Selektor `img[alt~=logo]` würden Sie also einen Selektor für alle Bilder erstellen, die im `alt`-Attribut das Wort »logo« enthalten. Um das zu testen, müssen Sie natürlich in die HTML-Seite ein Bild einfügen und mit dem `alt`-Attribut versehen.

```
<img src="logoff.jpg" alt="Das ist das Logo für Firefox">
```

Um nun diese Grafik mit einem roten Rahmen zu versehen, benötigen Sie noch den folgenden Stil:

```
img[alt~="Logo"] { border:1px solid red }
```

!! STOP

Achten Sie bei dem gesuchten Wert im Attribut-Selektor unbedingt auf Groß- und Kleinschreibung. Würden Sie im vorliegenden Beispiel `[alt~="logo"]` *angeben, würde der Stil nicht auf die Grafik angewendet, weil das »Logo« im* `alt`*-Attribut mit einem Großbuchstaben beginnt.*

Text ist als Teilzeichenfolge im Attributwert enthalten

Analog dazu gibt es einen weiteren Attribut-Selektor, der Teilzeichenfolgen innerhalb des Attributwerts findet, die durch Bindestriche begrenzt werden. Dazu verwenden Sie im Selektor den Vergleichsoperator |=.

CSS-Selektor:	Attribut-Selektor (»`[attribut|=wort]`«)	Beschreibung:
CSS-Version:	2.0, 2.1, 3.0	Wählt ein Element `E` aus, das das angegebene Attribut `A` hat, in dessen Wert das Wort `X` vorkommt, das durch Bindestriche begrenzt ist: `E[A|=X]`.

Palm	NN	Mozilla		Internet Explorer							Opera			Safari		iCab	Kq	FF
2.x	4.x	1.x	1.8	4.x	5.0x (Win)	5.0 (Mac)	5.1+ (Mac)	5.5 (Win)	6	7	6	7	8	1.x	2.0	2.9+	3.x	1.0
		●	●								●	●	●	●	●	●[1]	●	●

[1] ab iCab 3.0

Mit CSS 3.0 gibt es weitere Attribut-Selektoren, die noch weitergehende Möglichkeiten bieten. Teilweise werden diese Attribut-Selektoren auch schon unterstützt, beispielsweise von den auf Mozilla 1.4+ basierenden Browsern und Konqueror.

Anfang und Ende von Attributwerten

CSS 3.0 kennt beispielsweise spezielle Vergleichsoperatoren, mit denen Sie prüfen können, ob ein bestimmter Wert den Anfang ^= oder das Ende $= eines Attributwertes darstellt.

CSS-Selektor:	Attribut-Selektor (»`[attribut^=anfang]`«)	*Beschreibung:*
CSS-Version:	3.0	Wählt ein Element `E` aus, das das angegebene Attribut `A` hat, dessen Wert mit `X` beginnt: `E[A^=X]`.

Palm	NN	Mozilla		Internet Explorer							Opera			Safari		iCab	Kq	FF
2.x	4.x	1.x	1.8	4.x	5.0x (Win)	5.0 (Mac)	5.1+ (Mac)	5.5 (Win)	6	7	6	7	8	1.x	2.0	2.9	3.x	1.0
		●[1]	●												●	●[2]	●	●

[1] ab Mozilla 1.4+
[2] fehlerfrei erst ab iCab 3.0

CSS-Selektor:	Attribut-Selektor (»`[attribut$=ende]`«)	*Beschreibung:*
CSS-Version:	3.0	Wählt ein Element `E` aus, das das angegebene Attribut `A` hat, dessen Wert mit `X` endet: `E[A$=X]`.

Palm	NN	Mozilla		Internet Explorer							Opera			Safari		iCab	Kq	FF
2.x	4.x	1.x	1.8	4.x	5.0x (Win)	5.0 (Mac)	5.1+ (Mac)	5.5 (Win)	6	7	6	7	8	1.x	2.0	2.9	3.x	1.0
		●[1]	●												●	●[2]	●	●

[1] ab Mozilla 1.4+
[2] fehlerfrei erst ab iCab 3.0

Mit dem Stil `a[href^="ftp://"] {color:red}` können Sie beispielsweise erreichen, dass Links auf FTP-Adressen rot dargestellt werden, da diese URLs immer mit »ftp://« beginnen. Folgendes Beispiel zeigt dies. Vorausgesetzt, Sie haben folgenden HTML-Code in der Seite definiert:

Listing 2.10: Aufbau der Link-Liste

```
<h2>FTP-Downloads (sollten rot formatiert werden)</h2>
<ul>
    <li><a href="ftp://xyz.de">Download 1</a></li>
    <li><a href="ftp://xyz.de">Download 2</a></li>
</ul>
<h2>HTTP-Links (sollten grün formatiert werden)</h2>
<ul>
    <li><a href="http://xyz.de">Download 1</a></li>
    <li><a href="http://xyz.de">Download 2</a></li>
</ul>
```

Dann können Sie mit den folgenden beiden Stilen dafür sorgen, dass FTP- und HTTP-Links mit unterschiedlichen Farben formatiert werden.

```
a[href^="ftp://"]  { color:red }
a[href^="http://"] { color:lime }
```

Listing 2.11: Notwendige Stile

So können Sie beispielsweise alle Links auf FTP-Adressen rot formatieren.

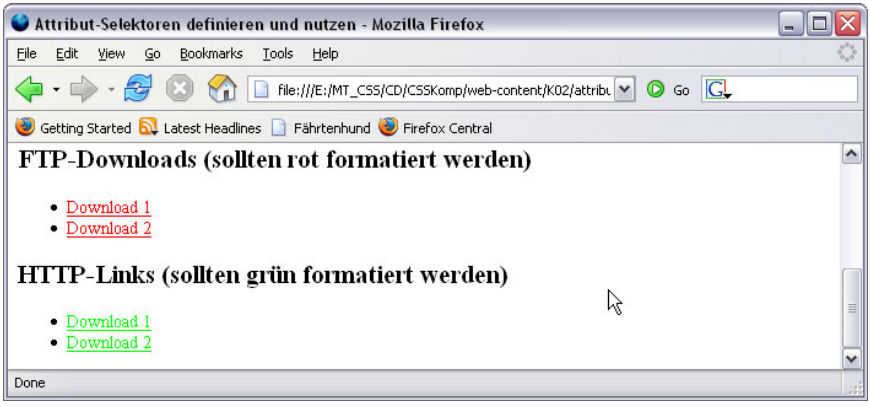


Abbildung 2.9: Das korrekte Ergebnis, hier in Firefox

iCab 2.9.x wendet den Attribut-Selektor an, allerdings auf alle Links, unabhängig vom Attributwert des `src`-Attributs. Es scheint so, als ob iCab nur prüft, ob das Attribut vorhanden ist, und dann den Stil anwendet. Somit behandelt iCab den Stil genauso wie den Attribut-Selektor `[Attribut]`. In iCab 3.0 funktionieren beide Attribut-Selektoren problemlos.

Näheres dazu finden Sie weiter oben im Abschnitt »Attribut-Selektoren«.

REF

Teilzeichenfolgen ermitteln

Analog dazu können Sie mit dem Vergleichsoperator *= feststellen, ob der Attributwert die angegebene Teilzeichenfolge enthält. Diese muss nicht durch Leerzeichen oder Bindestriche abgetrennt sein.

CSS-Selektor:	Attribut-Selektor (»`[attribut*=teilzeichenfolge]`«)	*Beschreibung:*
CSS-Version:	3.0	Wählt ein Element `E` aus, das das angegebene Attribut `A` hat, dessen Wert die Zeichenfolge `X` enthält: `E[A*=X]`.

Palm	NN	Mozilla		Internet Explorer							Opera			Safari		iCab	Kq	FF
2.x	4.x	1.x	1.8	4.x	5.0x (Win)	5.0 (Mac)	5.1+ (Mac)	5.5 (Win)	6	7	6	7	8	1.x	2.0	2.9+	3.x	1.0
		●[1]	●												●	●[2]	●	●

[1] ab Mozilla 1.4+
[2] fehlerfrei ab iCab 3.0

Sie können beispielsweise folgenden Stil erstellen, wenn Sie alle Links, die im Attribut `href` die Zeichenfolge »xyz.de« enthalten, fett formatieren möchten.

```
a[href*="xyz.de"] { font-weight:bold }
```

:-) TIPP

Einen solchen Stil können Sie sehr schön nutzen, um Links auf die eigene Seite anders zu formatieren, als Links auf fremde Seiten. Als Teilzeichenfolge geben Sie dann einfach den Domain-Namen an.

Selektieren nach fehlenden Attributen

Sie können sogar nach fehlenden Attributen suchen. Dazu stellt CSS 3 eine Pseudo-Klasse `:not` zur Verfügung, die Sie mit einem Attribut-Selektor kombinieren müssen. Gerade für die Fehlersuche ist das interessant.

CSS-Selektor:	Attribut-Selektor (»`not[attribut]`«)	*Beschreibung:*
CSS-Version:	3.0	Wählt ein Element `E` aus, das das angegebene Attribut `A` nicht hat: `E:not[A]`.

Palm	NN	Mozilla		Internet Explorer							Opera			Safari		iCab	Kq	FF
2.x	4.x	1.x	1.8	4.x	5.0x (Win)	5.0 (Mac)	5.1+ (Mac)	5.5 (Win)	6	7	6	7	8	1.x	2.0	2.9+	3.x	1.0
		●[1]	●												●		●	●

[1] ab Mozilla 1.4+

Wenn Sie beispielsweise alle Bilder markieren möchten, die kein `width`-Attribut haben, definieren Sie dazu folgenden Stil:

```
img:not([width]) { border:3px solid yellow }
```

Nun werden alle Bilder ohne `width`-Attribut mit einem 3 Pixel starken gelben Rahmen umgeben.

Attribut-Selektoren kombinieren

Attribut-Selektoren lassen sich auch miteinander kombinieren. Sie können beispielsweise festlegen, dass zwei Attribute vorhanden sein sollen oder dass das erste Attribut einen bestimmten Wert haben und ein zweites vorhanden sein muss.

CSS-Selektor:	Kombinierter Attribut-Selektor (»`*[attribut-selektor][attributselektor]`«)	*Beschreibung:* Wählt ein Element `E` aus, auf das alle angegebenen Attribut-Selektoren `A1` und `A2` zutreffen: `E[A1][A2]`.
CSS-Version:	2.0, 2.1, 3.0	

Palm	NN	Mozilla		Internet Explorer							Opera			Safari		iCab	Kq	FF
2.x	4.x	1.x	1.8	4.x	5.0x (Win)	5.0 (Mac)	5.1+ (Mac)	5.5 (Win)	6	7	6	7	8	1.x	2.0	2.9	3.x	1.0
		● 1+2	● 2								● 2	● 2	● 2	● 2	● 2	● 2+3	● 2	● 2

1 ab Mozilla 1.4+
2 vorausgesetzt der Browser unterstützt alle verwendeten Attribut-Selektoren
3 ab iCab 3.0

Dazu geben Sie die Attribut-Selektoren einfach nacheinander an. Mit dem folgenden Stil können Sie beispielsweise alle Links formatieren, die sowohl über das `title`- wie über das `href`-Attribut verfügen. Die CSS-Eigenschaften führen dazu, dass die Standard-Unterstreichung entfernt wird (`text-decoration:none`) und dafür unten eine gepunktete Rahmenlinie von einem Pixel Stärke rot angezeigt wird.

```
a[href][title] {
    text-decoration:none;
    border-bottom:1px dotted red
}
```

Listing 2.12: Der Stil mit zusammengesetztem Attribut-Selektor

Tabellen-Selektoren

In CSS 3.0 gibt es eine weitere Selektorgruppe, die Tabellen-Selektoren. Genau genommen sind dies Kombinationen von Pseudo-Klassen und arithmetischen Ausdrücken.

CSS-Selektor:	Tabellen-Selektoren (`table.row[zeile]` bzw. `table.column[Spalte]`)	*Beschreibung:* Wählt eine bestimmte Zeile `Z` oder Spalte `S` einer Tabelle `T` aus: `T.row[Z]` bzw. `T.column[S]`.
CSS-Version:	3.0	

Palm	NN	Mozilla		Internet Explorer							Opera			Safari		iCab	Kq	FF
2.x	4.x	1.x	1.8	4.x	5.0x (Win)	5.0 (Mac)	5.1+ (Mac)	5.5 (Win)	6	7	6	7	8	1.x	2.0	2.9+	3.x	1.0
			[1]															[1]

[1] Entgegen anders lautenden Angaben in der Literatur funktionieren diese Attribute in keinem Browser, auch nicht in Mozilla 1.8+ und Firefox 1.0+.

Das Beispiel zu den Tabellen-Selektoren finden Sie in der Datei BSP/K02/Tabellenselektoren.html. *Die zugehörigen Stile stehen in der Datei* BSP/K02/vorlage.css.

Eine bestimmte Zeile oder Spalte formatieren

Möchten Sie für bestimmte Zeilen oder Spalten, z.B. die erste Zeile, eine Formatierung festlegen, erstellen Sie dazu den Stil `table.row[1]{}`. Die Zeilennummer steht also immer in eckigen Klammern. Der Selektor `table.row` gibt an, dass Zeilen formatiert werden sollen. Analog dazu heißt der Stil zum Formatieren der zweiten Spalte `table.column[2]`.

Jede zweite Zeile formatieren

Sie können sogar festlegen, dass jede zweite Zeile/Spalte ein bestimmtes Format bekommt. Dazu geben Sie anstelle der Zeilen- oder Spaltennummer »%2« an. Soll dieses Format erst in einer bestimmten Zeile beginnen, nennen Sie diese Zeile dahinter. Mit `table.row[%2+3]` würde jede zweite Zeile (beginnend mit der dritten) formatiert werden. Auf diese Weise könnten Sie komplexe Tabellenformate erzeugen. Im folgenden Listing wird die erste Zeile mit rotem Hintergrund formatiert, die erste Spalte bekommt einen schwarzen Hintergrund und eine weiße Schrift und jede zweite Zeile wird mit gelbem Hintergrund formatiert.

Listing 2.13: Tabellen mit Tabellen-Selektoren formatieren

```
table           { border:1px solid black; empty-cells:show }
td              { border:1px dotted gray; }
table.row[1]    { background-color:red; }
table.row[%2]   { background-color:yellow }
table.column[1] { background-color:black;color:white }
```

Beachten Sie hier die Reihenfolge der Stile. Nur, wenn Sie die Formatierung für die erste Spalte an das Ende setzen, wird später auch die erste Spalte durchgängig schwarz sein. Ansonsten würde der Stil `table.row[%2]` *die schwarze Füllfarbe überschreiben und in jeder zweiten Zeile wäre die erste Spalte gelb.*

Fehlerverhalten standardkonformer Browser

Das Beispiel finden Sie in der Datei BSP/K02/Tabellenselektoren.html. *Die zugehörigen Stile stehen in der Datei* BSP/K02/vorlage.css.

Gemäß CSS-Spezifikation sollen Browser ungültige Selektoren komplett ignorieren. Ungültig meint in diesem Zusammenhang, dass sie syntaktisch nicht korrekt sind, weil z.B. die Selektoren ungültige Zeichen enthalten. Bei gruppierten Selektoren sollen alle Selektoren einer Gruppe ignoriert werden, wenn in der Gruppe ein Syntaxfehler vorhanden ist. Allerdings machen das nicht alle Browser.

CSS:	Korrektes Fehlerverhalten bei syntaktisch falschen Selektoren	*Beschreibung:* Browser sollen die ganze Gruppe von Selektoren ignorieren, wenn einer davon syntaktisch falsch ist.
CSS-Version:	1.0, 2.0, 2.1, 3.0, TV, Mobile	

Palm	NN	Mozilla		Internet Explorer							Opera			Safari		iCab	Kq	FF
2.x	4.x	1.x	1.8	4.x	5.0x (Win)	5.0 (Mac)	5.1+ (Mac)	5.5 (Win)	6	7	6	7	8	1.x	2.0	2.9	3.x	1.0
●	◎ [1]		●			●	●		●			●	●		●	● [2]	●	●

1 Der Netscape Navigator verhält sich abhängig von der Art der Fehler teilweise korrekt.
2 ab iCab 3.0

Alle Browser, die kein korrektes Fehlerverhalten aufweisen, stellen in einer Selektorgruppe die korrekten Selektoren dar, auch wenn die Gruppe syntaktisch falsche Selektoren enthält.

Definieren Sie beispielsweise folgende Selektorgruppe, ist diese fehlerhaft, da `.&17` kein gültiger Klassenname ist, unabhängig davon, ob die Klasse verwendet wird.

```
p, h1, .&17 { color:red }
```

Bei korrektem Fehlerverhalten dürfte der Browser auch die Elementstile `p` und `h1` nicht anwenden.

2.2 Vererbung von Stilen

Erst durch die Möglichkeit, Stile und Eigenschaften zu vererben und zu überschreiben, lässt sich CSS so richtig komfortabel und flexibel einsetzen. Beides gehört zu den wesentlichen Konzepten von CSS. Und genau hier gibt es auch größere Problem mit einigen Browsern, die zu starken Differenzen in der Darstellung führen. Zu diesen problematischen Browsern gehören vor allem iCab 2.9.x und niedriger sowie der Netscape Navigator 4.x-.

Vererbung

Vererbung besagt, dass CSS-Formatierungen, die Sie für ein bestimmtes Element festlegen, auch auf untergeordnete und verwandte Elemente angewendet werden. Wenn Sie Vererbung geschickt nutzen, können Sie sich damit eine Menge Aufwand sparen und Ihre Stylesheets noch wartungsfreundlicher gestalten.

Überschreiben von Werten

Mit Überschreibung ist ein anderes Verhalten gemeint. Sie können einmal definierte Eigenschaften durch eine Neudefinition etwas später im Stylesheet überschreiben. Neben der Reihenfolge, in der Sie die Stile definieren, spielt aber auch die Spezifität des Selektors eine Rolle, die sich berechnen lässt.

Mehr zur Spezifität der Selektoren finden Sie weiter unten im Abschnitt »Spezifität und Kaskadenreihenfolge der Selektoren«.

Dokument-Formatierung im Detail

Wichtig für das Verständnis der Vererbung und Überschreibung von Eigenschaften ist das Wissen darüber, wie ein Browser oder ein anderes Programm, das die Webseite ausgibt, die Formatierung des Dokuments vornimmt. Generell sind dazu mehrere Schritte erforderlich.

Abrufen und parsen

Zunächst ruft das Programm die gewünschte Webseite vom Server ab und erhält daraufhin den HTML-Code. Um diesen nun darstellen zu können und so Überschriften von Absätzen und anderen Elementen zu unterscheiden, wird das Dokument eingelesen und dessen Syntax geprüft. Dieser Vorgang wird Parsen genannt. Beim Parsen baut das Programm einen virtuellen Dokumentenbaum auf. Das ist eine Baumstruktur, in der alle Elemente des Dokuments vorhanden sind. Anhand dieses Dokumentenbaums erfolgt die weitere Verarbeitung.

Formatierungen berechnen

Nach dem Parsen beginnt die Formatierung des Dokuments, die natürlich abhängig vom Ausgabemedium erfolgt. Unter Berücksichtigung der Formatierungen, die ausschließlich für das passende Medium gelten, berechnet das Anzeigeprogramm nun für alle Elemente des Dokumentenbaums deren Eigenschaftswerte. Bestandteil der Berechnung sind neben den Standardwerten des Browsers auch statische und berechnete Werte aus den Stylesheets.

Die Berechnung der Werte verläuft wiederum in drei Teilschritten. Zunächst wird die Spezifizierung des Stils ermittelt. Das ergibt einen spezifischen Wert, der dann in einen absoluten umgerechnet wird, falls dies nötig und möglich ist. Der so berechnete Werte wird dann entsprechend des Ausgabemediums in einen tatsächlichen Wert umgerechnet.

Erst wenn alle Werte berechnet sind, werden die Formatierungen auf den Dokumentenbaum angewendet und die Seite wird ausgegeben.

Spezifität und Kaskadenreihenfolge der Selektoren

Um die Spezifität eines Selektors zu berechnen, ist die Kaskade der Stylesheets zu berücksichtigen. Als Kaskade wird die Gewichtung der verschiedenen Ursprünge eines Stylesheets bezeichnet. Stylesheets kommen in drei Formen vor, deren Reihenfolge auch deren Bedeutung wiedergibt.

Kaskadierung

Die drei Quellen, aus denen Stylesheets stammen, sind Autor, Benutzer und Benutzerprogramm (Ausgabeprogramm oder Browser). Als Autor legen Sie die Formatierung in Form von CSS-Dateien oder als Stylesheet innerhalb des Dokuments fest. Dies ist die oberste Stufe der Kaskade. Autoren-Stylesheets haben daher höchste Priorität. Darunter liegen Benutzer-Stylesheets, die der Benutzer des Ausgabeprogramms erstellen kann, sofern das Programm diese Funktion zur Verfügung stellt. Benutzer-Stylesheets sind den Autoren-Stylesheets untergeordnet, gehen also bei der Abarbeitung vor.

Als letzte Gruppe gibt es die Stylesheets des Ausgabeprogramms. Dabei handelt es sich um ein Standard-Stylesheet, das Default-Werte für alle Eigenschaften beinhaltet. Dieses Stylesheet muss nicht zwingend als CSS-Datei vorliegen. Es reicht aus, wenn der Browser sich so verhält, als gäbe es ein Standard-Stylesheet. Das Standard-Stylesheet hat die niedrigste Priorität.

Bei der Anwendung der Stile wird die Kaskade wie folgt berücksichtigt. Nehmen Sie an, Sie haben als Autor einen Stil

```
p { color:red }
```

erstellt, der Benutzer hat jedoch einen Stil

```
p { color:white }
```

festgelegt und der Standard-Stil des Anzeige-Programms lautet:

```
p { color:black }
```

In diesem Fall würde die Schrift der Absätze rot angezeigt, weil das Autoren-Stylesheet Vorrang hat und die Schriftfarbe aus diesem Stil verwendet wird.

Die Beispiele finden Sie in den Dateien BSP/K02/kaskadierung.html *und* BSP/K02/formate.css.

Wenn eine Seite mehrere CSS-Dateien oder Stylesheets nutzt, hängt deren Gewichtung innerhalb der Autoren-Stile von der Reihenfolge der Definition ab. Nehmen Sie an, Sie haben eine Datei mit folgendem `head`-Bereich:

Listing 2.14: Die Einbindung der Stile über link und style

```
<head>
    <meta http-equiv="content-type"
          content="text/html;charset=iso-8859-1">
    <title>Kaskadierung und Vererbung</title>
    <link href="formate.css"
          rel="stylesheet"
          type="text/css">
    <style type="text/css">
    <!--
       p { color:blue }
    -->
    </style>
</head>
```

Steht dann in der CSS-Datei der Stil:

```
p { color:red )
```

bedeutet dies, dass die Schrift normaler Absätze in Blau ausgegeben wird, weil der Stil im `style`-Element den Wert aus der CSS-Datei ersetzt. Würden Sie die Reihenfolge von `style`- und `link`-Element tauschen, würde die Schrift hingegen rot angezeigt, weil dann die CSS-Datei die höhere Priorität hat.

Je später das Stylesheet im Quellcode der Seite auftaucht, desto höhere Priorität hat es. Gleiches gilt auch für die Definition von Stilen innerhalb der Stylesheets. Sie können innerhalb eines Stylesheets mehrmals den gleichen Selektor definieren. Bei widersprüchlichen Angaben gilt dann immer die zuletzt vom Parser gelesene Angabe.

Details und Beispiele dazu folgen im Abschnitt »Überschreiben von Werten« weiter unten.

Für die Berechnung der Kaskadenreihenfolge wird zunächst der Ursprung, dann die Spezifität und anschließend, wenn weder Ursprung noch Spezifität einen Vorrang ergeben, die Definitionsreihenfolge der Stile herangezogen.

Bei der Spezifität des Selektors wird berücksichtigt, wie allgemein verbindlich oder speziell er ist. Spezifischere Selektoren überschreiben allgemeinere Selektoren, und Pseudo-Elemente und Pseudo-Klassen werden als normale

Elemente bzw. Klassen betrachtet. Daher ist die Spezifität für ID-Stile größer als die für Klassenstile und diese wiederum größer als die von Elementstilen. Es gibt sogar eine Formel zur Berechnung der Spezifität. Dabei wird die Anzahl ID-Selektoren in einem Stil als 100er-Stelle gezählt, die Klassen-Selektoren als 10er-Stelle und die Element-Selektoren als 1er-Stelle. Folgendes Beispiel soll das verdeutlichen. Wenn Sie folgenden Selektor definieren:

```
#banner span
```

gibt es einen ID-Selektor. Die 100er-Stelle ist also 1. Außerdem gibt es einen Element-Selektor und keinen Klassen-Selektor. Damit ist die 10er-Stelle 0 und die 1er-Stelle 1. Zusammengesetzt ergibt das die Zahl 101.

Analog dazu hat der Stil

```
#banner #logo .schrift
```

den Wert 210, da er zwei ID-Selektoren und einen Klassen-Selektor enthält. Somit wäre dieser zweite Stil spezifischer als der erste und würde im Zweifelsfall die Werte des ersten überschreiben.

Berücksichtigt werden bei der Berechnung nur die Selektoren, die zusammengesetzt einen komplexen Selektor bilden. Wenn Sie Selektoren mittels Komma gruppieren, wird jeder einzelne Selektor berechnet, weil diese Selektoren ja auch einzeln notiert werden könnten.

Alle Formatierungen innerhalb von `style`*-Attributen eines (X)HTML-Elements sind höherwertiger als Stile im* `style`*-Element einer Seite oder in verknüpften CSS-Dateien.*

Gewichtung anpassen

Sie können auch Einfluss auf die Gewichtung der Stile nehmen, um beispielsweise in einem Benutzer-Stylesheet Stile des Autors zu überschreiben. Dazu gibt es die `!important`-Regel.

Mit der `!important`-Regel können Sie festlegen, dass diese Einstellung Priorität hat. Das kann sowohl in einem Benutzer-Stylesheet als auch in einem Autoren-Stylesheet geschehen. In jedem Fall muss hinter jeder CSS-Eigenschaft, die Priorität haben soll, das Schlüsselwort `!important` stehen.

Wenn es in einem Benutzer-Stylesheet eine CSS-Eigenschaft gibt, die als `important` *markiert ist, hat diese Vorrang gegenüber einer anders lautenden Angabe in einem Autoren-Stylesheet.*

!! STOP

Wenn eine CSS-Eigenschaft sowohl im Benutzer- wie im Autoren-Stylesheet als `important` *markiert ist, hat die Regel des Benutzer-Stylesheets Vorrang vor den Einstellungen des Autors. Damit kommt CSS Benutzern mit Behinderungen entgegen, die beispielsweise durch eine eigene Schriftgröße in Benutzer-Stylesheets so besser lesbare Seiten erhalten.*

!! STOP

Diese Regelung ist allerdings erst ab CSS 2.1 gültig. In vorherigen Versionen von CSS haben `!important`*-Regeln des Autors die* `!important`*-Regeln des Benutzers überschrieben. Seit CSS 2.1 ist dies umgekehrt.*

Wenn Sie eine CSS-Eigenschaft als `important` markieren möchten, sieht das beispielsweise folgendermaßen aus:

```
p { color:red; font-size:10pt !important }
```

Hinter den Wert, den Sie als `important` markieren möchten (hier die Schriftgröße: `font-size`), setzen Sie ein Leerzeichen, danach ein Ausrufezeichen und dann das Schlüsselwort `important`.

f INFO

Diese höhere Priorität wirkt sich nicht nur auf das Verhältnis der verschiedenen Stylesheet-Quellen aus, sondern auch auf die Reihenfolge der Kaskadierung, die sich aus der Reihenfolge der Stildefinitionen ergibt. Wenn Sie erst die obige Regel definieren und etwas weiter unter im Stylesheet die Regel `p {font-size:20pt}`*, wird der Text in 10 Punkt Größe dargestellt.*

Der Netscape Navigator 4.x wendet die `!important`-Regel nicht korrekt an. Innerhalb eines Stylesheets hat weiterhin der spätere Stil Vorrang.

Überschreiben von Werten

Befinden sich innerhalb eines Stylesheets widersprüchliche Angaben zu einem Element, regelt die Priorität des Selektors, welche Werte verwendet werden. Folgendes Beispiel soll das zeigen.

Zunächst definiert der erste Stil die Schriftfarbe Blau mit der Schriftfamilie `sans-serif`. Der nächste Selektor, der sich speziell auf einen Absatz mit der ID `hinweis` bezieht, legt nun aber eine rote Schriftfarbe und zusätzlich den Schriftschnitt Fett (`font-weight:bold`) fest.

Listing 2.15: Beispiel zum Überschreiben von Werten

```
...
<head>
    <style type="text/css">
    <!--
        p        { color:blue; font-family:sans-serif }
        #hinweis { color:red; font-weight:bold }
    -->
    </style>
```

```
</head>
<body>
    <h1>Beispiele zur Kaskadierung und 
    Vererbung</h1>
    <p id="hinweis">Beherrscht der Browser das
    Überschreiben von Stilen, sollte dieser Text in
    einer serifenlosen Schrift, mit roter Farbe und
    in fett erscheinen.</p>
</body>
...
```

Was in diesem Fall genau passiert, ist Folgendes: Zunächst wird der Stil des `p`-Elements auf den Absatz mit der ID `hinweis` angewendet, da es sich ja bei diesem Absatz um ein `p`-Element handelt. Das führt dazu, dass dem Absatz die blaue Schriftfarbe und die Schriftfamilie `sans-serif` zugewiesen wird. Danach wird der ID-Stil angewendet und legt die Schriftfarbe auf Rot fest. Die blaue Schriftfarbe wird nun also durch die rote ersetzt, weil der ID-Stil ranghöher ist und außerdem auch nach dem Elementstil definiert ist. Außerdem wird die Schriftstärke auf Fett gesetzt.

Sie sehen also, dass nur die widersprüchlichen Angaben durch die Kaskadenrangfolge bestimmt werden. Die ranghöheren Stile überschreiben dabei die Werte in den rangniedrigeren. Alle nicht widersprüchlichen Angaben werden nur ergänzt und so zusammengefasst.

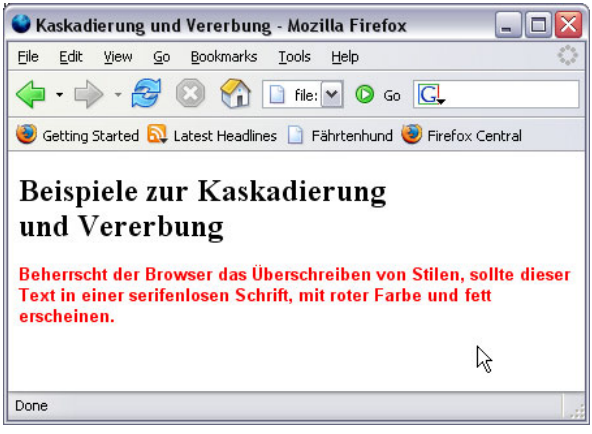


Abbildung 2.10: Korrekte Darstellung in Firefox

Den gleichen Effekt erzielen Sie, indem Sie alle Eigenschaften aus beiden Stilen im ID-Stil zusammenfassen:

```
#hinweis {
    color:red;
    font-weight:bold;
    font-family:sans-serif
}
```

Listing 2.16: Alternative Schreibweise ohne Überschreibung von Werten

INFO

Selbstverständlich hat die erste Deklaration dennoch ihren Sinn. Schließlich legt sie mit dem Elementstil für das `p`*-Element die Formatierung für alle Absätze fest; die zusammengefasste Definition definiert jedoch nur die Absätze mit der ID* `hinweis`*.*

Vererbung im Detail

Neben dem Überschreiben von Werten gibt es ein weiteres, wichtiges Konzept in CSS – die Vererbung. Gemeint ist damit, dass Sie bestimmte Eigenschaften nur für übergeordnete Elemente der Webseite definieren müssen und das diese automatisch oder durch entsprechende Angaben im Stil an untergeordnete Elemente weitergegeben werden.

CSS:	Vererbung einfacher Eigenschaften	*Beschreibung:* Werte von Eigenschaften werden an die untergeordneten Elemente vererbt, sofern sie für diese Elemente nicht explizit festgelegt werden.
CSS-Version:	1.0, 2.0, 2.1, 3.0, TV, Mobile	

Palm	NN	Mozilla		Internet Explorer							Opera			Safari		iCab	Kq	FF
2.x	4.x	1.x	1.8	4.x	5.0x (Win)	5.0 (Mac)	5.1+ (Mac)	5.5 (Win)	6	7	6	7	8	1.x	2.0	2.9+	3.x	1.0
●		●	●	●	●	●	●	●	●	●	●	●	●	●	●	●[1]	●	●

[1] ab Version 3.0

CD

Sie finden das Beispiel als Datei BSP/K02/vererbung.html *auf der Buch-CD.*

Sinnvoll ist dies vor allem, wenn Sie grundlegende Formatierungen für alle Elemente der Seite angeben möchten. Dann sollten Sie dazu einen Elementstil für das Element `html` erstellen. Dies ist das oberste Element der Seite, dem also alle anderen Elemente untergeordnet sind.

Listing 2.17: HTML-Code des Beispiels

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01//EN">
<html>
    <head>
        <meta http-equiv="content-type"
            content="text/html;charset=iso-8859-1">
        <meta name="generator" content="Adobe
           GoLive">
        <title>Vererbung</title>
        <style type="text/css">
        <!--
            html { font-family:sans-serif }
```

```
        -->
        </style>
    </head>

    <body>
        <h1>Beispiele zur Vererbung</h1>
        <p>Dieser Absatz müsste mit der vererbten
        Schriftfamilie sans-serif formatiert
        sein.</p>
    </body>
</html>
```

Allein durch die Definition der Schriftart für das `html`-Element werden nun alle Elemente der Seite mit einer serifenlosen Schrift formatiert.

Abbildung 2.11: Korrekt stellt Firefox die Seite dar.

Sowohl iCab (2.9.x und niedriger) als auch Netscape Navigator 4.x haben Probleme mit der Vererbung. Beide Browser stellen im Beispiel die Schrift in den Standardschriften der Browser dar. Da dies in iCab eine serifenlose Schrift ist, hat es zunächst den Anschein, als würde iCab die Seite korrekt ausführen. Fügen Sie jedoch dem Stil eine Anweisung hinzu, die die Schriftfarbe ändert, sehen Sie, dass iCab die Eigenschaften nicht auf die untergeordneten Elemente vererbt.

Nicht alle Eigenschaften können überhaupt vererbt werden. Ob das der Fall ist, wird im Referenzteil zu jeder Eigenschaft angegeben.

Solange Sie nur Eigenschaften vererben möchten, die Textwerte haben, ist das verhältnismäßig unproblematisch. Bei prozentualen Größen wird es da schon etwas komplizierter. Nehmen Sie an, Sie möchten im Element-Selektor für das `html`-Element die Schriftgröße auf 150% festlegen. Die prozentuale Angabe bezieht sich dann auf die Standardschriftgröße des Browsers. Wenn diese 12 Punkt beträgt, bedeutet dies, dass die Schrift im Dokument nun mit einer 18 Punkt Schrift formatiert wird.

Es gibt zwei Möglichkeiten, wie dieser Wert auf die untergeordneten Elemente vererbt werden kann: als prozentualer Wert oder als absoluter, das heißt, berechneter Wert. Wird er als prozentualer Wert vererbt, bedeutet

das, dass beispielsweise auch Überschriften um 50% größer werden als standardmäßig. Somit würden sich alle Schriftgrößen um 50% erhöhen. Das Größenverhältnis zwischen Überschriften und Absätzen müsste dann also gleich bleiben.

Wird nur der berechnete Wert vererbt, bedeutet dies, dass alle Texte, einschließlich der Überschriften, nun die Größe 18 Punkt haben. Es gibt also keinen Größenunterschied mehr zwischen Überschriften und Absätze.

CSS:	Vererbung relativer Werte	*Beschreibung:*
CSS-Version:	1.0, 2.0, 2.1, 3.0, TV, Mobile	Bei Werten mit relativen Einheiten darf nicht die Größenangabe an sich, sondern nur der berechnete Werte vererbt werden.

Palm	NN	Mozilla		Internet Explorer							Opera			Safari		iCab	Kq	FF
2.x	4.x	1.x	1.8	4.x	5.0x (Win)	5.0 (Mac)	5.1+ (Mac)	5.5 (Win)	6	7	6	7	8	1.x	2.0	2.9+	3.x	1.0
●		○ [1]	○ [1]	○ [2]	○ [2]	○ [2]	○ [2]	○ [2]	○ [2]	○ [2]	○ [1]	○ [1]	○ [1]	○ [1]	○ [1]	● [3]	○ [1]	○ [1]

1 vererbt die Ursprungsgröße, nicht die berechnete
2 Die Schriftgröße in prozentualen Werten wird nicht auf Überschriften vererbt.
3 ab iCab 3.0

Gemäß CSS-Standard werden die berechneten Werte vererbt, nicht die prozentualen Werte. Allerdings halten sich zur Zeit nur Amaya, iCab 3.0 und der Palm-Browser wirklich an diese Vorgaben.

Alle großen Browser, mit Ausnahme des Internet Explorers, verhalten sich insofern fehlerhaft, als nicht die berechneten Werte, sondern die prozentualen Ausgangswerte vererbt werden. Da sich die Browser diesbezüglich wenigstens einheitlich verhalten, ist das kein größeres Problem.

Die Internet Explorer bis einschließlich Version 7 (auch die Mac-Version) vererben Eigenschaften zwar grundsätzlich an alle untergeordneten Elemente, nicht jedoch die Schriftgröße. Bei der Schriftgröße erben die Überschriften die Einstellung nicht. Sie behalten ihre Größe, während die Schriftgröße in Absätzen korrekt angepasst wird.

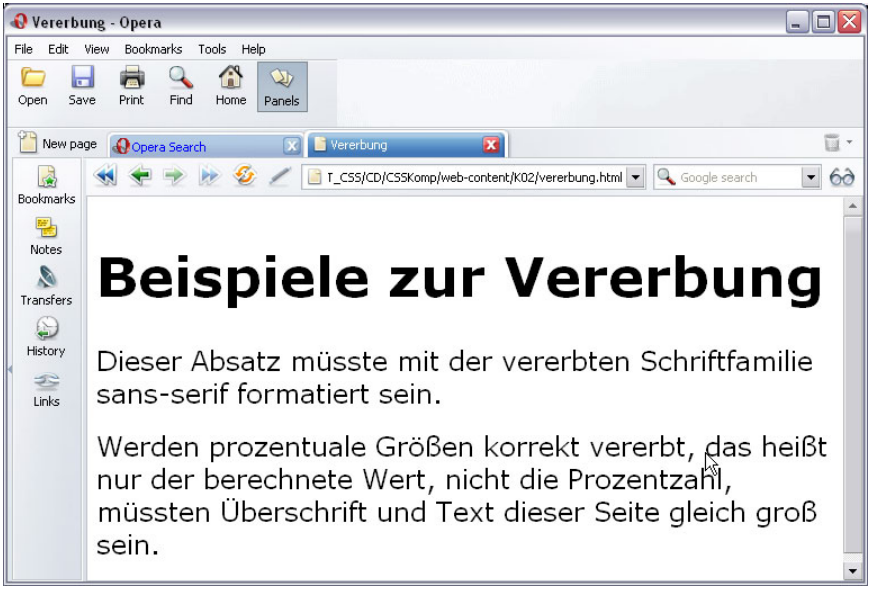


Abbildung 2.12: Durch Vererbung der Ausgangswerte, nicht der berechneten Werte, bleibt der Größenunterschied zwischen Überschriften und Text erhalten.

Vererbung erzwingen

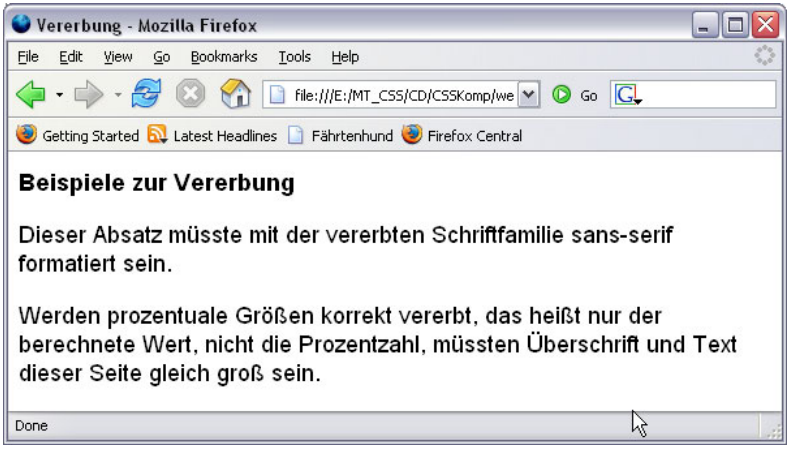
Die Vererbung, wie sie oben beschrieben wurde, dient lediglich dazu, Eigenschaftswerte zu ermitteln, die nicht explizit angegeben werden. Sie können einer Eigenschaft aber auch den Wert `inherit` zuweisen, um zu erzwingen, dass sie ihren Wert vom übergeordneten Element erbt. Wenn Sie beispielsweise sicherstellen möchten, dass auch für die Überschriften der berechnete Wert verwendet wird, geben Sie an:

```
h1 { font-size:inherit }
```

Wenn Sie für das `h1`-Element explizit die Vererbung erzwingen, vererben Firefox, Safari, Mozilla, Netscape, der Opera-Browser und der Internet Explorer 5.2 für Mac tatsächlich den berechneten Wert und zeigen die Schriftgrößen korrekt an.

Die Windows-Version des Internet Explorers lässt sich von diesem kleinen Trick nicht positiv beeinflussen und zeigt weiterhin das falsche Ergebnis an.

Abbildung 2.13: So muss die Seite bei korrekt vererbter Schriftgröße aussehen.



2.3 Zahlen und Maßeinheiten

Es gibt viele CSS-Eigenschaften, die numerische Werte haben. Meist sind dies Größenangaben für Höhen, Breiten, Abstände oder auch Schriftgrößen.

INFO

In der Regel gilt, dass numerische Werte (außer 0) nur mit einer Maßeinheit zusammen angegeben werden dürfen.

Vor einem numerischen Wert kann ein Vorzeichen stehen (je nach Eigenschaft, der der Wert zugewiesen wird), also entweder ein + oder -. Geben Sie keines an, wird dies als + interpretiert. Sowohl zwischen Vorzeichen und Zahl, als auch zwischen Zahl und Einheit darf kein Leerzeichen stehen. Falls es sich um eine Dezimalzahl handelt, wird der Punkt als Dezimaltrennzeichen verwendet.

Relative und absolute Maße

Bei Werten, die Längen, Breiten oder Abstände angeben, aber auch bei Schriftgrößen gibt es relative und absolute Werte. Um was es sich handelt, hängt von der verwendeten Maßeinheit ab.

Bei relativen Einheiten beziehen sich die Angaben immer auf das übergeordnete Element. Alle relativen Einheiten müssen zunächst berechnet werden, damit der Browser die Länge darstellen kann. Bei absoluten Werten ist das nicht erforderlich, weil die absolute Länge direkt verfügbar und nicht kontextabhängig ist.

Je nach Ausgabegerät kann es jedoch sein, dass der berechnete oder von vornherein absolute Wert noch in einen gerätespezifischen Wert konvertiert werden muss.

Als relative Längeneinheiten kennt CSS `em`, `ex` und `px`. Absolut sind die Einheiten `in` (Zoll), `cm`, `mm`, `pt` und `pc` (Pica).

Pica, Point und Pixel

Pixel und Point werden oftmals synonym verwendet. Viele Webdesigner gehen davon aus, dass ein Pixel und ein Punkt dasselbe seien, eben ein Bildpunkt. Das ist soweit zwar richtig, sie unterscheiden sich jedoch in der Größe der Punkte voneinander. Point ist eine absolute Maßeinheit. Unabhängig vom Ausgabemedium ist ein Point immer 1/72 Zoll groß. Damit hat ein Punkt ca. 0,35 mm Durchmesser. Beim Pixel sieht das anders aus. Zwar stellt auch ein Pixel einen Bildpunkt dar, dennoch handelt es sich hier um eine relative Größe. Relativ ist sie jedoch nur bezüglich des Ausgabemediums. Das heißt, ein Pixel auf einem Bildschirm hat eine andere Größe als ein Pixel auf dem Drucker. Allerdings hat ein Pixel auf verschiedenen (gleich großen) Bildschirmen immer die gleiche Größe, sofern die Auflösung die gleiche ist.

Das Problem an Angaben in Pixeln ist also, dass Sie nicht wissen können, mit welcher Auflösung ein Besucher die Seite betrachtet oder mit welcher Auflösung er sie druckt. Falls Sie beispielsweise eine Schrift mit `12px` angeben, kann sie auf einem Monitor mit einer Auflösung von 640 x 480 durchaus sehr gut lesbar sein. Auf einem Macintosh-Bildschirm mit einer Auflösung von 1240 x 1024 und 96 dpi ist diese Schrift hingegen nicht mehr lesbar. Und noch schlimmer wird es auf hochauflösenden Displays von Mobile-Geräten, die zum Teil 133 dpi verwenden. Dort ist die Schrift dann sicherlich nur noch mit der Lupe zu lesen.

Pica stellt eine Maßeinheit dar, die deshalb absolut ist, weil sie an die Einheit Point gebunden ist. Ein Pica ist immer 12 Point groß.

Da Point und Pica im Gegensatz zu Pixel absolut sind, ist es sehr gefährlich die Einheiten Pica und Point innerhalb einer Seite mit Pixel zu mischen.

Die Einheiten em und ex

Die Einheiten em und ex sind wirklich relativ, das heißt sie können vom Benutzer beeinflusst werden. Die Einheit em bezeichnet die Größe abhängig von der Standardschriftgröße des Browsers. 1em stellt die unveränderte Standardschriftgröße dar, 2em ist die doppelte Größe. Die Standardschriftgröße kann der Benutzer in seinem Browser festlegen, sofern er einen dazu geeigneten Browser verwendet.

Die Einheit ex bezieht sich auf die Höhe des Buchstaben »x«. Sie kann also abhängig von der gewählten Schrift und deren Größe unterschiedlich ausfallen, auch wenn verschiedene Schriftarten in der gleichen Pixel-Größe dargestellt werden.

!! STOP

Die Einheit ex *wird derzeit noch von keinem der großen Browser unterstützt und sollte daher noch nicht eingesetzt werden.*

Bei Schriftgrößen in em bezieht sich der Wert immer auf die aktuell für das Element gültige Schriftgröße. Bei 0.9em würde die Schrift als 90% der Größe haben, die sie hätte, wenn Sie die Formatierung nicht angeben. Folgendes Beispiel soll das verdeutlichen.

CD

Sie finden das Beispiel als BSP/K02/masseinheit_em.html *auf der Buch-CD.*

Listing 2.18: Beispiel zur Verwendung der Maßeinheit em

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01//EN">

<html>
    <head>
        <title>Maßeinheiten: em</title>
        <style type="text/css">
        <!--
            html { font-family:sans-serif; font-size:12px }
            h1   { font-size:2em }
            span { font-size:0.9em; color:gray }
        -->
        </style>
    </head>

    <body>
        <h1>Beispiele zur Einheit <span>em</span></h1>
        <p>Wenn der Browser die Einheit
            <span>em</span> korrekt anwendet,
            sollte die Überschrift 24 Pixel hoch
            sein, dieser Text halb so groß. Die
            innerhalb der <span>span</span>-
            Elemente enthaltenen Texte, werden grau
            dargestellt und haben 90% der Größe des
            umgebenden Textes.</p>
    </body>
</html>
```

In diesem Beispiel wird die Standardschriftgröße für alle Elemente der Seite zunächst auf 12 Pixel festgelegt. Für Überschriften der Ebene 1 wird dann eine Größe von `2em` definiert. Das bedeutet, die Überschrift wird mit einer Größe von 24 Pixel dargestellt. Da für `span`-Elemente eine Größe von `0,9em` definiert wird, heißt das, dass `span`-Elemente innerhalb einer Überschrift mit 21,6 Pixel und innerhalb eines normalen Absatzes mit 10,8 Pixel angezeigt werden.

Natürlich stellen die Browser keine Schriftgrößen in Dezimalzahlen dar, sondern runden die so berechneten Werte.

INFO

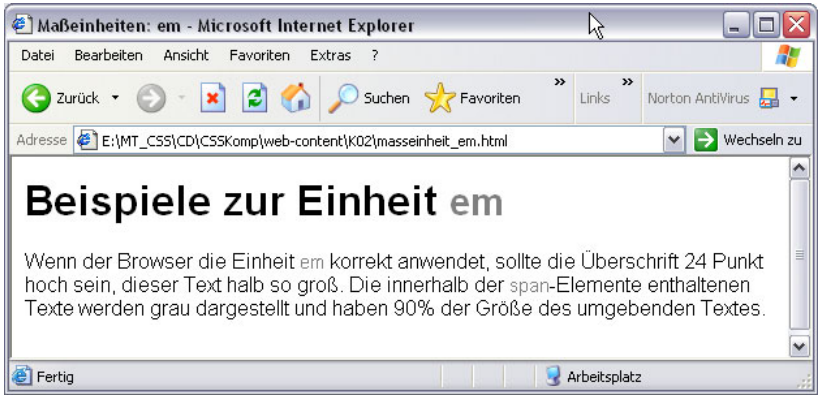


Abbildung 2.14: Die korrekte Darstellung des Beispiels im Internet Explorer

Der Netscape Navigator 4.x kann mit der Einheit `em` nichts anfangen. Er ignoriert daher die Größenangabe für die Schrift. Der Internet Explorer 7 (BETA) zeigt die Schriften allgemein zu klein an. Zwar hat die Überschrift die doppelte Größe wie der normale Text, aber weder hält der Text die Größe von 12px noch die Überschrift die Größe von 24px ein.

Absolute Längeneinheiten sind nur dann sinnvoll, wenn die physischen Eigenschaften des Ausgabemediums bekannt sind.

:-) TIPP

Prozentwerte

Prozentwerte werden durch eine Zahl unmittelbar gefolgt von einem Prozentzeichen angegeben. Vor der Zahl können Sie ebenfalls ein Vorzeichen angeben. Prozentwerte sind immer relativ zu einem anderen Wert, wie einer Länge oder Schriftgröße.

Auf welchen Wert sich der Prozentwert bezieht, bestimmt die Eigenschaft, für die Sie den Prozentwert angeben.

2.4 Farben

Müssen Sie Farbwerte angeben, beispielsweise als Hintergrund- oder Vordergrundfarbe (`background-color` und `color`), gibt es dafür zig verschiedene Möglichkeiten. Sie können RGB-Werte und Farbnamen verwenden, wobei es wiederum verschiedene Möglichkeiten zur Definition der RGB-Werte gibt.

Sie finden das Beispiel auf der Buch-CD in der Datei BSP/K02/farben.html.

Farbnamen

HTML kennt 16 benannte Farben, die in HTML 4.0 definiert sind. Darüber hinaus gibt es Farbnamen in Abhängigkeit vom Betriebssystem. Auf diese Weise können Sie beispielsweise Webseiten an das Farbschema des Betriebssystems anpassen.

Tabelle 2.1: Farbnamen zur Verwendung als Farbangaben

HTML-Farbname	Farbe
`aqua`	aquamarinblau
`black`	schwarz
`blue`	blau
`fuchsia`	fuchsia
`gray`	grau
`green`	grün
`lime`	limonengelb
`maroon`	kastanienbraun
`navy`	navygrün
`olive`	olivgrün
`purple`	violett
`red`	rot
`silver`	silber
`teal`	blau-grün (teal)
`white`	weiß
`yellow`	gelb

RGB-Farben

RGB-Farben werden im SRGB-Farbraum dargestellt. RGB ist dabei die Abkürzung für Rot-Grün-Blau. Das sind die drei Farbbestandteile, aus denen die Farben zusammen gesetzt werden.

rgb-Anweisung

Eine Möglichkeit, eine RGB-Farbe anzugeben, ist die RGB-Anweisung. Ihr übergeben Sie nacheinander und durch Kommata getrennt die Farbwerte für Rot, Grün und Blau. Jeder Farbwert kann dabei eine positive ganze Zahl von 0 bis 255 (einschließlich) sein. Je höher der Wert, desto größer der entsprechende Farbanteil. Die Angabe `rgb(255,0,0)` definiert damit ein reines Rot, da der Anteil von Grün und Blau bei 0 liegt. Die Angabe `rgb(0,0,0)` stellt Schwarz und `rgb(255,255,255)` Weiß dar. Alle Grautöne haben für alle drei Farbanteile immer die gleichen Wert, je höher, desto heller das Grau.

Anstelle der absoluten Farbwerte sind jedoch auch Prozentwerte erlaubt. Auch `rgb(100%,0%,0%)` würde also die Farbe Rot definieren.

Hexadezimale Angabe

Alternativ können Sie die RGB-Werte mit hexadezimalen Farbangaben definieren. Solche Farbwerte werden durch das Zeichen # eingeleitet. Danach folgen nacheinander die Hexwerte für die einzelnen Farbanteile. Generell gibt es dazu eine drei- und eine sechsstellige Notation. Die dreistellige können Sie nur dann anwenden, wenn für jeden Farbwert zwei gleiche Zeichen (0-9 und A-F) verwendet werden. Die Angabe `#FF0000` (für Rot) können Sie also abkürzen, indem Sie von jedem Zeichenpaar ein Zeichen streichen. Die Angabe wäre als gleichbedeutend mit `#F00`. Der Farbewert für Weiß `#FFFFFF` könnte also mit `#FFF` abgekürzt werden.

Groß- und Kleinschreibung spielt bei der hexadezimalen Angabe keine Rolle. Sie könnten ebenso `#f00` schreiben.

Beispiel

Folgendes Beispiel zeigt die verschiedenen Möglichkeiten, den Text des Absatzes in Rot (bzw. Grün (zweiter Absatz) und Blau (dritter Absatz) darzustellen.

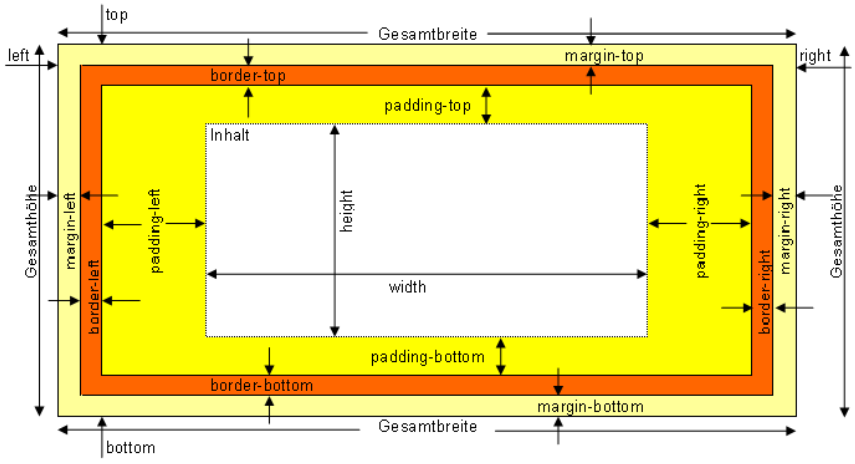
Listing 2.19: Farben angeben

```
<p style="color:rgb(255,0,0)">ROT: Farbe angegeben mit rgb(255,0,0).</p>
<p style="color:rgb(0,255,0)">GRÜN: Farbe angegeben mit rgb(0,255,0).</p>
<p style="color:rgb(0,0,255)">BLAU: Farbe angegeben mit rgb(0,0,255).</p>
<p style="color:rgb(100%,0%,0%)">
    ROT: Farbe angegeben mit rgb(100%,0%,0%).</p>
<p style="color:red">ROT: Farbe angegeben mit Farbname.</p>
<p style="color:#FF0000">ROT: Farbe angegeben mit #FF0000.</p>
<p style="color:#F00">ROT: Farbe angegeben mit #F00.</p>
```

2.5 Das Boxmodell

Das Boxmodell von CSS gehört mit zu dem wichtigsten Eigenschaften des CSS-Standards. Es regelt, wie die Elemente des Dokuments in rechteckiger Form dargestellt werden. Grafisch lässt sich das wie folgt darstellen.

Abbildung 2.15: Das Boxmodell, grafisch dargestellt



Der weiße Bereich in der Mitte, der mit Inhalt gekennzeichnet ist, enthält dann den eigentlichen Inhalt des Elements, beispielsweise den Text eines mit dem `p`-Element definierten Absatzes. Geben Sie für ein Element die Breite oder Höhe mit den CSS-Eigenschaften `width` und `height` an, bezeichnet dies die Größe des Element-Inhalts. Hinzu kommen dann der Innenabstand (`padding`), der Rahmen (`border`) und der Außenabstand (`margin`). Alle zusammen ergeben die Gesamtbreite bzw. Gesamthöhe. Die Eigenschaften `left`, `top`, `bottom` und `right` bestimmen die Position des Elements.

Und genau hier liegt das Problem des Internet Explorers. Beim Internet Explorer umfasst der Wert der Eigenschaft `width` nämlich auch die Abstände und Ränder. Für dieses Problem gibt es eine Lösung, den so genannten Boxmodell-Hack, der aber auch nicht ganz unproblematisch ist.

Mehr dazu erfahren Sie in Kapitel 3, »Browseroptimierung«.

In CSS gibt es im Wesentlichen zwei Typen von Elementen, Blockelemente und Inline-Elemente. Letztere erzeugen keinen eigenen Absatz, sondern sind Bestandteil des Fließtextes. Dazu gehören beispielsweise die Elemente `span`, `a`, `b`, und `i`.

Genau anders verhalten sich die Blockelemente (auch Block-Level-Elemente genannt). Sie erzeugen immer einen neuen Absatz, beginnen also immer in einer neuen Zeile. Zu diesen Elementen gehören beispielsweise Absätze und Überschriften. Diese Blockelemente können jedoch untergeordnete Inline-Elemente haben. Wenn ein Absatz beispielsweise aus mehreren Zeilen besteht, bildet jede Zeile ein Inline-Zeilenelement, das allerdings vom Browser durch entsprechende Zeilenumbrüche selbst erzeugt wird. Solche Inline-Zeilenelemente stellen Inline-Boxen dar, die wie Blockelemente auch über Abstände, Rahmen etc. verfügen.

Wichtig wird das beispielsweise, wenn Sie mit der vertikalen Ausrichtung (`vertical-align`) oder mit der Zeilenhöhe (`line-height`) arbeiten.

Etwas aus dem Rahmen fallen die Listen-Elemente (`li`). Sie sind zwar eigentlich Block-Level-Elemente, aber abhängig von den Formatierungen können die Aufzählungszeichen außerhalb der durch das Boxmodell definierten Fläche des Elements stehen.

Weitere Ausführungen zum Boxmodell und dem visuellen Formatierungsmodell von CSS finden Sie im Referenzteil in Kapitel 5, »Größen, Abstände und Positionierung«.

3 Browseroptimierung

Im vorherigen Kapitel haben Sie schon ansatzweise erfahren, wie unterschiedlich sich die Browser trotz des eigentlich klaren und eindeutigen CSS-Standards verhalten. Für Sie als Webentwickler entsteht dadurch das Problem, dass es sehr schwer wird, ansprechend formatierte Webseiten zu gestalten, die in allen großen Browsern gut aussehen und gleichzeitig auch in älteren Browsern noch funktionieren.

Dennoch ist natürlich auch dies möglich. Wichtig ist dazu nur, dass Sie alle Möglichkeiten kennen, mit denen Sie Stylesheets in eine (X)HTML-Seite integrieren können. Dafür gibt es nämlich mehr Möglichkeiten, als bisher verwendet wurden. Diese sollen deshalb zunächst ausführlich beschrieben werden, bevor es dann um die kleinen Tricks und Kniffe geht, mit denen Sie bestimmte Stile vor einem Browser verbergen oder nur für einen ganz bestimmten Browser definieren können.

3.1 Stylesheets in HTML-Seiten integrieren

Als Stylesheet wird die Gesamtheit aller CSS-Stile bezeichnet, die für ein HTML-Dokument gelten. Generell gibt es drei verschiedene Möglichkeiten, diese Stile zu definieren.

Entweder verwenden Sie sie als CSS-Regeln im `style`-Attribut der zu formatierenden HTML-Elemente, integrieren sie als Stile im `style`-Element oder verknüpfen eine CSS-Datei mit Hilfe des `style`- oder `link`-Elements.

Das style-Attribut

Das `style`-Attribut ist die einfachste Möglichkeit, ein einzelnes Element der Webseite zu formatieren.

Sie finden das Beispiel als Datei BSP/K03/styleattribut.html *auf der Buch-CD.*

Sie ergänzen das zu formatierende Element einfach um ein `style`-Attribut und geben als Wert die CSS-Formatierungen so an, wie Sie sie auch im Deklarationsteil eines Stils aufführen würden.

HTML/CSS-Element: `style`-Attribut	*Beschreibung:* CSS-Formatierungen im `style`-Attribut.

Palm	NN	Mozilla		Internet Explorer							Opera			Safari		iCab	Kq	FF
2.x	4.x	1.x	1.8	4.x	5.0x (Win)	5.0 (Mac)	5.1+ (Mac)	5.5 (Win)	6	7	6	7	8	1.x	2.0	2.9+	3.x	1.0
●	⊙	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●

Der Netscape Navigator 4.x- unterstützt prinzipiell die Verwendung des `style`-Attributs. Probleme bereiten ihm jedoch `style`-Attribute in Auflistungen. Hier werden nicht alle Formatierungen korrekt angewendet, die als Stile durchaus ausgeführt werden.

Listing 3.1: Zuweisen der Formatierungen über das style-Attribut

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01//EN">
<html>
    <head>
        <meta http-equiv="content-type"
        content="text/html;charset=iso-8859-1">
        <title>Beispiel für das Style-Attribut</title>
    </head>

    <body>
        <h1>Beispiel zum Style-Attribut</h1>
        <p>Style-Attribute k&ouml;nnen Sie fast
        allen HTML-Elementen zuordnen. F&uuml;hrt
        der Browser die damit definierten CSS-
        Formatierungen aus, sollten Teile dieses
        Textes in fetter Schrift erscheinen. Sie
        befinden sich dann in <span style="font-
        weight:bold">span</span>-Elementen mit
        einem <span style="font-weight:bold"
        >style</span>-Attribut. </p>
    </body>
</html>
```

INFO

Generell haben Formatierungen im `style`*-Attribut Vorrang vor allen anderen CSS-Stilen im Dokument. Diese Formatierungen haben allerdings den Nachteil, dass Sie womöglich immer wieder die gleichen Formatierungen definieren müssen. Falls Sie sie einmal ändern möchten, macht das eine Menge Arbeit. Daher sollten Sie die* `style`*-Attribute nur als Notlösung ansehen. Besser ist es, Stile zu verwenden, die im Kopf der Seite definiert und/oder eingebunden werden.*

Das style-Element im head-Bereich

Das `style`-Element, das Sie im `head`-Element der Seite einfügen können, ermöglicht Ihnen, sowohl externe CSS-Dateien zu verknüpfen als auch Stile zu definieren. Die hier definierten Stile können Sie allerdings nur innerhalb der Seite verwenden; Sie müssten sie bei Bedarf also in jede Webseite einfügen und dann natürlich auch in jeder Webseite ändern, falls Änderungen notwendig sind.

Externe CSS-Dateien, die Sie nur verknüpfen, können Sie hingegen in mehreren Dateien nutzen und können die hier definierten Stile an zentraler Stelle verwalten. Dennoch macht es in einigen Fällen Sinn, Stile im `style`-Element zu definieren.

Sie finden das Beispiel als Datei BSP/K03/styleelement.html *auf der Buch-CD.*

Syntax und Attribute des style-Elements

Um gültigen HTML-Code zu erzeugen, müssen Sie für das `style`-Element mindestens das Attribut `type` mit dem Wert `text/css` angeben. Der Wert legt fest, dass es sich bei dem Inhalt des Elements um CSS-Code handelt.

HTML/CSS-Element: `style`-Element	*Beschreibung:* CSS-Formatierungen im `style`-Element.

Palm	NN	Mozilla		Internet Explorer							Opera			Safari		iCab	Kq	FF
2.x	4.x	1.x	1.8	4.x	5.0x (Win)	5.0 (Mac)	5.1+ (Mac)	5.5 (Win)	6	7	6	7	8	1.x	2.0	2.9+	3.x	1.0
●	⊙ [1]	●	●	●	●	●	●	●	●	●	●	●	●	●	●	⊙	●	●

[1] Stile werden nicht angewendet, wenn Sie für das `style`-Element einen Wert ungleich `media="screen"` angeben.

Ein gültiges `style`-Element könnte somit wie folgt aussehen:

Listing 3.2: Gültiges style-Element

```
<head>
    <meta http-equiv="content-type"
        content="text/html;charset=iso-8859-1">
    <title>Beispiel für das Style-Element</title>
    <style type="text/css">
    </style>
</head>
```

Kommentare

Besser ist aber, Sie fügen innerhalb des Elements noch HTML-Kommentaranfangs- (`<!--`) und -endzeichen (`-->`) ein. Das verhindert, dass Browser, die kein CSS können, das Element als Text der Seite darstellen.

Listing 3.3: Kommentare innerhalb des style-Elements

```
<style type="text/css">
<!--
    /* hier folgt der CSS-Code */
    body { color:red }
-->
</style>
```

Innerhalb des `style`-Elements können Sie auch CSS-Kommentare einfügen. Diese beginnen mit `/*` und enden mit `*/`. Dazwischen können Sie beliebigen Text, beispielsweise zur Erläuterung des Codes, einfügen.

Das media-Attribut

Mit dem optionalen `media`-Attribut können Sie das Ausgabemedium bestimmen, für das die Stile gelten sollen. So können Sie für die Druckausgabe ganz gezielt andere Stile erstellen als für die Ausgabe am Bildschirm oder per Screenreader. Sie geben dazu einfach das passende Ausgabemedium an.

Welche Ausgabemedien es gibt und welche davon in der Praxis bereits eingesetzt werden können, wird etwas weiter unten im Abschnitt »Ausgabemedien im Detail« beschrieben.

Lassen Sie das Attribut weg, wird der Standardwert `all` verwendet. Das heißt, die definierten Stile gelten für alle Ausgabemedien. Alternativ kommen die häufig verwendeten Werte `screen` für die Bildschirmausgabe und `print` für den Druck in Frage.

Geben Sie für das `media`-Attribut den Wert `all` an, ignoriert der Netscape Navigator 4.x- den Inhalt des `style`-Elements. Sie sollten daher explizit `screen` angeben, wenn die Stile ohnehin nur Formatierungen für die Bildschirmausgabe enthalten. Andere Medientypen als `screen` unterstützt der Navigator nicht.

iCab 2.9.x stellt Stylesheets, die mit `media="print"` definiert wurden, auch bei der Bildschirmausgabe dar – zumindest dann, wenn kein Stylesheet explizit für das Medium `screen` definiert wurde.

Selbstverständlich kann eine Seite auch mehrere `style`-Elemente enthalten, wie das folgende Beispiel zeigt. Es definiert zunächst die Stile für die Bildschirmausgabe und anschließend die Stile für den Druck.

In dem Stylesheet für die Druckausgabe wird der Bereich mit der ID `navigation` ausgeblendet, indem die `visibility`-Eigenschaft auf `hidden` gesetzt wird. Außerdem wird die Hintergrundfarbe mit `background-color` auf Weiß gesetzt und die Schriftfarbe auf Schwarz.

Für die Bildschirmausgabe wird die Schriftfarbe auf Rot gesetzt und der Navigationsbereich eingeblendet und mit einem Rahmen versehen.

Listing 3.4: Unterschiedliche Stile abhängig vom Medium

```
<html>
    <head>
        <meta http-equiv="content-type"
        content="text/html;charset=iso-8859-1">
        <title>Beispiel für das Style-Element</title>
        <style type="text/css" media="screen">
        <!--
            /* hier folgt der CSS-Code */
            body        { color:red }
            #navigation { visibility:visible;
                          border:1px solid black }
        -->
        </style>
        <style type="text/css" media="print">
        <!--
            /* hier folgt der CSS-Code für die Druckausgabe*/
            body        { color:black; background-color:white }
            #navigation { visibility:hidden }
        -->
        </style>
    </head>

    <body>
        <div id="navigation">- Hier könnte die
        Navigationsleiste stehen -</div>
        <h1>Beispiel zum Style-Element</h1>
        <p>Falls die Schrift nicht rot ist,
        beherrscht der Browser das media-Attribut
        des style-Elements nicht richtig.</p>
    </body>
</html>
```

Wenn Sie das Dokument in einem Browser mit Druckvorschau öffnen, beispielsweise im Opera-Browser, sehen Sie, dass das Stylesheet für die Druckausgabe korrekt angewendet wird. Der Navigationsbereich ist nicht sichtbar und die Schriftfarbe schwarz.

Bei der normalen Anzeige im Browser ist dagegen der Navigationsbereich sichtbar.

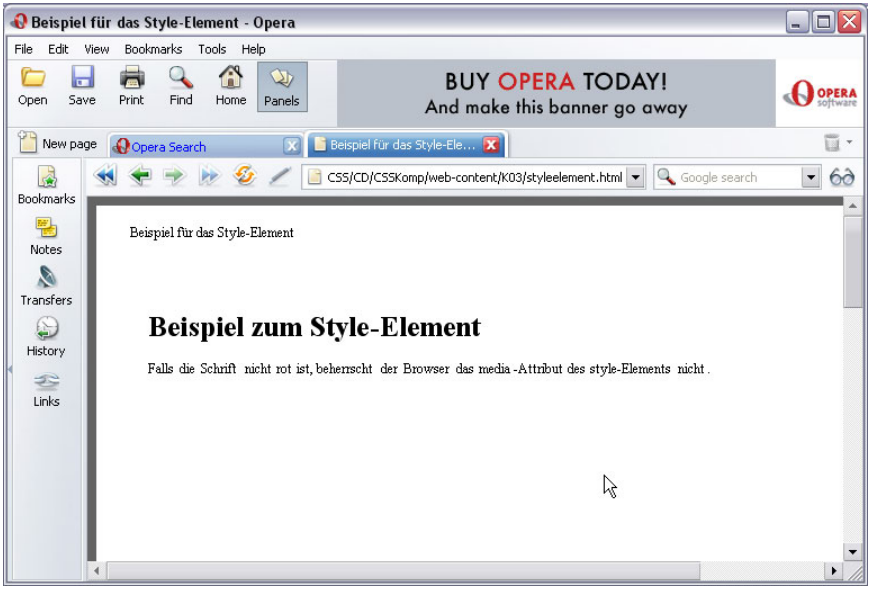


Abbildung 3.1: Die Druckvorschau zeigt die Formatierungen für das Medium print an.

:-) TIPP

Immer wenn Sie ein spezielles Stylesheet für die Druckausgabe erstellen, sollten Sie die Bereiche der Webseite ausblenden, die für den Ausdruck keine Rolle spielen, wie Navigationsleisten, Links zur Navigation innerhalb der Seite etc.

Abbildung 3.2: Bei der normalen Anzeige im Browser wird hingegen das Stylesheet mit der Angabe media="screen" verwendet.

REF

Alternativ können Sie auch die `@media`-Regel verwenden, um das Ausgabemedium zu bestimmen. Mehr dazu erfahren Sie im folgenden Abschnitt »@-Regeln«.

@-Regeln

Innerhalb eines `style`-Elements können Sie @-Regeln einsetzen. Sie dienen ganz verschiedenen Zwecken. In CSS gibt es derzeit folgende @-Regeln:

- `@import`, zum Importieren von CSS-Dateien,
- `@media`, um den Medientyp zu definieren, für den die Stile gelten sollen,
- `@font-face`, um Schriftartdateien zu importieren (siehe *Kapitel 4, »Textformatierungen«*),
- `@page`, um das Seitenlayout für seitenbasierte Medien zu definieren (siehe *Kapitel 10, »Spezielle Medien: Druck- und Sprachausgabe«*).

Am ehesten benötigen Sie noch die beiden Regeln `@import` und `@media`. Mit der letztgenannten Regel haben Sie die Möglichkeit, innerhalb eines `style`-Elements CSS-Regeln für mehrere Medientypen zu erstellen. Dazu verwenden Sie folgende Syntax:

```
@media Medientyp1, Medientyp2, ... {...}
```

Der Netscape Navigator 4.x- stellt die mit `@media` definierten Stile nicht dar, unabhängig vom Medientyp. Gleiches gilt für iCab 2.9.x und den Internet Explorer für Mac. Die Windows-Version stellt die `@media`-Regel ab der Version 6.0 dar.

Sie finden das Beispiel als Datei BSP/K03/styleelement2.html *auf der Buch-CD.*

HTML/CSS-Element: `@media`-Regel	*Beschreibung:* Ausgabemedien im `style`-Element mit der `@media`-Regel angeben.

Palm	NN	Mozilla		Internet Explorer							Opera			Safari		iCab	Kq	FF
2.x	4.x	1.x	1.8	4.x	5.0x (Win)	5.0 (Mac)	5.1+ (Mac)	5.5 (Win)	6	7	6	7	8	1.x	2.0	2.9	3.x	1.0
●			●						●	●		●	●		●	●[1]	●	●

[1] ab iCab 3.0

Innerhalb der geschweiften Klammern definieren Sie die Stile. Im einfachsten Fall definieren Sie einfach für jeden Medientyp alle Stile einmal.

Listing 3.5: Verwendung der @media-Regel

```
<style type="text/css" >
<!--
    /* hier folgt der CSS-Code für verschiedene Medien */
    @media print {
            body {
                color:red;
                background-color:white;
                font-family:sans-serif
            }
            h1 {
                color:red;
                font-weight:bold;
                text-decoration:underline;
                font-family:sans-serif
            }
    }
    @media screen {
            body {
                color:red;
                background-color:white;
                font-family:sans-serif
            }
            h1 {
                color:red;
                font-weight:bold;
                text-decoration:underline;
                font-family:sans-serif
            }
    }
-->
</style>
```

Ein solches Stylesheet können Sie aber auch noch einfacher gestalten, indem Sie die identischen Stile einfach zusammenfassen. Dazu geben Sie beide Medientypen durch Komma getrennt nach dem Schlüsselwort `@media` an.

Listing 3.6: Medien in der @media-Regel gruppieren

```
<style type="text/css" >
<!--
    /* hier folgt der CSS-Code für verschiedene Medien */
        @media print, screen, handheld {
            body {
                color:red;
                background-color:white;
                font-family:sans-serif}
            }
        @media screen {
            h1 {
                color:red;
                font-weight:bold;
                text-decoration:underline;
                font-family:sans-serif}
            }
        }
```

```
        #navigation { border:1px solid red }
-->
</style>
```

Hier wurde in der einen `@media`*-Regel nur der* `body`*-Element-Stil definiert und in der anderen nur der Stil für die Überschriften der ersten Ebene. Damit können Sie dann auch im Browser prüfen, ob vielleicht nur die Kombination verschiedener Medientypen nicht unterstützt wird. Der dazu notwendige HTML-Code müsste dann wie folgt aussehen:*

:-) TIPP

Listing 3.7: Notwendiger HTML-Code zum Testen der @media-Regel

```
<body>
    <div id="navigation">- Hier könnte die
    Navigationsleiste stehen -</div>
    <h1>Beispiel zum Style-Element</h1>
    <p>Falls die Schrift nicht rot ist,
    unterstützt der Browser die @media-Regel mit
    kombinierten Ausgabemedien nicht.</p>
    <p>Falls die Überschrift nicht unterstrichen
    ist, unterstützt der Browser die @media-
    Regel gar nicht.</p>
</body>
```

Die `@import`-Regel können Sie verwenden, um eine CSS-Datei in ein Stylesheet zu importieren. Die dort enthaltenen Stile werden dann hinsichtlich der Kaskade und der Rangfolge genauso behandelt, als wären sie innerhalb des `style`-Elements definiert. Dies gilt auch für relative Pfade. Ist in der CSS-Datei ein relativer Pfad, beispielsweise zu einem Hintergrundbild definiert, werden diese relativ zu dem Dokument interpretiert, in dem sich die `@import`-Regel befindet.

Das Beispiel befindet sich in den Dateien BSP/K03/verknuepfung.html, BSP/K03/formate.css, BSP/K03/formate3.css, BSP/K03/formate2.css *und* BSP/K03/formate1.css.

CD

Die `@import`-Regel hat folgende Syntax:

```
@import "Dateiname"; oder

@import url("Dateiname");
```

In beiden Fällen ersetzen Sie `Dateiname` durch den Namen (eventuell mit Pfad) der zu importierenden CSS-Datei.

Die `@import`*-Regel muss immer zwingend am Anfang eines* `style`*-Elements stehen. Erst danach dürfen weitere Stildefinitionen folgen. Außerdem müssen Sie sie mit einem Semikolon abschließen. Vergessen Sie das, führen die Browser die Regel nicht aus und der Internet Explorer für Macintosh kann sogar abstürzen.*

!! STOP

Ob Sie nur den Dateinamen angeben oder ihn in `url()` einfassen, spielt gemäß CSS-Standard keine Rolle. Für die Browser macht es aber schon einen Unterschied. Nicht jeder Browser akzeptiert beide Schreibweisen.

HTML/CSS-Element: `@import`-Regel ohne `url()`

Beschreibung: Import von CSS-Dateien über `@import` im `style`-Element.

Palm	NN	Mozilla		Internet Explorer							Opera			Safari		iCab	Kq	FF
2.x	4.x	1.x	1.8	4.x	5.0x (Win)	5.0 (Mac)	5.1+ (Mac)	5.5 (Win)	6	7	6	7	8	1.x	2.0	2.9+	3.x	1.0
●		●	●		●	●	●	●	●	●	●	●	●	●	●	●	●	●

HTML/CSS-Element: `@import`-Regel mit `url()`

Beschreibung: Import von CSS-Dateien über `@import` im `style`-Element.

Palm	NN	Mozilla		Internet Explorer							Opera			Safari		iCab	Kq	FF
2.x	4.x	1.x	1.8	4.x	5.0x (Win)	5.0 (Mac)	5.1+ (Mac)	5.5 (Win)	6	7	6	7	8	1.x	2.0+	2.9	3.x	1.0
●		●	●		●	●	●	●	●	●	●	●	●	●	●	●	●	●

Der Netscape Navigator 4.x unterstützt die `@import`-Regel nicht. Sie können damit also sehr schön CSS-Dateien laden, die Sie vor dem Netscape Navigator verbergen möchten.

Zusätzlich können Sie auch bei der `@import`-Regel den Medientyp angeben, für den das importierte Stylesheet gelten soll. Dazu geben Sie nach dem URL die Medientypen an, beispielsweise so:

```
@import "formate3.css" screen, print;
```

HTML/CSS-Element: `@import`-Regel ohne `url()` und mit Medientyp

Beschreibung: Import von CSS-Dateien über `@import` im `style`-Element.

Palm	NN	Mozilla		Internet Explorer							Opera			Safari		iCab	Kq	FF
2.x	4.x	1.x	1.8	4.x	5.0x (Win)	5.0 (Mac)	5.1+ (Mac)	5.5 (Win)	6	7	6	7	8	1.x	2.0	2.9	3.x	1.0
●		●	●								●	●	●	●	●	●	●	●

Der Internet Explorer unterstützt die `@import`-Regel mit Angabe des Medientyps weder auf dem Mac noch unter Windows. Falls Sie das Semikolon hinter der Regel vergessen, kann dies bei der Mac-Version sogar zu einem kompletten Systemabsturz führen.

CSS-Dateien verknüpfen

Alternativ zur `@import`-Regel gibt es eine zweite Möglichkeit, CSS-Dateien mit einem Dokument zu verknüpfen: das `link`-Element. Es wird wie das `style`-Element im `head`-Bereich der Seite definiert, hat aber eine gänzlich andere Syntax. Das liegt daran, dass die Verknüpfung von CSS-Dateien nur eine von mehreren Aufgaben des `link`-Elements ist. Da nur dies aber einer Rolle für CSS spielt, werden die anderen Aufgaben des `link`-Elements hier nicht behandelt.

Die Syntax zur Einbindung von CSS-Dateien lautet:

```
<link rel="stylesheet" href="Dateiname">
```

wobei Sie natürlich `Dateiname` durch den Namen (gegebenenfalls mit Pfad) der CSS-Datei ersetzen.

Das `link`-Element hat keinen schließenden Tag. Wenn Sie einen angeben, erzeugen Sie damit invaliden Code.

!! STOP

HTML/CSS-Element: `link`-Element ohne Medientyp, mit und ohne `type`-Attribut	*Beschreibung:* Verknüpfen von CSS-Dateien mit einem Dokument.

Palm	NN	Mozilla		Internet Explorer							Opera			Safari		iCab	Kq	FF
2.x	4.x	1.x	1.8	4.x	5.0x (Win)	5.0 (Mac)	5.1+ (Mac)	5.5 (Win)	6	7	6	7	8	1.x	2.0	2.9	3.x	1.0
●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●

Optional verfügt das `link`-Element über das `type`-Attribut. Damit können Sie den MIME-Typ der verknüpften Datei festlegen. Hier geben Sie wie beim `style`-Element `text/css` an.

Alternative CSS-Dateien

Gemäß CSS-Standard können Sie als Autor einer Seite nicht nur eine CSS-Datei definieren, sondern auch Alternativen, die der Benutzer dann wählen kann. Generell gibt es drei Typen von Stylesheets: das Standard-Stylesheet, das immer geladen wird; bevorzugte Stylesheets, die nur geladen werden, wenn der Benutzer kein Stylesheet gewählt hat; und alternative Stylesheets, die nur nach Auswahl durch den Benutzer geladen werden. Alle diese Stylesheets definieren Sie über das `link`-Element, indem Sie bestimmte Werte für das Attribut `rel` und das optionale Attribut `title` definieren. Folgende Tabelle zeigt, welche Kombination von Werten für `title` und `rel` welchen Typ Stylesheet definieren.

Tabelle 3.1: Definieren von alternativen Stylesheets

Typ	**Attribute**	
	`title`	`rel`
Standard	-	stylesheet
Bevorzugt	beliebiger Text	stylesheet
Alternativ	beliebiger Text	alternate stylesheet

Geben Sie also kein `title`-Attribut an, handelt es sich um das Standard-Stylesheet, das von allen Browsern unterstützt wird. Sobald Sie ein `title`-Attribut festlegen, wird die CSS-Datei zum bevorzugten Stylesheet. Auf die Darstellung im Browser hat das keine Auswirkungen. Mit dem folgenden Code definieren Sie ein bevorzugtes und ein alternatives Stylesheet.

Listing 3.8: Verknüpfen von CSS-Dateien mit dem link-Element

```
<link rel="stylesheet" href="formate.css" type="text/css"
      title="default">
<link rel="alternate stylesheet" href="formateAlt.css" type="text/css"
      title="Alternative">
```

HTML/CSS-Element: Alternative und Default-Stylesheets	*Beschreibung:* Definieren von Default- und alternativen Stylesheets über das `link`-Element.

Palm	NN	Mozilla		Internet Explorer							Opera			Safari		iCab	Kq	FF
2.x	4.x	1.x	1.8	4.x	5.0x (Win)	5.0 (Mac)	5.1+ (Mac)	5.5 (Win)	6	7	6	7	8	1.x	2.0	2.9+	3.x	1.0
		●[2]	●									●	●			○[1]		●

[1] Im Menü wird nur das Default-Stylesheet angezeigt, keine alternativen.
[2] ab Version 1.4

Die Werte für das `title`-Attribut zeigt der Browser zur Auswahl des Stylesheets als Menüeintrag an.

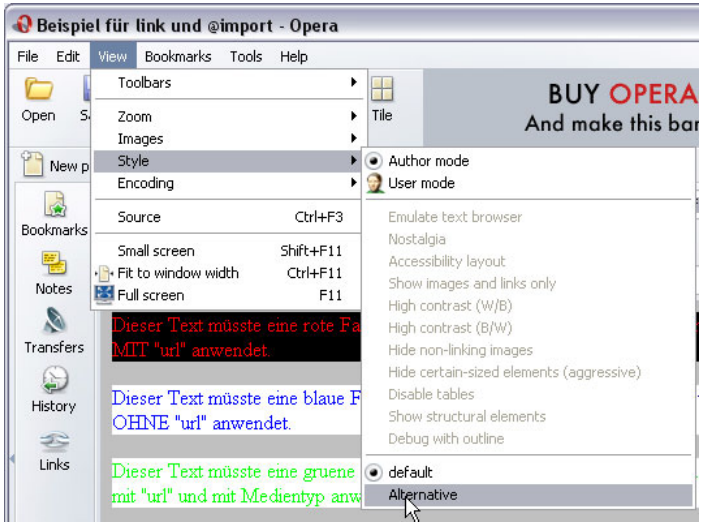


Abbildung 3.3: Auswahl der Stylesheets im Opera-Browser

3.2 Ausgabemedien im Detail

Ab CSS 2.0 ist es möglich, spezielle Stylesheets abhängig vom Ausgabemedium zu erstellen. Damit können Sie ganz gezielt die Formatierungen auf das Ausgabemedium abstimmen.

Das Ausgabemedium können Sie entweder über die `@media`-Regel bestimmen (siehe »*@-Regeln*«) oder über das `media`-Attribut des `style`- (siehe »*Das media-Attribut*«) bzw. `link`-Elements (siehe »*CSS-Dateien verknüpfen*«).

Für alle drei Möglichkeiten nutzen Sie die gleichen Schlüsselwörter, um das Ausgabemedium zu bestimmen:

Tabelle 3.2: Verfügbare Ausgabemedien gemäß CSS 2.0

Schlüsselwort	Ausgabemedium
`all`	alle Medien
`aural`	Sprachausgabe
`braille`	Ausgabegeräte für Blindenschrift, Braillezeilen
`embossed`	Blindenschrift-Drucker
`handheld`	Palmtops, PDAs, Windows-CE-Geräte, teilweise auch Handys

Tabelle 3.2: Verfügbare Ausgabemedien gemäß CSS 2.0 (Forts.)

Schlüsselwort	Ausgabemedium
`print`	Drucker
`projection`	Projektoren und Beamer
`screen`	Bildschirme jeder Art
`tty`	Geräte mit unveränderlichen Zeichentypen, wie Terminals, Handys, Fernschreiber …
`tv`	TV und Multimedia-Geräte

Wenn Sie spezielle Stylesheets für verschiedene Ausgabegeräte definieren möchten, sollten Sie darauf achten, dass Sie auch für alle Medien welche erstellen. Sie sollten daher auf jeden Fall grundlegende Formatierungen mit dem Medientyp `all` oder ohne Medientyp definieren.

Außerdem empfiehlt es sich, die Formatierungen für den Druck für alle Medientypen zu definieren, die für Druckausgaben brauchbar sind, und für die Medientypen der Bildschirmausgaben ebenfalls Mediengruppen zu definieren. Das könnte beispielsweise wie im folgenden Beispiel aussehen. Hier wird zunächst für alle Ausgabemedien (ohne `media`-Attribut) die Schriftfamilie festgelegt.

Achten Sie darauf, auch eine generische Schriftart wie `sans-serif` *anzugeben, da Sie ja nicht wissen können, welche Schriftarten beispielsweise auf PDAs, TV-Geräten etc. zur Verfügung stehen.*

Das nächste `style`-Element definiert dann die Stile, die speziell für die Ausgabe auf Bildschirmen gedacht sind. Sie können innerhalb des `style`-Elements natürlich mit der `@media`-Regel für einzelne Geräte (hier PDAs und Handys) gesonderte Formatierungen festlegen. Im Beispiel wird auf diesen Geräten die Schriftgröße reduziert und auf Blau gesetzt.

Das dritte `style`-Element legt hingegen die Formatierungen für Druckausgaben fest.

Listing 3.9: Verschiedene Ausgabemedien berücksichtigen

```
<style type="text/css">
<!--
    /* grundlegende Formatierungen für
    alle Browser und Ausgabegeräte */
    body, p, td, h1, h2, h3 {
        font-family: Arial, Helvetica, sans-serif
    }
-->
</style>
<style type="text/css" media="screen,
     projection,  tty,  tv,  handheld">
```

```
<!--
    /* grundlegende Formatierungen für Bildschirmausgaben */
    @media handheld, tty {
        /* Besondere Formate für PDAs und Handy */
        body, p, td, h1, h2, h3 {
            font-size:0.9em;
            color:blue;
         }
    }
    body, p, td, h1, h2, h3 {
        color:red
    }
-->
</style>

<style type="text/css" media="print, embossed, braille">
<!--
    /* grundlegende Formatierungen für
    alle Druckausgaben oder vergleichbare
    Ausgabegeräte */
    body, p, td, h1, h2, h3 { color:black }
-->
</style>
```

Wider Erwarten berücksichtigt der Palm Web Browser 2.0.1 nicht die Stile für das Medium `handheld`, sondern verwendet nur die Stile, die für die Medien `all` und `screen` definiert sind. Gleiches gilt auch für den Palm Simulator.

Abbildung 3.4: Die Seite im Palm-Simulator mit roter statt blauer Schrift

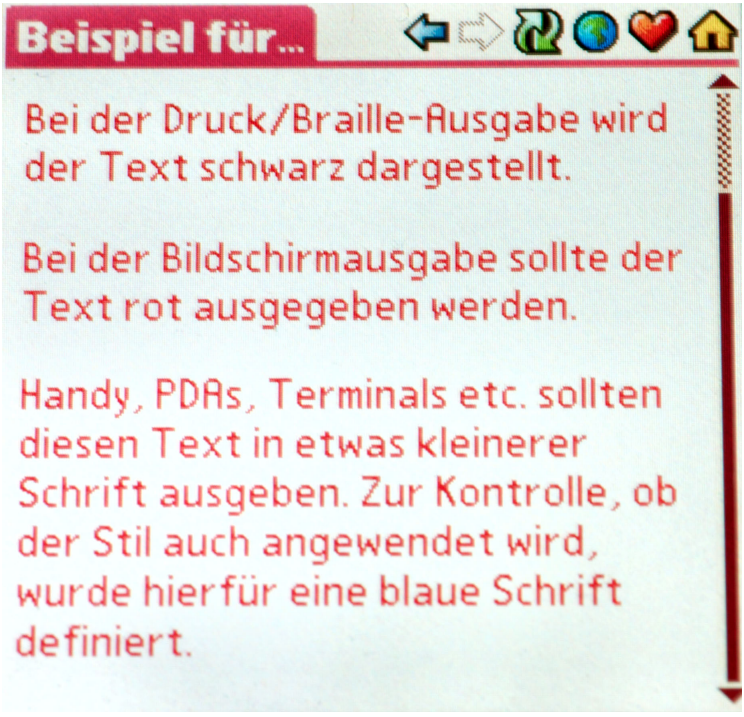


Abbildung 3.5: Der Palm Web Browser stellt nur die Formatierungen für das Medium screen dar.

INFO

Selbstverständlich können Sie obige Lösung auch mit der `@import`*-Regel oder mit Stylesheets verwirklichen, die mit dem* `link`*-Element verknüpft sind.*

3.3 CSS-Browserweichen

Wie die vorstehenden Seiten und Kapitel gezeigt haben, verhalten sich die Browser recht unterschiedlich beim Einbinden von Stilen und bei der Unterstützung von Selektoren. Wenn Sie den Anspruch haben, dass Ihre Webseite in allen Browsern gleich oder sogar nahezu identisch aussehen soll, müssen Sie sich die Mühe machen, Ihren CSS-Code an die Browser anzupassen.

Das funktioniert natürlich nur, wenn Sie es schaffen, bestimmte Browser von der Verarbeitung der Stile auszuschließen oder die Stile ganz gezielt nur für einen bestimmten Browser zu erstellen. Diese Techniken werden Browserweichen oder Browser-Hacks genannt.

All diese Vorschläge können natürlich nur Notlösungen sein, um Fehler oder Unzulänglichkeiten der Browser zu beheben. Sie haben die gleichen Nachteile wie Browserweichen zum Einsatz von Flash oder anderen proprietären Techniken. Sie funktionieren nur solange, wie die Browserhersteller diese Tricks nicht mit neuen Browser-Versionen, Service-Packs und Updates zunichte machen. Ein gutes Beispiel dafür ist iCab. Was in iCab 2.9.x noch nicht funktionierte, ist nun in Version 3.0 integriert, wodurch die Browserweiche nicht mehr funktioniert.

Setzen Sie also komplexe Browserweichen für viele verschiedene Browser und Versionen ein, müssten Sie diese regelmäßig prüfen, nämlich immer dann, wenn eine neue Browserversion oder auch nur ein Update herauskommt. Daher ist der extensive Einsatz von Browserweichen nicht zu empfehlen.

Nicht jede Browserweiche erzeugt zudem validen CSS-Code. Darauf sollten Sie achten, insbesondere wenn Sie Browserweichen aus dem Internet in Ihre Webseiten integrieren.

Da es aber durchaus Probleme gibt, bei denen Sie ohne Browserweichen nicht weiter kommen, sollen nachfolgend einige ausgewählte Browserweichen vorgestellt werden, die validen Code darstellen.

Netscape Navigator

Am wichtigsten sind in der Praxis sicherlich Lösungen für den Netscape Navigator 4.x und früher, da er hinsichtlich der CSS-Darstellung der schlechteste noch verwendete Browser ist. Dafür ist seine Behandlung jedoch auch ganz einfach.

Das Beispiel finden Sie in der Datei BSP/K03/Netscape4.html *auf der Buch-CD. Dazu gehören die CSS-Dateien* ohneNN4.css *und* NN4.css.

Netscape Navigator ausschließen

Wenn Sie vermeiden möchten, dass der Netscape Navigator eine CSS-Datei oder Stile im `style`-Element verarbeitet, gibt es dazu zwei Möglichkeiten. Die erste besteht darin, dass Sie im `link`-Element, mit dem Sie die CSS-Datei verknüpfen, das Attribut `media="all"` angeben. Das verhindert, dass der Netscape Navigator die CSS-Datei lädt, da er nur den Wert `screen` für das `media`-Attribut unterstützt.

Listing 3.10: Eine CSS-Datei vor dem Netscape Navigator verstecken

```
<!-- Stile, die der NN 4 nicht laden soll -->
<!-- erste Möglichkeit -->
<link rel="stylesheet" type="text/css"
    href="ohneNN4.css" media="all">
<!-- zweite Möglichkeit -->
<style type="text/css">
<!--
    @import url("ohneNN4.css");
-->
</style>
```

Die zweite Möglichkeit besteht darin, die `@import`-Regel zu verwenden, um die CSS-Datei zu importieren, da der Netscape Navigator diese ebenfalls nicht beherrscht.

Stile für Netscape Navigator

Wenn Sie verhindern, dass der Netscape Navigator eine bestimmte CSS-Datei lädt, wäre es natürlich auch sinnvoll, Sie würden speziell für den Netscape Navigator eine andere CSS-Datei laden. Das geht so einfach allerdings nicht.

Die einzige Möglichkeit besteht darin, dass Sie die Stile für den Netscape Navigator als Allererstes über das `link`-Element laden. Deren Stile können Sie dann in den weiteren CSS-Dateien, die Sie vor Netscape verbergen, mit den für die anderen Browser nötigen Formatierungen überschreiben, indem Sie dort die gleichen Stile mit anderen Formatierungen definieren.

Erst laden Sie also immer die CSS-Datei für Netscape und danach diejenigen, die Sie vor Netscape verstecken möchten.

Listing 3.11: CSS-Dateien für Netscape Navigator laden

```
<!-- Stile für Netscape Navigator 4.x -->
<link rel="stylesheet" type="text/css"
      href="NN4.css">
<!-- Stile, die der NN 4 nicht laden soll -->
<!-- erste Möglichkeit -->
<link rel="stylesheet" type="text/css" href="ohneNN4.css" media="all">
<!-- zweite Möglichkeit -->
<style type="text/css">
<!--
    @import url("ohneNN4.css");
-->
</style>
```

Einzelne Stile verbergen

Sie können vor dem Netscape Navigator auch einzelne Stile verbergen. Das geht, indem Sie vor dem Stil, den Netscape nicht berücksichtigen soll, einen CSS-Kommentar einfügen, nämlich `/*/*/`.

Dies ist ein gültiger Kommentar, nämlich ein / eingeleitet durch / und abgeschlossen durch */. Netscapes CSS-Parser kommt damit aber nicht zurecht und behandelt dies als öffnendes Kommentarzeichen. Alles, was danach folgt, wird ignoriert, bis Sie einen nächsten Kommentar einfügen, der aber normalen Text zwischen /* und */ enthalten sollte.*

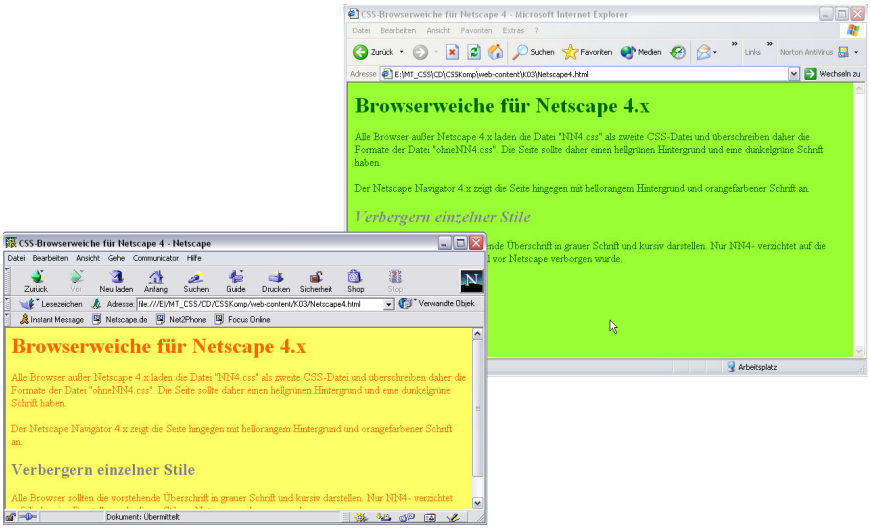


Abbildung 3.6: Darstellung der Beispielseite im Netscape Navigator 4 und im Internet Explorer 6

```
<style type="text/css">
<!--
    /* Einzelner Stil, der vor Netscape verborgen werden soll. */
    /*/*/
    h2 { font-style:italic }
    /* Alle nachfolgenden Stile werden wieder von Netscape beachtet */
    h2 { color:gray }
-->
</style>
```

Listing 3.12: Verbergen einzelner Stile vor dem Netscape Navigator 4

Der Opera-Browser

Für den Opera-Browser gibt es keine solche Browserweiche, mit der Sie ganze CSS-Dateien verstecken oder neu laden können. Sie können maximal bestimmte Stile vor dem Opera-Browser verstecken und auch das funktioniert nur bis zum Opera-Browser 7.3 einschließlich. Basis dieser Browserweiche ist nämlich das Pseudo-Element `:first-child`, das erst ab Opera 7.4x in Teilen unterstützt wird. Die Browserweiche funktioniert allerdings auch in Opera 7.4 noch, da diese Version das Pseudo-Element `:first-child` nicht für das `head`-Element unterstützt.

Sie laden zunächst eine CSS-Datei, die der Opera-Browser verarbeiten soll, oder definieren die Stile in einem `style`-Element. Für die Stile, die Sie vor dem Opera-Browser verbergen möchten, stellen Sie dem Selektor den Selektor `head:first-child+body` voran. Danach geben Sie getrennt durch ein Leerzeichen den Element- oder Klassen-Selektor an. Wenn Sie beispielsweise einen Stil für das `h1`-Element definieren möchten, den Sie vor dem Opera-Browser verbergen möchten, geben Sie `head:first-child+body h1` an. Mit `head:first-child+body` definieren Sie gleich die Formatierung für das `body`-Element.

Listing 3.13: Browserweiche für den Opera-Browser

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01//EN">
<html>
    <head>
        <title>Browserweiche f&uuml;r den Opera-Browser</title>
        <!-- CSS-Datei für Opera -->
        <link href="Opera.css" rel="stylesheet" type="text/css">
        <!-- Stile, die vor dem Opera-Browser vorborgen werden sollen -->
        <style type="text/css">
        <!--
        head:first-child+body {
            color: #060;
            background-color: #9f3
        }
        head:first-child+body h1 {
            color: #060;
            background-color: #9f3;
            text-decoration:underline
        }
         -->
        </style>

    </head>
    <body>
        <h1>Browserweiche f&uuml;r den Opera-Browser</h1>
        <p>Der Opera-Browser liest nur die CSS-Datei
        &quot;Opera.css&quot;, in der bspw. der Stil
        für das body-Element definiert ist. Er legt
        für die Seite einen hellorangen Hintergrund
        mit roter Schrift fest. Alle anderen
        Browser, die das first-child-Pseudo-Element
        unterstützen, verwenden einen
        hellgr&uuml;nen Hintergrund mit gr&uuml;ner
        Schrift. Diesen Stil können Sie, wie hier
        gezeigt, in einem style-Element nach dem
        link-Element definieren, aber auch in die
        gleiche oder eine zweite CSS-Datei setzen.
        Das spielt keine Rolle.</p>
    </body>
</html>
```

Die Sache hat einen großen Nachteil. Auch andere Browser, die wie der Opera-Browser das Pseudo-Element `first-child` *nicht unterstützen, führen die Stile natürlich nicht aus. Dazu gehören iCab 2.9.x, Netscape Navigator 4.x und niedriger, der Internet Explorer 7 und niedriger für Windows und der Internet Explorer 5 und niedriger für Mac. Erst die Mac-Version ab 5.2 führt die Stile aus. Im Prinzip können Sie also auf diese Weise die Stile nur für die folgenden Browser definieren, alle anderen führen sie wie der Opera-Browser 7.x- nicht aus:*

- *Safari 2.x*
- *Konqueror 3.x*
- *Internet Explorer 5.2+ (Mac)*
- *iCab 3.x+*
- *Opera 8+*
- *Mozilla 1.4+*
- *Firefox 1.0+*
- *Netscape 7.1+*
- *Palm Web Browser 2.x+*

Eine richtige Browserweiche für den Opera-Browser ist das also nicht. Um gleiche Formatierungen für den Internet Explorer 5+ für Windows zu erstellen, könnten Sie aber wiederum eine Internet Explorer-Browserweiche nutzen: die bedingten Kommentare. Mehr dazu folgt im Abschnitt »Bedingte Kommentare« weiter unten.

Internet Explorer

Für den Internet Explorer gibt es zahlreiche verschiedene Möglichkeiten, Stile vor ihm zu verbergen oder speziell für eine bestimmte Internet Explorer-Version zu definieren.

Stile vor dem Internet Explorer verbergen

Wenn Sie verhindern möchten, dass der Internet Explorer bis einschließlich Version 5.x (Windows und Mac) einen Stil ausführt, können Sie sich dessen Probleme mit Kommentaren im Selektor zunutze machen. Der Internet Explorer führt Stile nicht aus, wenn unmittelbar nach dem Selektor und vor den geschweiften Klammern ein CSS-Kommentar steht.

Das Beispiel finden Sie auf der Buch-CD in der Datei BSP/K03/ie.html.

Das können Sie sich zunutze machen. Wenn der Internet Explorer 5- einen Stil nicht ausführen soll, definieren Sie ihn beispielsweise wie im folgenden Beispiel:

Listing 3.14: Stile vor dem Internet Explorer (bis Version 5.x) verbergen

```
<html>
    <head>
        <title>IE-Browser-Weiche 1</title>
        <style type="text/css">
        <!--
           /* Den folgenden Stil sollen alle
           Browser anzeigen, nur nicht der IE5.x- */
           .nichtIE/* */ { color:red }
        -->
        </style>
    </head>
    <body>
        <h1>Beispiel zum Verbergen von Stilen vor
        dem IE</h1>
        <p class="nichtIE">Alle Browser außer dem IE
        5 und niedriger sollten hier eine rote
        Schrift anzeigen, dann wurde der Stil mit
        Kommentar nach dem Selektor ordnungsgemäß
        ausgeführt.</p>
    </body>
</html>
```

Das Kommentarzeichen sorgt dafür, dass der Internet Explorer den Stil nicht verarbeitet, im Gegensatz zu allen anderen Browsern und dem Internet Explorer ab der Version 6.

Da der Internet Explorer 6 den Fehler im Kasten-Modell von CSS nicht mehr aufweist, können Sie sich diese Browserweiche auch für den Boxmodell-Hack zunutze machen.

Mehr dazu erfahren Sie weiter unten im Abschnitt »Boxmodell-Hack«.

Stile nur vom Internet Explorer ausführen lassen

Um einzelne Stile nur vom Internet Explorer ausführen zu lassen, gibt es eine einfache Möglichkeit. Sie besteht darin, die fehlerhafte Verarbeitung des Universal-Selektors auszunutzen.

Mehr zum Universal-Selektor und den anderen Selektortypen finden Sie in Kapitel 2, »Stile definieren«.

Der Internet Explorer verfährt wie folgt, wenn er einen Universal-Selektor am Anfang eines Selektors findet. Er schneidet das Sternchen ab und verarbeitet dann den Stil. Dadurch erkennt er nicht, dass der Selektor `* html` nicht gültig ist, da es kein Element vor dem `html`-Element gibt. Der CSS-Standard gibt aber vor, dass ungültige Selektoren nicht verarbeitet werden dürfen. Daran halten sich alle Browser, nur der Internet Explorer nicht und zwar sowohl die Mac- wie die Windows-Version einschließlich Version 6 und höher.

Das Beispiel finden Sie auf der Buch-CD in der Datei BSP/K03/ie.html.

Möchten Sie beispielsweise einen Stil definieren, der nur vom Internet Explorer ausgeführt wird, bietet sich dazu der folgende Selektor an: `* html .nurIE`. Wobei der Klassen-Selektor `.nurIE` auch durch einen Element-Selektor oder einen ID-Selektor ersetzt werden kann. Das Element bzw. das Element mit der angegebenen ID muss sich dann nur innerhalb des `body`-Elements befinden.

Listing 3.15: Stile nur vom Internet Explorer ausführen lassen

```
<html>
    <head>
        <title>IE-Browser-Weiche 1</title>
        <style type="text/css">
        <!--
           /* Den folgenden Stil soll nur der IE
           anwenden */
            * html .nurIE {
                border:1px solid black;
            }
        -->
        </style>
    </head>
    <body>
        <h1>Beispiel zum Verbergen von Stilen vor dem IE</h1>
        <div class="nurIE">Nur der IE
        (einschließlich Version 6) sollte hier einen
        Rahmen anzeigen.</div>
    </body>
</html>
```

Alle anderen Browser, die den Fehler korrekt erkennen, führen den Stil nicht aus.

Abbildung 3.7: Der Internet Explorer für Macintosh führt nur den Stil .nurIE korrekt aus, nicht jedoch den Stil .ohneIE.

Boxmodell-Hack

Der Boxmodell-Hack ist für den Internet Explorer 5.x und früher notwendig, da er das Boxmodell (siehe *Kapitel 2, »Stile definieren«*) nicht korrekt darstellt. Er rechnet in die Breite eines Elements, die Sie mit `width` bestimmen, auch die Abstände (`margin`, `padding`) und den Rahmen (`border`) mit ein. Dadurch werden Elemente in der Regel zu klein dargestellt.

Der Boxmodell-Hack dient dazu, nur für den Internet Explorer eine andere Größe von Elementen zu definieren. Da es viele Möglichkeiten gibt, Stile für den Internet Explorer zu erstellen und vor diesem zu verbergen, gibt es den Boxmodell-Hack auch in verschiedenen Variationen. Das Prinzip ist jedoch immer das gleiche:

- Sie definieren zunächst die Größe eines Elements für alle Browser.
- Dann definieren Sie einen weiteren Stil, der nur vom Internet Explorer ausgeführt wird, indem Sie die Größe so korrigieren, dass der Internet Explorer das Element in der gewünschten Größe anzeigt.

Auch andersherum geht es:

- Sie legen erst in einem Stil, der für alle Browser gilt, die Größe für den Internet Explorer fest.
- Anschließend definieren Sie einen Stil, der vom Internet Explorer ignoriert wird, und legen dort die für alle anderen Browser gültige Größe des Elements fest.

Diese zweite Variante ist recht günstig, wenn Sie eine DocType-Angabe verwenden, in der der Internet Explorer 6 den Standard-Modus verwendet, z.B. `HTML Strict`. Dann können Sie nämlich obige Browserweiche nutzen, die mittels Kommentar nach dem Selektor die Abarbeitung durch den Internet Explorer bis zur Version 5.x verhindert. Genau das sind dann die Versionen, die auch das Boxmodell fehlerhaft darstellen.

Mehr zu den DocType-Angaben und deren Auswirkungen auf den Anzeige-Modus finden Sie in Kapitel 1, »CSS-Grundlagen«. Eine Übersicht über die Anzeige-Modi, abhängig von Browser und DocType-Angabe finden Sie ebenso im Anhang.

Das Beispiel finden Sie auf der Buch-CD in der Datei BSP/K03/ieBox-Modell.html.

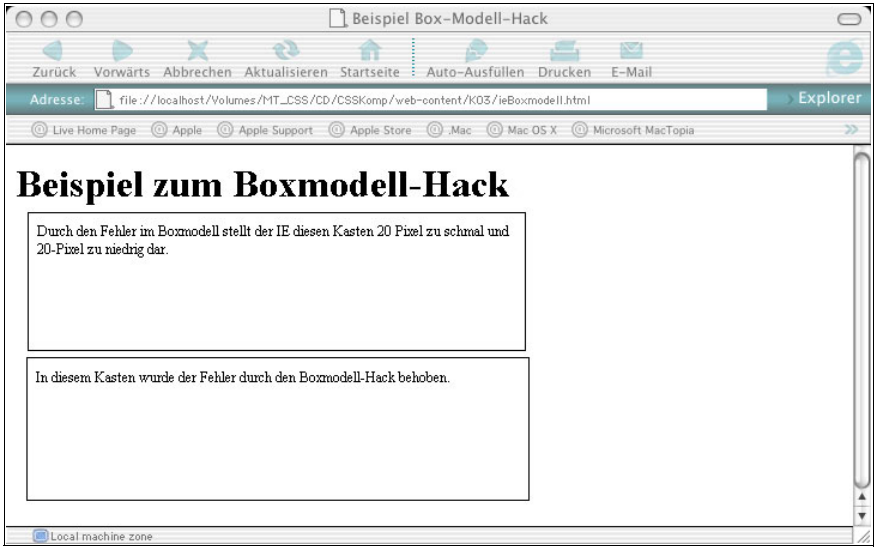


Abbildung 3.8: Die fehlerhafte Darstellung (oberer Kasten) und die korrigierte (¼) hier im Internet Explorer 5.2 für Macintosh

Um den Fehler mit dem Boxmodell-Hack zu beheben, können Sie folgendermaßen vorgehen. Sie definieren zunächst einen Stil mit den Größen für den Internet Explorer (hier der Selektor `.richtig`). Danach erstellen Sie einen Stil mit dem Selektor `.richtig/* */`, der für alle anderen Browser als dem Internet Explorer 5.x und niedriger die korrekten Werte definiert.

Im Beispiel wurde mit dem Stil `.falsch` *ein nicht korrigierter Stil für den ersten Kasten der Seite erstellt, um einen Vergleichswert für die fehlerhafte und korrigierte Darstellung des Internet Explorers zu haben. Für die Praxis benötigen Sie einen solchen Stil natürlich nicht.*

Beachten Sie bitte, dass es unbedingt notwendig ist, eine DocType-Angabe zu verwenden, die dafür sorgt, dass der Internet Explorer 6 nicht den Quirks-Modus verwendet.

Listing 3.16: Der Boxmodell-Hack zur Korrektur des Boxmodell-Fehlers im Internet Explorer 5.x-

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01//EN">
<html>
    <head>
        <title>Beispiel Boxmodell-Hack</title>
        <style type="text/css">
        <!--
            /* Vergleichsstil */
            .falsch {
                width:400px;
                height:100px;
                padding:5pt;
                margin:5pt;
                border:1px solid black;
                font-size:9pt;
            }
            /* Erst der Stil mit den Größen für den IE 5.x- */
            .richtig {
                width:422px;
                height:122px;
                padding:5pt;
                margin:5pt;
                border:1px solid black;
                font-size:9pt;
            }
            /* Dann der Stil mit den Größen für den
            IE6 und alle anderen */
            .richtig/* */ {
                width:400px;
                height:100px;
                padding:5pt;
                margin:5pt;
                border:1px solid black;
                font-size:9pt;
            }
        -->
        </style>
    </head>
    <body>
        <h1>Beispiel zum Boxmodell-Hack</h1>
        <div class="falsch">Durch den Fehler im Box-
        Modell stellt der IE diesen Kasten 20 Pixel
        zu schmal und 20-Pixel zu niedrig dar.</div>
        <div class="richtig">In diesem Kasten wurde
        der Fehler durch den Boxmodell-Hack
        behoben.</div>
    </body>
</html>
```

Für die Berechnung der Größenangabe beim Internet Explorer können Sie folgende Formel nutzen, wobei mit `width` *bzw.* `height` *die gewünschte Breite bzw. Höhe für den Inhalt gemeint ist, so wie sie gemäß Boxmodell definiert sind. Das Ergebnis der Formel weisen Sie dann im Stil für den Internet Explorer der* `width`*- bzw.* `height`*-Eigenschaft zu:*

*Höhe=height + (2 * padding) + (2 * margin) + (2 * border-width)*

*Breite=width + (2 * padding) + (2 * margin) + (2 * border-width)*

Falls Sie unterschiedliche Werte für die linken und rechten Abstände oder Rahmen verwenden:

Höhe=height + padding-top + padding-bottom + margin-bottom + margin-top + border-top + border-bottom

Breite=width + padding-left + padding-right + margin-left + margin-right + border-left + border-right

Bedingte Kommentare

Eine weitere, sehr effektive Möglichkeit zur Steuerung des Internet Explorers (ab der Version 5.x aufwärts) sind bedingte Kommentare. Sie sind zwar eine Entwicklung von Microsoft, dennoch behindern sie weder die Validierung des (X)HTML-Codes noch die Validierung des CSS-Codes.

Sie finden das Beispiel in der Datei BSP/K03/ieBedKommentare.html *auf der Buch-CD.*

Bedingte Kommentare werden nur von den Windows-Versionen des Internet Explorers ausgeführt.

Bedingte Kommentare kennzeichnen Teile der Webseite als Inhalte, die nur von einer bestimmten Internet Explorer-Version ausgeführt werden sollen. Für alle anderen Browser ist dieser Teil des Codes nur ein Kommentar. Daher behindert der Einsatz der bedingten Kommentare auch keine anderen Browser an der Darstellung der Seite und sie sind valide.

Zudem haben die bedingten Kommentare einen großen Vorteil. Es ist zu erwarten, dass sie auch von künftigen Internet Explorer-Versionen unterstützt werden, weil Microsoft sie für verschiedene Office-Erweiterungen benötigt. Außerdem basiert diese Art Browserweiche nicht auf Browserfehlern, bei denen man ja immer damit rechnen sollte, dass sie in der Zukunft beseitigt werden.

Die Syntax für bedingte Kommentare sieht wie folgt aus:

```
<!--[if Bedingung]>
...
<![endif]-->
```

Dabei ersetzen Sie `Bedingung` durch einen Ausdruck, der bestimmt, in welchen Internet Explorer-Versionen der Code ausgeführt werden soll. Zwischen den beiden Elementen folgt dann der Code, der von der entsprechenden Internet Explorer-Version ausgeführt werden soll.

Andere Browser ignorieren den Code. Für sie ist der ganze Block von `<!--` bis `-->` ein HTML-Kommentar.

Da es sich bei den bedingten Kommentaren also gemäß der Syntax um einen HTML-Kommentar handelt, können Sie sie nicht innerhalb eines `style`-Elements einfügen. Sie können aber natürlich ein `style`-Element, das Stile für bestimmte Internet Explorer-Versionen enthält, in einen bedingten Kommentar einfassen.

Das bewirkt, dass der Stil für das `p`-Element nur vom Internet Explorer 6 angewendet wird, und könnte dann beispielsweise so aussehen:

Listing 3.17: Einsetzen eines bedingten Kommentars, um Stile für den Internet Explorer 6 zu definieren.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">

<html>
    <head>
        <title>Beispiel f&uuml;r bedingte
        Kommentare</title>
        <!--[if IE 6]>
        <style type="text/css">
            p { color:blue }
        </style>
        <![endif]-->
    </head>
    <body>
        <h1>Beispiel f&uuml;r bedingte Kommentare</h1>
        <p>Im Internet Explorer 6 wird dieser Absatz
        blau formatiert. Alle anderen Browser zeigen
        ihn in Schwarz an.</p>
    </body>
</html>
```

Es gibt außerdem eine Möglichkeit, auch Stile zu definieren, die nicht nur von dem in der Bedingung ermittelten Internet Explorer ausgeführt werden, sondern auch von solchen Browsern, die keine bedingten Kommentare unterstützen. Das sind alle Browser anderer Hersteller sowie der Internet Explorer für Macintosh und alle Internet Explorer-Versionen vor 5.0 für Windows.

Dazu lassen Sie einfach die Kommentarzeichen weg. Ein solcher bedingter Kommentar könnte wie folgt aussehen:

```
<![if IE 5]>
<style type="text/css">
    h1 { color:red }
</style>
<![endif]>
```

Listing 3.18: So wird der Stil auch von anderen Browsern ausgeführt.

Diese zweite Form des bedingten Kommentars ist nicht valide. Das liegt daran, dass es sich nun nicht mehr um einen HTML-Kommentar handelt! Besser wäre daher, für die anderen Browser den Stil außerhalb eines bedingten Kommentars zu definieren und mit anderen Maßnahmen vor dem Internet Explorer zu verbergen bzw. diese Stile einfach vor denen für den Internet Explorer zu definieren, so dass der bedingte Kommentar dafür sorgt, dass die Stile für den Internet Explorer neu definiert werden.

!! STOP

Abbildung 3.9: Die Anzeige der Beispielseite im Internet Explorer 6. Der erste bedingte Kommentar wird ausgeführt, der zweite (bestimmt für IE 5) nicht.

Falls Sie zum Erstellen Ihrer Webseite einen Editor verwenden, der die Möglichkeit bietet, Kommentare beim Upload der Seite zu entfernen, oder dies vielleicht automatisch tut, laufen Sie Gefahr, die bedingten Kommentare zu verlieren. Das sollten Sie daher unbedingt vermeiden.

!! STOP

Selbstverständlich können Sie nicht nur, wie im Beispiel gezeigt, definieren, dass der Browser eine bestimmte Version haben muss. Es gibt auch zahlreiche weitere Vergleichsoperatoren:

Tabelle 3.3: Verfügbare Operatoren für die Vergleichsausdrücke der bedingten Kommentare

Operator	Beispiel	Beschreibung
	`IE 5`	Der Internet Explorer 5 führt den Code aus.
`!`	`!IE 5`	Alle Versionen, außer Version 5 führen den Code aus.
`lt`	`lt IE 6`	Die Bedingung ist erfüllt, wenn die Internet Explorer-Version kleiner als Version 6 ist. `lt` ist die Abkürzung für »less than« (kleiner als).
`lte`	`lte IE 6`	Die Bedingung ist erfüllt, wenn die Version kleiner oder gleich 6 ist. `lte` ist die Abkürzung für »less than or equal« (kleiner oder gleich).
`gt`	`gt IE 6`	Die Bedingung ist erfüllt, wenn die Version des Internet Explorers größer als 6 ist. `gt` ist die Abkürzung für »greater than« (größer als).
`gte`	`gte IE 6`	Diese Bedingung ist erfüllt, wenn die Version größer als 6 oder gleich 6 ist. `gte` ist die Abkürzung für »greater than or equal« (größer oder gleich).

Wenn Sie in den Bedingungen Versionsnummern angeben (z.B. die 5), bedeutet dies, dass die Versionsnummer des Browsers mit 5 beginnen muss, damit er den Code ausführt. In diesem Fall würden also sowohl der Internet Explorer 5 wie 5.01 und 5.5 den Code ausführen. Sie können aber auch die Unterversion mit angeben. Ein solcher Ausdruck könnte beispielsweise `if IE 5.5` *lauten. In diesem Fall würde nur der Internet Explorer 5.5 den Code ausführen.*

iCab

iCab ist ein recht neuer Browser und beruht nicht auf einer schon vorhandenen Rendering-Engine. Daher sind die CSS-Fähigkeiten bis zur Version 2.9.x noch etwas beschränkt. Es ist also wichtig, vor iCab hauptsächlich Stylesheets zu verbergen, die Anweisungen enthalten, die iCab nicht ausführen kann.

Das Beispiel finden Sie auf der Buch-CD im Ordner BSP/K03 *als* iCab.html.

Das geht recht einfach und behindert andere Browser nicht. Sie ersetzen dazu einfach ein oder mehrere Zeichen im `type`-Attribut des `style`-Elements durch ihre HTML-Entities. iCab ignoriert das Stylesheet dann, alle anderen Browser stören sich nicht daran. Sogar der Netscape Navigator 4.x führt das Stylesheet korrekt aus.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01//EN">
<html>
    <head>
        <title>iCab aussperren</title>
        <style type="&#116;ext/css">
            p { color:red }
        </style>
    </head>

    <body>
        <h1>Beispiel zum Ausschlie&szlig;en von iCab</h1>
        <p>Alle Browser stellen den Inhalt
        dieses Absatzes rot dar. Lediglich iCab
        nicht, da er das Stylesheet nicht
        anwendet.</p>
    </body>
</html>
```

Listing 3.19: iCab an der Ausführung von Stilen hindern

iCab ab der Version 3.0 führt die Stile ebenfalls aus. Das ist aber nicht schlimm, da er den CSS 2.1-Standard fast vollständig unterstützt, sogar incl. der Sprachausgabe.

Lediglich der Palm Web Browser 2.x ignoriert ein solches Stylesheet ebenfalls. Allerdings ist das eher nützlich als störend, da auch hier die CSS-Fähigkeiten noch nicht sehr gut ausgeprägt sind, wenngleich sie in einigen Bereichen die von iCab 2.9.x übersteigen.

Teil 2 CSS-Referenz

4 Textformatierungen

Textformatierungen sind sicherlich die wichtigsten Formatierungen für eine Webseite, stellen sie doch sicher, dass ihr Text nicht nur lesbar ist, sondern auch noch ansprechend aussieht. Und das Gute ist: Fast jeder Browser unterstützt die gängigen Befehle für die Textformatierung.

4.1 Einführung

Um die Schriftformatierungen für eine (X)HTML-Seite anzugeben, stellen sowohl CSS 1.0 wie auch CSS 2.0 eine Reihe von Eigenschaften zur Verfügung. Allerdings gibt es grundlegende Unterschiede bei der Handhabung dieser Angaben.

CSS 1-kompatible Browser verstehen Angaben zu Schriftarten als verbindliche Vorgaben. Wenn Sie beispielsweise als Schriftfamilie »Arial« angeben und die Schrift auf dem Zielrechner nicht existiert, wird der Text einfach in der Standardschrift des Browsers ausgegeben. Der CSS 2-Standard sieht hier eine andere Vorgehensweise vor. Hier sind die Angaben zur Schriftfamilie und anderen Eigenschaften lediglich Vorschläge. Der Browser kann anhand seiner Schriftdatenbank ähnliche Schriftarten auswählen und verwenden, wenn die im CSS-Code angegebene Schriftart nicht vorhanden ist. Für die Suche nach einer passenden, möglichst ähnlichen Schrift schreibt der CSS 2-Standard den Browsern bestimmte Suchverfahren und Schrifteigenschaften vor, die berücksichtigt werden sollen.

Zusätzlich können Sie über die `@font-face`-Regel festlegen, ob und welche Schriften wo heruntergeladen werden können.

4.2 Praxistaugliche Attribute

Nachfolgend finden Sie CSS-Attribute, die Sie problemlos oder mit nur geringen Einschränkungen in der Praxis einsetzen können. Entweder weil die Browserunterstützung sehr gut ist, oder aber ein Nichtausführen der CSS-Anweisung sich nicht so negativ auswirkt, dass die Seite nicht mehr bedienbar ist.

Buchstabenabstand (letter-spacing)

Der Buchstabenabstand legt fest, wie weit die einzelnen Buchstaben des Textes voneinander entfernt sind. Sie können damit also gesperrte oder zusammengedrückte Textformatierungen festlegen.

CSS-Element: `letter-spacing`	*Mögliche Werte:* `normal`, numerische Werte mit Einheit, `inherit`
CSS-Version: CSS 1, CSS 2, CSS 2.1, CSS 3	*Zulässig für folgende (X)HTML-Elemente:* alle
Medium: Visual	*Vererbt:* Ja

Palm	NN	Mozilla		Internet Explorer							Opera			Safari		iCab	Kq	FF
2.x	4.x	1.x	1.8	4.x	5.0x (Win)	5.0 (Mac)	5.1+ (Mac)	5.5 (Win)	6	7	6	7	8	1.x	2.0	2.9+	3.x	1.0
		●	●	●	●	●	●	●	●	●	●	●	●	●	●	●[1]	●	●

[1] siehe Erläuterungen

Wenn Sie den Wert `normal` verwenden, wird der Abstand der Buchstaben abhängig von der Breite der Zeichen der jeweiligen Schrift berechnet. Zudem kann der Browser dann auch die Zeichenabstände variieren, um beispielsweise einen optimalen Blocksatz zu erreichen.

Bei numerischen Werten darf der Browser die Zeichenabstände nicht für eine optimierte Ausrichtung variieren, sondern muss sich an das angegebene Maß halten. Numerische Werte können auch negativ sein.

Sie finden das Beispiel in der Datei BSP/K04/letter_spacing.html.

Möchten Sie einen Text, wie z.B. einen wichtigen Hinweis, gesperrt anzeigen lassen, geben Sie dazu einen positiven Wert an. Bei negativen Werten wird der Abstand der Buchstaben ausgehend vom Normalzustand um den angegebenen Wert verringert.

Beachten Sie, dass stark verengte Buchstaben nicht mehr lesbar sind. Sie sollten es daher nicht übertreiben. Schon die `-2pt` *im Beispiel sind zu viel, wenn der Text lesbar bleiben soll.*

Abbildung 4.1: Darstellung der letter-spacing-Eigenschaft, wie sie auch die anderen Browser mit Ausnahme von iCab anzeigen.

```
.normal { letter-spacing: normal }
.weiter { letter-spacing: 0.5em }
.enger  { letter-spacing: -2pt }
```

Listing 4.1: Beispiel für die letter-spacing-Eigenschaft

Alle Browser bewerten auch ein Leerzeichen als normales Zeichen, verringern bzw. erweitern daher den Abstand zwischen einem Leerzeichen und dem benachbarten Zeichen genauso wie die Abstände zwischen anderen Zeichen (siehe Abbildung 4.1). Einzig iCab 2.9.x verfährt hier anders. Er ändert die Zwischenräume zwischen Leerzeichen und anderen Zeichen nicht. Das führt dann dazu, dass die einzelnen Wortabstände gleich bleiben, d.h. in stark gedrängtem Text sind die Wortabstände erkennbar, während sie in stark erweiterten Texten nicht mehr zu sehen sind. Dies ist genau der umgekehrte Effekt wie bei den anderen Browsern.

Abbildung 4.2: So stellt iCab 2.9.x die letter-spacing-Eigenschaft dar.

:-) TIPP

Sie können dieses Verhalten von iCab korrigieren, indem Sie zusätzlich auch die `word-spacing`*-Eigenschaft setzen und damit den Wortabstand korrigieren. Mehr Informationen dazu finden Sie im Abschnitt »Wortabstand (word-spacing)«.*

Großschreibung (text-transform)

Die `text-transform`-Eigenschaft ermöglicht es Ihnen, die Groß- und Kleinschreibung von Texten zu beeinflussen. Über entsprechende Schlüsselwörter können Sie festlegen, ob der Text in Großbuchstaben oder Kleinbuchstaben umgewandelt werden soll oder ob nur der Wortanfang groß geschrieben werden soll.

CSS-Element: `text-transform`	*Mögliche Werte:* `capitalize`, `uppercase`, `lowercase`, *none*, `inherit`
CSS-Version: CSS 1, CSS 2, CSS 2.1, CSS 3, TV, Mobile	*Zulässig für folgende (X)HTML-Elemente:* alle
Medium: Visual	*Vererbt:* Ja

Palm	NN	Mozilla		Internet Explorer							Opera			Safari		iCab	Kq	FF
2.x	4.x	1.x	1.8	4.x	5.0x (Win)	5.0 (Mac)	5.1+ (Mac)	5.5 (Win)	6	7	6	7	8	1.x	2.0	2.9+	3.x	1.0
	⊙	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●

Mögliche Werte

Die möglichen Werte für die Eigenschaft haben die folgende Bedeutung:

- `capitalize`: Nur der erste Buchstabe des Wortes wird groß geschrieben.
- `uppercase`: Der Text wird in Großbuchstaben umgewandelt.
- `lowercase`: Der Text wird in Kleinbuchstaben konvertiert.
- `none`: Es gibt keine Änderung in der Groß- und Kleinschreibung.
- `inherit`: Der Wert wird vom übergeordneten Element geerbt.

CD

Das Beispiel zu dieser Eigenschaft finden Sie in der Datei BSP/K04/text_transform.html.

Einzig der Netscape Navigator 4.x hat hier ein Problem, allerdings nur ein kleines. Er wandelt keine deutschen Umlaute in Groß- oder Kleinbuchstaben um.

Abbildung 4.3: Fehlerhafte Umwandlung der deutschen Umlaute im Wort »Äquivalente«

```
.normal     { text-transform: normal }
.wortanfang { text-transform: capitalize }
.gross      { text-transform: uppercase }
.klein      { text-transform: lowercase }
```

Listing 4.2: Die Anwendung der verschiedenen Schlüsselwörter

Leerraum (white-space)

Die Eigenschaft `white-space` legt fest, wie der Browser mit Leerraum (Leerzeichen etc.) umgehen soll. So können Sie mit dieser Eigenschaft festlegen, ob ein Zeilenumbruch erfolgen darf oder nicht.

CSS-Element: `white-space`	*Mögliche Werte: normal*, `pre`, `nowrap`, `inherit`, `pre-wrap`[1], `pre-line`[1]
CSS-Version: CSS 1, CSS 2, CSS 2.1, CSS 3, TV, Mobile	*Zulässig für folgende (X)HTML-Elemente:* alle Blockelemente
Medium: Visual	*Vererbt:* Ja

[1] nur CSS 2.1

Palm	NN	Mozilla		Internet Explorer							Opera			Safari		iCab	Kq	FF
2.x	4.x	1.x	1.8	4.x	5.0x (Win)	5.0 (Mac)	5.1+ (Mac)	5.5 (Win)	6	7	6	7	8	1.x	2.0	2.9+	3.x	1.0
⊙ 3+5	○ 2+5	○ 5	○ 5		○ 5	○ 5	○ 5	○ 5	○ 5	○ 5	○ 5	○ 5	●	○ 5	●	● 3+4	●	○ 5

2 ohne `nowrap`
3 Bei `pre` werden überflüssige Leerzeichen entfernt.
4 fehlerfrei ab iCab 3.0
5 ohne CSS 2.1-Werte `pre-wrap` und `pre-line`

Die Werte für diese Eigenschaft bestimmen, wie der Browser mit Leerräumen im Text umgeht:

- `normal`: keine besondere Einstellung. Folgen von Leerzeichen werden komprimiert, das heißt, überflüssige Leerzeichen werden gelöscht, zwischen den Wörtern erfolgt bei Bedarf ein Zeilenumbruch.
- `pre`: verhindert, dass Benutzerprogramme Leerraum-Folgen komprimieren. Zeilen werden nur an Zeilenumbruchzeichen umbrochen und mehrfache Leerzeichen werden erhalten.
- `nowrap`: komprimiert Leerraum wie beim Wert `normal`, fügt aber keine Zeilenumbrüche ein.
- `pre-wrap`: verhindert das Löschen von überzähligen Leerzeichen und stellt sicher, dass der Zeilenumbruch nur an den Stellen erfolgt, an denen im Quellcode eine Absatzmarke steht.
- `pre-line`: Überzählige Leerzeichen werden gelöscht, ein Zeilenumbruch wird aber nur an der Stelle eingefügt, an der eine Absatzmarke im Quellcode steht.

Sie finden das Beispiel auf der Buch-CD im Verzeichnis BSP/K04/white_space.html.

Das folgende Beispiel zeigt dieses Verhalten. Jeder der drei Absätze im `body`-Element enthält eine Folge von Leerzeichen, die nur im mittleren Absatz auch im Browser angezeigt werden sollten.

Abbildung 4.4: Korrekt dargestellt müssen die letzten vier Absätze über die Fensterbreite hinaus laufen und im zweiten und vierten Absatz die Leerzeichen sichtbar sein.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01//EN">
<html>
    <head>
        <title>white-space-Eigenschaft</title>
        <style type="text/css">
        <!--
            .normal { white-space:normal }
            .pre    { white-space:pre }
            .nowrap { white-space:nowrap }
        -->
        </style>
    </head>
    <body>
        <h1>white-space-Eigenschaft</h1>
        <p class="normal">normal, keine besondere Einstellung.
        Folgen von Leerzeichen werden komprimiert, das hei&szlig;t,
        &uuml;berfl&uuml;ssige Leerzeichen werden gel&ouml;scht und
        zwischen den W&ouml;rtern erfolgt bei Bedarf ein
        Zeilenumbruch.</p>
        <p class="pre">pre, verhindert, dass Benutzerprogramme
        Leerraumfolgen komprimieren. Zeilen werden nur an
        Zeilenumbruchzeichen umbrochen und mehrfache Leerzeichen
        werden erhalten.</p>
        <p class="nowrap">nowrap, komprimiert Leerraum wie beim Wert
        normal, f&uuml;gt aber keine Zeilenumbr&uuml;che ein.</p>
        <p class="prewrap">pre-wrap, verhindert das L&ouml;schen von
        &uuml;berz&auml;hligen              Leerzeichen und stellt
        sicher, dass der Zeilenumbruch nur an den Stellen erfolgt,
        an denen im Quellcode eine Absatzmarke steht..</p>
        <p class="preline">pre-line, &uuml;berz&auml;hlige Leerzeichen
```

Listing 4.3: Beispiele für die white-space-Eigenschaft

```
        werden gel&ouml;scht,              ein Zeilenumbruch wird aber
        nur an der Stelle eingef&uuml;gt, an der eine Absatzmarke im
        Quellcode steht..</p>
    </body>
</html>
```

Der Netscape Navigator führt nur das Schlüsselwort `nowrap` nicht aus, unterbindet aber bei `pre` korrekt den Zeilenumbruch. Wenn Sie also dafür sorgen, dass ein mit `white-space` formatierter Absatz keine überflüssigen Leerzeichen und Zeilenumbrüche enthält, können Sie `pre` anstelle von `nowrap` verwenden.

iCab 2.9.x und der Palm Web Browser führen das Schlüsselwort `pre` nicht korrekt aus. Beide Browser komprimieren Leerzeichen, verhindern jedoch korrekt den Zeilenumbruch. In iCab 3.0 ist das Problem behoben.

Schriftfamilie (font-family)

Die Schriftfamilie legt fest, welche Schriftart verwendet werden soll. Geben Sie Arial an, heißt das jedoch nicht, dass zwingend die Schriftart »Arial« verwendet wird. Setzen Sie für den zu formatierenden Text die Schriftstärke auf Fett, kann der Browser dazu beispielsweise die Schriftart »Arial Black« verwenden. Auch diese Schrift gehört zur Schriftfamilie Arial. Die Schriftfamilie bestimmen Sie über die `font-family`-Eigenschaft. Der Standardwert für die Eigenschaft hängt vom Browser ab.

CSS-Element: `font-family`	*Mögliche Werte:* Schriftfamilie, generische Schriftart, `inherit`
CSS-Version: CSS 1, CSS 2, CSS 2.1, CSS 3, TV, Mobile	*Zulässig für folgende (X)HTML-Elemente:* alle
Medium: Visual	*Vererbt:* ja

Palm	NN	Mozilla		Internet Explorer							Opera			Safari		iCab	Kq	FF
2.x	4.x	1.x	1.8	4.x	5.0x (Win)	5.0 (Mac)	5.1+ (Mac)	5.5 (Win)	6	7	6	7	8	1.x	2.0	2.9+	3.x	1.0
[1]	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●

[1] Es wird immer die serifenlose Standardschrift verwendet.

Als Werte für die Eigenschaft können Sie einen oder mehrere Namen von Schriftfamilien oder generische Schriftfamilien angeben. Geben Sie mehrere Namen an, trennen Sie diese durch Kommata:

```
font-family: Arial, sans-serif
```

Dies würde beispielsweise die Schriftfamilie Arial als bevorzugte Schrift definieren. Steht diese Schrift dem Browser nicht zur Verfügung und kann er gemäß CSS 2-Standard auch keine geeignete Ersatzschrift finden, wird die zweite angegebene Schrift verwendet. Hier wäre dies die generische Schriftfamilie `sans-serif`.

Generische Schriftfamilien sind CSS-Schlüsselwörter, die einen Satz von ähnlichen Schriften definieren. In CSS sind fünf generische Schriftfamilien definiert:

- `serif`*: Serifen-Schriften wie Times New Roman, Palatino Linotype, Times, Tms Rmn,*
- `sans-serif`*: serifenlose Schriften. Dazu gehöre unter anderem Arial, Verdana, Helvetica etc.,*
- `cursive`*: kursive Schriften wie beispielsweise Adobe Poetica, Sanvito, Ex Ponto, Snell Roundhand, Zapf-Chancery,*
- `fantasy`*: dabei handelt es sich um Dekor-Schriften,*
- `monospace`*: nichtproportionale Schriften, bei denen alle Buchstaben die gleiche Breite beanspruchen. Das sind beispielsweise Courier und Courier New.*

Schriftfamilien, die Leerzeichen im Namen enthalten, sollten Sie in Anführungszeichen einfassen:

```
font-family:"Palatino Linotype",
            "Times New Roman", Times, serif
```

Sie sollten unbedingt am Ende der Schriftliste eine generische Schriftfamilie angeben. Damit haben Sie wenigstens einen kleinen Einfluss auf die Darstellung, falls die bevorzugten Schriften nicht vorhanden sind.

Das Beispiel finden Sie in der Datei BSP/K04/font_family.html.

```
.eineschrift      { font-family:sans-serif}
.mehrereschriften {
          font-family:Arial, Helvetica, Optima, Tahoma, sans-serif
}
.mitLeerzeichen   {
          font-family:Palatino Linotype, Hoefler Text,
                      Times New Roman, Times, Serif
}
.mitLeerzeichenAnfuehrungszeichen {
          font-family:"Palatino Linotype", "Hoefler Text",
                      "Times New Roman", Times, Serif
}
```

Listing 4.4: Mögliche Definitionen für Schriftfamilien

Schriftgröße (font-size)

Die Schriftgröße definieren Sie mit der `font-size`-Eigenschaft. Sie definiert die Größe der Schrift relativ oder absolut zur Schriftgröße des übergeordneten Elements. Ob eine Größeneinheit absolut oder relativ ist, hängt wiederum vom Ausgabemedium ab.

CSS-Element: `font-size`	*Mögliche Werte:* `xx-small` (winzig), `x-small` (sehr klein), `small` (klein), *medium* (mittel), `large` (groß), `x-large` (sehr groß), `xx-large` (extrem groß), `larger` (größer), `smaller` (kleiner), absolute und relative numerische Werte mit Einheit, prozentuale Werte, `inherit`
CSS-Version: CSS 1, CSS 2, CSS 2.1, CSS 3, TV, Mobile	*Zulässig für folgende (X)HTML-Elemente:* alle
Medium: Visual	*Vererbt:* Ja, nur der berechnete Wert

Palm	NN	Mozilla		Internet Explorer							Opera			Safari		iCab	Kq	FF
2.x	4.x	1.x	1.8	4.x	5.0x (Win)	5.0 (Mac)	5.1+ (Mac)	5.5 (Win)	6	7	6	7	8	1.x	2.0	2.9+	3.x	1.0
○ [3]	● [1]	●	●	●	●	●	●	●	●	●	●	●	●	●	●	● [2]	●	●

1 Der Skalierungsfaktor von CSS 1 wird verwendet.

2 Die Schlüsselwörter `xx-small`, `x-small` und `small` führen bis einschließlich Version 2.9.x zur gleichen Schriftgröße und die Angabe in `em` wird zu groß dargestellt. Ab iCab 3.0 fehlerfrei.

3 Die Schlüsselwörter `xx-small` und `x-small` führen zur gleichen Schriftgröße.

Als Größeneinheiten stehen die bereits in Kapitel 2, »Stile definieren« beschriebenen Maßeinheiten zur Verfügung. Die wichtigsten sind `px` *(Pixel),* `pt` *(Point),* `em` *und* `%`*.*

Beachten Sie auch die Erläuterung zur Vererbung prozentualer Werte in Kapitel 2, »Stile definieren«. Diese Schwierigkeiten wirken sich nämlich vor allem auf die Vererbung von Schriftgrößen aus.

Optimal für skalierbare Stylesheets sind Größenangaben in `em` *oder Prozentwerten. Die Schlüsselwörter* `larger` *und* `smaller` *weisen der Schrift einen um den Faktor 1,2 größeren (*`larger`*) bzw. kleineren (*`smaller`*) Wert zu, als das Element von seinem übergeordneten Element geerbt hat. Wird der Wert* `medium` *vererbt, ist* `smaller` *identisch mit* `small` *und* `larger` *gleichwertig mit* `large`*.*

Das Beispiel finden Sie als Datei BSP/K04/font_size.html *auf der Buch-CD.*

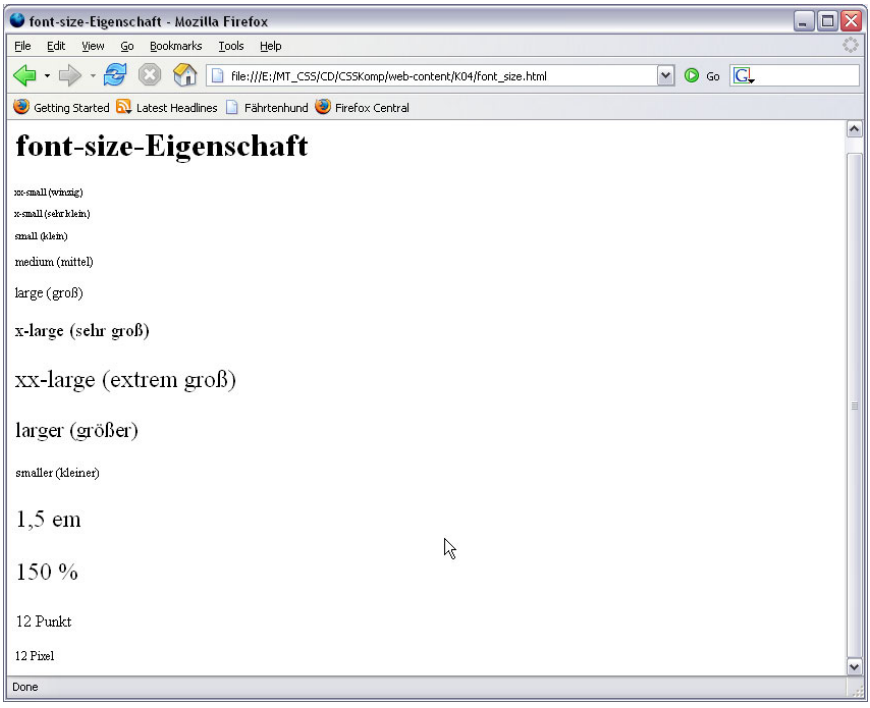


Abbildung 4.5: Anzeige der verfügbaren Schriftgrößen in Firefox

Im Gegensatz zu CSS 1 liegt bei CSS 2 ein Skalierungsfaktor von 1,2 zwischen den benachbarten Werten der Schlüsselwörter `xx-small` *bis* `xx-large`*. Bei CSS 1 lag ein Skalierungsfaktor von 1,5 zwischen den Werten. Hat also die Standardschrift des Browsers eine Größe von 10 Pixel, hat die Schriftgröße* `large` *die Größe 12 Pixel.*

Schriftstärke (font-weight)

Mit der Schriftstärke, auch Schriftgewichtung genannt, können Sie bestimmen, wie fett oder dünn eine Schrift dargestellt werden soll.

CSS-Element: `font-weight`	*Mögliche Werte: normal*, `bold`, `bolder`, `lighter`, `100`, `200`, `300`, *`400`*, `500`, `600`, `700`, `800`, `900`, `inherit`
CSS-Version: CSS 1, CSS 2, CSS 2.1, CSS 3, TV, Mobile	*Zulässig für folgende (X)HTML-Elemente:* alle
Medium: Visual	*Vererbt:* Ja

Palm	NN	Mozilla		Internet Explorer							Opera			Safari		iCab	Kq	FF
2.x	4.x	1.x	1.8	4.x	5.0x (Win)	5.0 (Mac)	5.1+ (Mac)	5.5 (Win)	6	7	6	7	8	1.x	2.0	2.9+	3.x	1.0
○ [1]	○ [2]	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●

[1] nur die Schlüsselwörter, keine numerischen Werte
[2] nur die Schlüsselwörter `bold` und `normal` sowie numerische Werte

Sie finden das Beispiel in der Datei BSP/K04/font_weight.html.

Für die Eigenschaft können Sie sowohl Schlüsselwörter wie `bold`, `bolder` und `lighter` angeben als auch numerische Werte. Bei diesen gilt: je kleiner die Zahl, desto dünner die Schrift. Der Wert 400 gilt als normal.

In der Regel verfügen die Schriftfamilien aber nicht über entsprechend viele Schriftvarianten, um die feinen Abstufungen zu erreichen, die Sie über die numerischen Werte festlegen können. In diesen Fällen werden Werte von 100 bis 500 normal und höhere Werte fett dargestellt.

iCab, auch in der Version 3.0, und der Internet Explorer für Macintosh stellen schon den Wert 500 fett dar.

Schriftstil (font-style)

Der Schriftstil gibt an, welchen Stil der Text haben soll, ob er normal, kursiv oder geneigt angezeigt wird. Den Schriftstil legen Sie über die `font-style`-Eigenschaft fest.

CSS-Element: `font-style`	*Mögliche Werte: normal*, `italic`, `oblique`, `inherit`
CSS-Version: CSS 1, CSS 2, CSS 2.1, CSS 3, TV, Mobile	*Zulässig für folgende (X)HTML-Elemente:* alle
Medium: Visual	*Vererbt:* Ja

Palm	NN	Mozilla		Internet Explorer							Opera			Safari		iCab	Kq	FF
2.x	4.x	1.x	1.8	4.x	5.0x (Win)	5.0 (Mac)	5.1+ (Mac)	5.5 (Win)	6	7	6	7	8	1.x	2.0	2.9+	3.x	1.0
●	○ [1]	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●

[1] kein `oblique`

Für den Schriftstil stehen die Werte

- `normal`: ohne besonderen Stil
- `italic`: kursive Schrift
- `oblique`: geneigte Schrift

zur Verfügung.

Wenn Sie einen vererbten Schriftstil wie `italic` wieder ausschalten möchten, setzen Sie den Schriftstil für das Element auf `normal`.

Ob die Ausgaben für die Werte `oblique` *und* `italic` *unterschiedlich aussehen, hängt davon ab, ob der entsprechende Schriftstil für die Schriftfamilie vorhanden ist. Falls der Schriftstil* `oblique` *nicht zur Verfügung steht, wird von den meisten Browser der kursive Schriftstil auch für* `oblique` *verwendet.*

Das Beispiel finden Sie in der Datei BSP/K04/font_style.html *auf der Buch-CD.*

Die Anwendung der Stile könnte wie folgt aussehen. Zunächst wird hier die Schriftfamilie festgelegt und dann über die Klassenstile für die einzelnen Absätze der Schriftstil definiert.

Listing 4.5: Der CSS-Code für die Testseite

```
body, p {
        font-family:arial,tahoma, verdana, helvetica, sans-serif
}
.inherit { font-style:inherit }
.normal  { font-style:normal }
.kursiv  { font-style:italic }
.oblique { font-style:oblique }
```

Schriftvariante (font-variant)

Mit der Schriftvariante bestimmen Sie, ob der Text normal oder mit Kapitälchen angezeigt werden soll.

Kapitälchen sehen aus wie Großbuchstaben, werden jedoch als Kleinbuchstaben im Text in einer geringeren Größe oder zumindest einer geringeren Höhe angezeigt, sofern die Schriftart nicht explizit einen Zeichensatz für Kapitälchen zur Verfügung stellt.

CSS-Element: `font-variant`	*Mögliche Werte: normal*, `small-caps`, `inherit`
CSS-Version: CSS 1, CSS 2, CSS 2.1, CSS 3, TV, Mobile	*Zulässig für folgende (X)HTML-Elemente:* alle
Medium: Visual	*Vererbt:* Ja

Palm	NN	Mozilla		Internet Explorer							Opera			Safari		iCab	Kq	FF
2.x	4.x	1.x	1.8	4.x	5.0x (Win)	5.0 (Mac)	5.1+ (Mac)	5.5 (Win)	6	7	6	7	8	1.x	2.0	2.9+	3.x	1.0
		●	●	○ [1]	○ [1]	●	●	○ [1]	●	●	●	●	●	●	●	● [2]	●	●

1 Alle Buchstaben werden als Großbuchstaben dargestellt.
2 ab iCab 3.0

CD

Das Beispiel finden Sie in der Datei BSP/K04/font_variant.html.

Abbildung 4.6: Korrekte Darstellung der Testseite im Internet Explorer 6

:-) TIPP

Sie können auch die `text-transform`*-Eigenschaft verwenden, um Einfluss auf die Groß- und Kleinschreibung zu nehmen.*

Textausrichtung (text-align)

Mit der Eigenschaft `text-align` können Sie die horizontale Ausrichtung des Inhalts eines Blocks einstellen.

CSS-Element: `text-align`	*Mögliche Werte:* `left` (linksbündig), `right` (rechtsbündig), `center` (zentriert), `justify` (Blocksatz), Zeichenkette[1], `inherit`
CSS-Version: CSS 1, CSS 2, CSS 2.1, CSS 3, TV, Mobile	*Zulässig für folgende (X)HTML-Elemente:* alle Blockelemente
Medium: Visual	*Vererbt:* Ja

Palm	NN	Mozilla		Internet Explorer							Opera			Safari		iCab	Kq	FF
2.x	4.x	1.x	1.8	4.x	5.0x (Win)	5.0 (Mac)	5.1+ (Mac)	5.5 (Win)	6	7	6	7	8	1.x	2.0	2.9+	3.x	1.0
○[2]	○[3]	●	●	○[3]	●	●	●	●	●	●	●	●	●	●	●	○[2]	●	●

1 nicht in CSS 2.1, TV und Mobile-Spezifikation
2 kein Blocksatz
3 kein Blocksatz in der Linux/Mac-Version

Das Beispiel finden Sie in der Datei BSP/K04/text_align.html.

Neben den oben genannten Schlüsselwörtern können Sie auch Zeichenketten einsetzen. Sie dienen jedoch ausschließlich dazu, Werte in Tabellenzellen an einem Zeichen, wie beispielsweise einem Dezimalzeichen, auszurichten.

Mehr dazu erfahren Sie in Kapitel 9, »Tabellen«. Dort folgt dann auch ein Beispiel und die Angabe der Browserkompatibilität speziell für diesen Wert.

Alle Browser, außer dem Internet Explorer 6 und 7 für Windows, richten mit der `text-align`*-Eigenschaft keine Tabellen innerhalb von übergeordneten Tabellenzellen aus. Die innere Tabelle wird immer linksbündig innerhalb der übergeordneten Zelle ausgegeben.*

Textdekoration (text-decoration)

Mit der Eigenschaft `text-decoration` können Sie festlegen, mit welchen zusätzlichen Attributen der Text versehen werden soll. Diese Attribute werden auch als Textauszeichnung bezeichnet.

CSS-Element: `text-decoration`	*Mögliche Werte: None* (keine Dekoration), `underline` (unterstrichen), `overline`[2] (überstrichen), `line-through`[2] (durchgestrichen), `blink`[2+1] (blinkend), `inherit`
CSS-Version: CSS 1, CSS 2, CSS 2.1, CSS 3, TV, Mobile	*Zulässig für folgende (X)HTML-Elemente:* alle
Medium: Visual	*Vererbt:* Nein

Palm	NN	Mozilla		Internet Explorer							Opera			Safari		iCab	Kq	FF
2.x	4.x	1.x	1.8	4.x	5.0x (Win)	5.0 (Mac)	5.1+ (Mac)	5.5 (Win)	6	7	6	7	8	1.x	2.0	2.9+	3.x	1.0
●	○ 3	●	●	○ 4	○ 4	○ 4	○ 4	●	● 6	○ 4	○ 4	●	●	○ 4	○ 4	● 5	○ 4	●

1 nicht in TV
2 nicht in Mobile
3 keine Überstreichung
4 kein Blinken
5 Darstellungsfehler beim Blinken und falsche Anwendung auf untergeordnete Elemente in iCab bis zur Version 2.9.x. Ab Version 3.0 fehlerfrei.
6 siehe Erläuterungen

INFO

Die Werte `underline`, `overline` *und* `line-through` *und* `blink` *dürfen auch zusammen verwendet werden. Nur* `none` *und ein anderer Wert bzw.* `inherit` *und ein anderer Wert schließen sich aus. Wenn Sie Werte kombinieren möchten, trennen Sie die Schlüsselwörter einfach durch ein Leerzeichen.*

Wenden Sie die Eigenschaft auf Elemente an, die keinen Textinhalt haben, wie beispielsweise Grafiken oder Multimedia-Elemente, wird die Eigenschaft ignoriert, wenn sich der Browser standardkonform verhält.

CD

Sie finden das Beispiel auf der Buch-CD in der Datei BSP/K04/text_decoration.html. *Es zeigt die Anwendung, speziell auch die Kombination von Werten für die* `text-decoration`*-Eigenschaft.*

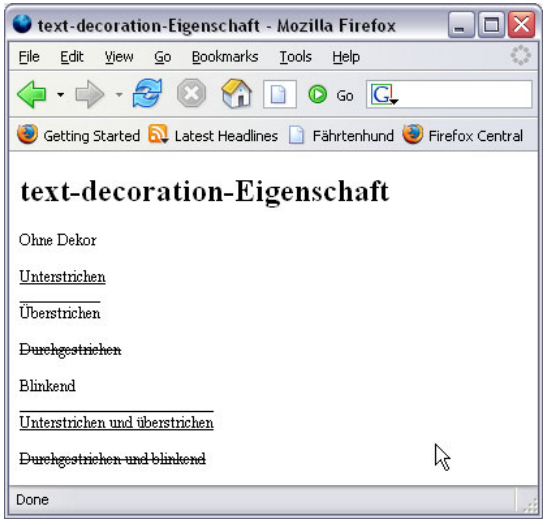


Abbildung 4.7: Firefox stellt alle Eigenschaft korrekt dar.

```
.ohne            { text-decoration: none }
.unterstrichen   { text-decoration: underline }
.ueberstrichen   { text-decoration: overline }
.durchgestrichen { text-decoration: line-through }
.blinkend        { text-decoration: blink }
.unterueber      { text-decoration: overline underline }
.durchblink      { text-decoration: blink line-through }
```

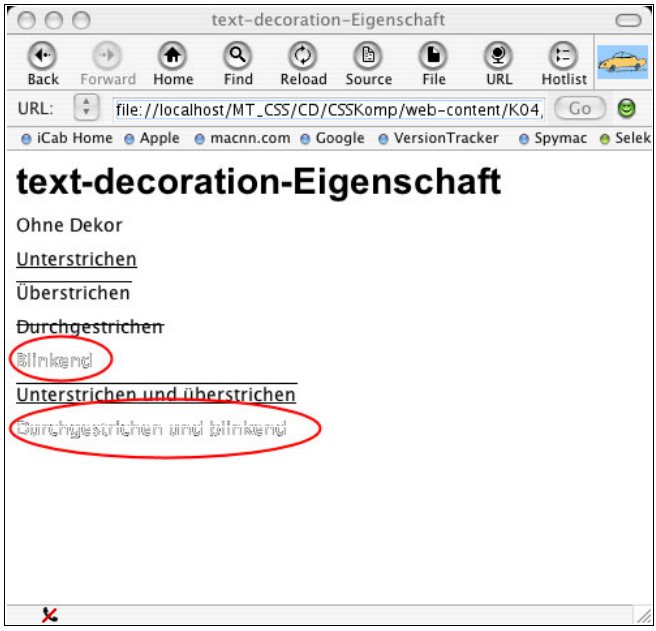
Listing 4.6: Anwendung der text-decoration-Eigenschaft

iCab 2.9.x stellt zwar alle Werte dar, verursacht jedoch beim Blinken Darstellungsfehler. Obwohl der CSS-Standard eindeutig besagt, dass Blinken das abwechselnde Aus- und Einblenden des Elements bezeichnet, bleibt der ausgeblendete Text in iCab sichtbar, wenn auch nur als grauer Schatten. In iCab 3.0 ist die Darstellung fehlerfrei.

Die Eigenschaft wird zwar nicht auf die untergeordneten Elemente vererbt, dennoch übt sie darauf Einfluss aus. Gemäß CSS-Standard sollen alle untergeordneten Elemente, die sich innerhalb eines Blockelements befinden, das mit `text-decoration` *formatiert ist, mit den gleichen Ausschmückungen angezeigt werden. Außerdem müssen sie die ursprüngliche Farbe beibehalten, auch wenn beispielsweise der unterstrichene Text eine andere Farbe hat. Folgendes Beispiel verdeutlicht das.*

Das Inline-Element im `p`*-Element hat die Farbe Rot und es wurde keine Textdekoration festgelegt. Korrekt muss der Browser aber auf das Inline-Element auch die Unterstreichung anwenden und zwar in der Farbe, die durch die* `color`*-Eigenschaft des übergeordneten Elements bestimmt wird.*

Abbildung 4.8: Darstellungsfehler beim blinkenden Text



Listing 4.7: Beispiel für die Übernahme der Werte auf untergeordnete Elemente

```
<p style="text-decoration:underline">Dieser Text und alle seine <span
   style="color:red">inline</span>-Elemente sollten unterstrichen sein
   und zwar in der Farbe Schwarz.</p>
```

Abbildung 4.9: So wird das Beispiel korrekt dargestellt.

Dieser Text und alle seine inline-Elemente sollten unterstrichen sein und zwar in der Farbe Schwarz.

iCab 2.9.x stellt das Inline-Element fälschlicherweise mit roter Unterstreichung dar.

Der Internet Explorer 6 zeigt ein recht unterschiedliches Verhalten. Abhängig von der Unterversion wird das Schlüsselwort `blink` manchmal korrekt ausgeführt, manchmal nicht.

Texteinrückung (text-indent)

Die Texteinrückung, die Sie mit der `text-indent`-Eigenschaft festlegen, bezeichnet den Abstand der ersten Zeile zum umgebenden Block. Prozentuale Werte beziehen sich auf die Breite des umschließenden Blockelements. Der angegebene Wert darf auch negativ sein, dann wird die erste Zeile aus dem Block herausgerückt.

CSS-Element: `text-indent`	*Mögliche Werte: 0*, numerische Werte mit Größeneinheit, prozentuale Werte, `inherit`
CSS-Version: CSS 1, CSS 2, CSS 2.1, CSS 3, TV, Mobile	*Zulässig für folgende (X)HTML-Elemente:* alle Blockelemente
Medium: Visual	*Vererbt:* Ja

Palm	NN	Mozilla		Internet Explorer							Opera			Safari		iCab	Kq	FF
2.x	4.x	1.x	1.8	4.x	5.0x (Win)	5.0 (Mac)	5.1+ (Mac)	5.5 (Win)	6	7	6	7	8	1.x	2.0	2.9+	3.x	1.0
○ [1]	●	●	●	● [2]	● [2]	●	●	● [2]	●	●	●	●	●	●	●	● [3]	●	●

1 Nur negative Werte werden korrekt dargestellt.
2 Prozentuale Werte können abhängig von der Vererbungshierarchie fehlerhaft dargestellt werden.
3 fehlerfrei ab iCab 3.0

Das Beispiel finden Sie auf der Buch-CD als Datei BSP/K04/text_indent.html.

Ein negativer Wert für die `text-indent`*-Eigenschaft kann nur dann korrekt dargestellt werden, wenn das umgebende Blockelement ausreichend Platz bietet, indem der mit* `padding` *definierte Innenabstand entsprechend großzügig definiert ist. Ansonsten ist die negative Einrückung durch den Innenabstand des umgebenden Elements begrenzt.*

Aus diesem Grund legt das Beispiel zunächst für das umgebende `body`-Element einen Innenabstand von 20 Punkt fest.

Listing 4.8: Beispiel zur text-indent-Eigenschaft

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01//EN">
<html>
    <head>
        <title>text-indent-Eigenschaft</title>
        <style type="text/css">
        <!--
            body     { padding:20pt }
            .ti1cm   { text-indent:1cm }
            .ti3em   { text-indent:3em }
            .ti50pro { text-indent:50% }
            .tineg   { text-indent:-15pt; border:1px solid black }
        -->
        </style>
    </head>
...
</html>
```

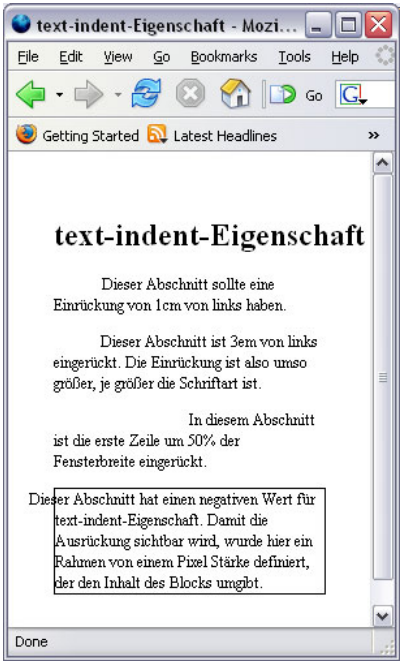


Abbildung 4.10: Korrekte Darstellung der text-indent-Eigenschaft in Mozilla

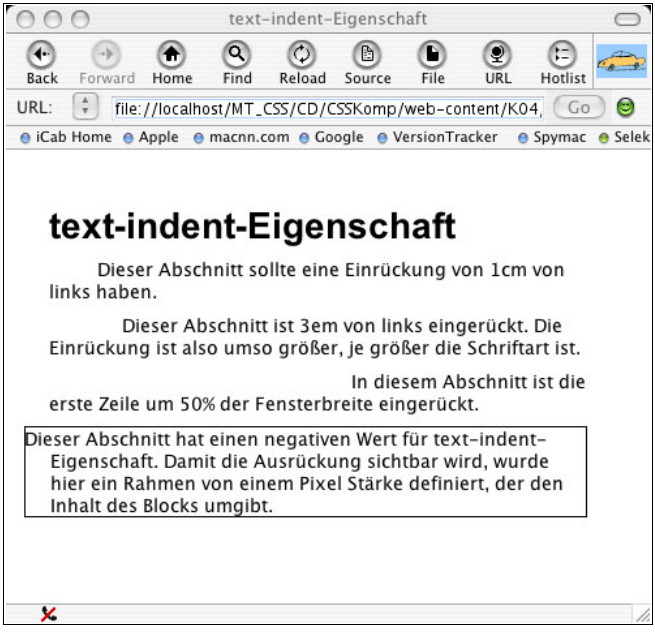


Abbildung 4.11: Fehlerhafte Darstellung in iCab 2.9.x

iCab 2.9.x hat ganz erhebliche Probleme mit der Darstellung der Eigenschaft. Der Einzug in `em` wird zu groß dargestellt, ebenso die prozentuale Einrückung. Hier scheint es so, als ob iCab nicht die verfügbare Innenbreite des `body`-Elements als Grundlage verwendet, wie es gemäß Boxmodell korrekt wäre, sondern die Gesamtbreite des Fensters.

Auch der negative Einzug wird nicht korrekt dargestellt. Zwar rückt iCab den Text tatsächlich aus und in den Innenabstand des umgebenden Blockelements hinein, aber den Rahmen zeichnet er nunmehr um den Einzug herum, was gemäß Boxmodell nicht korrekt ist. iCab 3.0 stellt die Eigenschaft korrekt dar.

Mehr zum Boxmodell finden Sie in Kapitel 5, »Größen, Abstände und Positionierung«.

Vertikale Textausrichtung (vertical-align)

Mit der vertikalen Ausrichtung können Sie bestimmen, wie der Inhalt eines Blockelements mit enthaltenen Inline-Elementen in Inline-Elementen oder in Tabellenzellen in der Vertikalen ausgerichtet werden soll.

CSS-Element: `vertical-align`	*Mögliche Werte: baseline*, `sub`, `super`, `top`[1], `text-top`[2], `middle`[1], `bottom`[1], `text-bottom`[2], Prozentwert[2] (negativ und positiv), numerischer Wert[3] mit Einheit, `inherit`
CSS-Version: CSS 2, CSS 3, Mobile, TV	*Zulässig für folgende (X)HTML-Elemente:* Inline-Elemente und Tabellenzellen
Medium: Visual	*Vererbt:* Nein

Palm	NN	Mozilla		Internet Explorer							Opera			Safari		iCab	Kq	FF
2.x	4.x	1.x	1.8	4.x	5.0x (Win)	5.0 (Mac)	5.1+ (Mac)	5.5 (Win)	6	7	6	7	8	1.x	2.0	2.9+	3.x	1.0
●	○[4]	●	●	○	○[5]	●	●	●	●	●	●	●	●	●	●	●[6]	●	●

1 nicht in Mobile-Spezifikation
2 nicht in Mobile- und TV-Spezifikation
3 nur in CSS 2 und 3
4 nur in Tabellenzellen
5 ohne numerische und prozentuale Werte, `text-top` und `middle`
6 ab Version 3.0

Da nur iCab 2.9.x die Eigenschaft nicht unterstützt und der Netscape Navigator 4.x die Eigenschaft zumindest in Tabellenzellen berücksichtigt, wird diese Eigenschaft noch als praxistauglich eingestuft. Schließlich haben beide Browser zusammen immer noch einen äußerst geringen Marktanteil und die neue, aktuelle iCab-Version 3.0 bietet Unterstützung für diese Eigenschaft.

Die Werte der `vertical-align`-Eigenschaft lassen sich in zwei Typen einteilen. Der erste Typ, dazu gehören die Werte `top`, `bottom` und `middle`, richtet den Text des Inline-Elements an den Grenzen der Inline-Box aus. Die Ausrichtung erfolgt also unabhängig von der Grundlinie des umgebenden Textes.

Näheres zu Inline-Elementen und Boxen finden Sie in Kapitel 5, »Größen, Abstände und Positionierung«.

Bei allen anderen Werten wird die Grundlinie des formatierten Inline-Elements verschoben und zwar in Abhängigkeit von der Grundlinie des umgebenden Textes. Die Grundlinie eines Textes ist die Linie, auf der alle Buchstaben angeordnet werden. Buchstaben wie »j«, »g« etc. ragen nach unten jedoch über die Grundlinie heraus. Die untere Kante dieser Buchstaben stellt die untere Textkante dar, die obere Kante der Großbuchstaben und der hohen Kleinbuchstaben wie »l« und »k« stellen die obere Textkante dar.

Abbildung 4.12: Darstellung der Grundlinie eines Textes

Die Beispieldatei finden Sie auf der Buch-CD als Datei BSP/K04/vertical_align.html.

Wie der Text angeordnet wird, hängt also nicht nur von der Formatierung des übergeordneten Elements ab, sondern auch von den angegebenen Werten. Daher bedürfen sie einer näheren Betrachtung.

Mögliche Werte

- Prozentwerte können sowohl negativ als auch positiv sein und beziehen sich immer auf den Wert der `line-height`-Eigenschaft. Das bedeutet, hat die Zeile beispielsweise eine Höhe von 30 Pixel und geben Sie `vertical-align: 33%` an, wird die Grundlinie des Textes um ca. 10 Pixel

nach oben geschoben. Bei `vertical-align: -33%` würde der Text hingegen um ca. 10 Pixel nach unten verschoben.

- Numerische Werte können ebenfalls negativ oder positiv sein. Sie verschieben die Grundlinie des Textes um den angegebenen Wert nach oben (positiv) oder unten (negativ) und zwar relativ zur Einstellung `baseline`.

Mit den folgenden beiden Stilen können Sie erreichen, dass der Text um 30% bzw. um 30 Pixel hochgestellt angezeigt wird.

```
.prozent { vertical-align:30% }
.pixel30 { vertical-align:30px }
```

Allerdings ist es dazu erforderlich, dass auch die Schriftgröße abweicht, da sonst eine unterschiedliche Ausrichtung in Prozenten nicht sichtbar wäre. Dafür sorgen die beiden Elementstile `span` und `p`.

```
span { font-size:60% }
p    { font-size:1.5em }
```

Um die Klassen `.prozent` und `.pixel30` anzuwenden, fügen Sie den zu formatierenden Inhalt einfach in `span`-Elemente ein:

```
<p>Dieser Text befindet sich auf der Grundlinie, <span
class="prozent">dieser 30% &uuml;ber der Grundlinie</span> und
<span class="pixel30">dieser 30 Pixel &uuml;ber der Grundlinie</
span>.</p>
```

Abbildung 4.13: Darstellung der Anwendung von numerischen und prozentualen Werten

- `baseline`: richtet die Grundlinie des Textes an der Grundlinie der übergeordneten Box aus. Dies ist der Standardwert.
- `top`: richtet die obere Kante der Inline-Box an der oberen Kante der übergeordneten Inline-Box aus.
- `bottom`: richtet die untere Kante der Inline-Box an der unteren Kante der übergeordneten Inline-Box aus.
- `middle`: zentriert den Text mittig in der Box. Dabei wird jedoch nicht wirklich die Mitte der Box berücksichtigt, sondern der Text wird an der Grundlinie des übergeordneten Elements zuzüglich der Hälfte der

Höhe des Buchstaben »x« des übergeordneten Elements verwendet. Eine mittige Ausrichtung erreichen Sie daher nur dann, wenn die Grundlinie des übergeordneten Elements nicht verschoben wurde.

- `sub`: verschiebt die Grundlinie der Box nach unten, um eine Tiefstellung des Textes zu erreichen.
- `super`: verschiebt die Grundlinie für hochgestellte Texte nach oben.

!! STOP

Die Werte `sub` *und* `super` *verändern nicht die Schriftgröße. Wenn Sie also Formeln setzen möchten oder Fußnotenzeichen einfügen möchten, bei denen üblicherweise die hoch- bzw. tiefgestellten Texte kleiner dargestellt werden, müssen Sie selbst im Stil die Schriftgröße reduzieren.*

- `text-top`: richtet die obere Kante der Box an der oberen Schriftkante der übergeordneten Box aus.
- `text-bottom`: richtet die untere Kante der Box an der unteren Textkante der übergeordneten Box aus.

Folgendes Beispiel demonstriert die Verwendung der Werte. Zunächst werden für die Werte `baseline` bis `text-bottom` eigene Klassenstile definiert. Anschließend werden diese dem `span`-Element im Seiteninhalt zugewiesen.

Listing 4.9: Erforderliche Stile

```
<style type="text/css">
<!--
    .grundlinie   { vertical-align:baseline }
    .tiefgestellt { vertical-align:sub }
    .hochgestellt { vertical-align:super }
    .oben         { vertical-align:top }
    .mittig       { vertical-align:middle }
    .unten        { vertical-align:bottom }
    .textoben     { vertical-align:text-top }
    .textunten    { vertical-align:text-bottom }
    span          { font-size:60% }
    p             { font-size:1.5em }
-->
</style>
```

!! STOP

Beachten Sie, dass die meisten Formatierungen nur dann sichtbar sind, wenn das formatierte Element eine kleinere Schriftgröße als das übergeordnete Element hat. Daher wurde hier im Elementstil für das `span`*-Element eine Schriftgröße von 60% definiert. Das ist auch der Grund dafür, dass die hoch- und tiefgestellten Zeichen in der Formel kleiner dargestellt werden.*

Listing 4.10: Anwendung der Stile

```
<p>Mit Hilfe der vertical-align-Eigenschaft legen Sie die
vertikale Ausrichtung von Text fest. So k&ouml;nnen Sie bspw.
auch Formeln setzen:</p>
<p>Y=X<span class="hochgestellt">2</span>
-(2<span class="tiefgestellt">n</span>*e)</p>
<p>Aber auch f&uuml;r andere Zwecke l&auml;sst sich die
```

```
Eigenschaft einsetzen. <span class="grundlinie">So
ist dieser Text bspw. an der Grundlinie ausgerichtet</span>,
dieser <span class="textoben">an der oberen Kante
</span>des umgebenden Textes und
<span class="textunten">dieser an der unteren
Kante</span>.
Mit den Werten <span class="mittig">middle</span>,
<span class="unten">bottom</span> und
<span class="oben">top</span> richten Sie den Text
innerhalb des Inline-Elements und nicht an der oberen oder
unteren Kante des umgebenden Textes aus.</p>
```

Abbildung 4.14: Ergebnis des Codes in Firefox

Vordergrundfarbe (color)

Die Vordergrundfarbe legt neben der Schriftfarbe für normalen Text auch die Farben von Rahmenlinien und anderen Elementen fest, wenn sie nicht separat definiert werden. Sie wird über die CSS-Eigenschaft `color` definiert.

CSS-Element: `color`	*Mögliche Werte:* Farbwert, `inherit`
CSS-Version: CSS 1, CSS 2, CSS 2.1, CSS 3, TV, Mobile	*Zulässig für folgende (X)HTML-Elemente:* alle
Medium: Visual	*Vererbt:* ja

Palm	NN	Mozilla		Internet Explorer							Opera			Safari		iCab	Kq	FF
2.x	4.x	1.x	1.8	4.x	5.0x (Win)	5.0 (Mac)	5.1+ (Mac)	5.5 (Win)	6	7	6	7	8	1.x	2.0	2.9+	3.x	1.0
● [1]	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	● [2]	●	●

[1] nur Farbnamen und hexadezimale Kurzform
[2] ab Version 3.0 vollständig und fehlerfrei

Als Werte für die `color`-Eigenschaft kommen neben `inherit` Farbwerte in Frage.

INFO

Farbwerte können Sie in CSS als Farbnamen, als RGB-Werte oder in hexadezimaler Weise angeben. HTML 4 definiert 16 Farbnamen: `aqua`, `black`, `blue`, `fuchsia`, `gray`, `green`, `lime`, `maroon`, `navy`, `olive`, `purple`, `red`, `silver`, `teal`, `white` ***und*** `yellow`.

Wenn Sie die Farben als RGB-Werte angeben möchten, funktioniert das mit folgender Syntax: `color:rgb(rot,grün,blau)`. *Jeder Farbanteil kann Werte von 0 bis einschließlich 255 haben. Grautöne erzielen Sie, indem Sie für jede Farbe den gleichen Wert angeben. Außerdem können Sie die Farben auch in prozentualen Werten angeben, beispielsweise mit* `rgb(100%,0%,0%)` *für die Farbe Rot.*

Alternativ können Sie auch hexadezimale Werte angeben. Diese haben die folgende Syntax: `#rrggbb`, *wobei* `rr`, `gg` *und* `bb` *wieder jeweils den Rot-, Grün- und Blauanteil in hexadezimaler Schreibweise angeben. Für hexadezimale Farbangaben steht eine Kurzform zur Verfügung. Wenn die Farbwerte für die einzelnen Farbanteile zwei gleiche Zeichen haben, also* `00`, `FF`, `AA`, `EE`, `22` *etc. können Sie sie zusammenfassen. Das ergibt dann eine dreistellige Farbangabe. Aus* `#FF0000` *können Sie somit* `#F00` *machen, aus* `#FF2200` *entsprechend* `#F20`.

Demnach sind die folgenden Stile gleichwertig:

Listing 4.11: Möglichkeiten zur Definition der Vordergrundfarbe

```
.farbname        { color:red} }
.rgbwert        { color:rgb(255,0,0) }
.rgbwertProzent { color:rgb(100%,0%,0%) }
.hexad          { color:#FF0000 }
.hexadkurz       { color:#F00 }
```

REF

Das Beispiel finden Sie in der Datei BSP/K04/color.html.

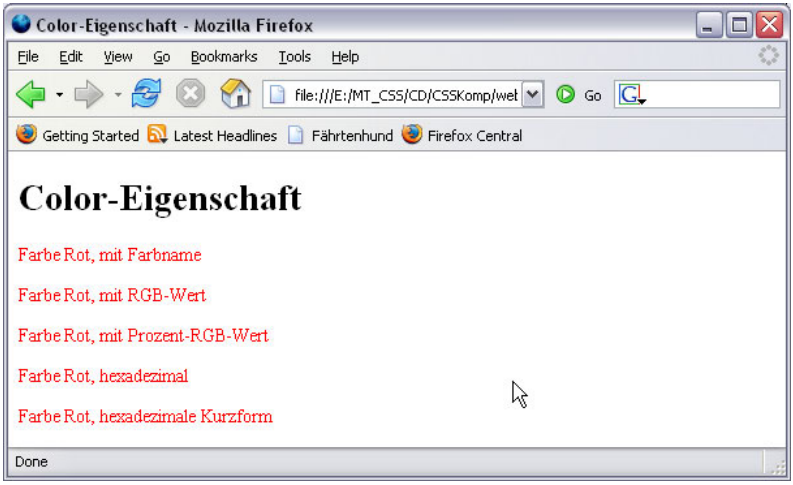


Abbildung 4.15: Darstellung der verschiedenen Farbwerte für die color-Eigenschaft in Firefox

Wortabstand (word-spacing)

Anders als der Buchstabenabstand beeinflusst der Wortabstand lediglich den Abstand zwischen den Wörtern. Dabei ist ein Wort definiert als eine Zeichenfolge, die von Leerzeichen oder dem Anfang bzw. Ende des Textes begrenzt ist.

CSS-Element: `word-spacing`	*Mögliche Werte: normal*, numerische Werte mit Einheit, `inherit`
CSS-Version: CSS 1, CSS 2, CSS 2.1, CSS 3	*Zulässig für folgende (X)HTML-Elemente:* alle
Medium: Visual	*Vererbt:* Ja

Palm	NN	Mozilla		Internet Explorer							Opera			Safari		iCab	Kq	FF
2.x	4.x	1.x	1.8	4.x	5.0x (Win)	5.0 (Mac)	5.1+ (Mac)	5.5 (Win)	6	7	6	7	8	1.x	2.0	2.9+	3.x	1.0
		●	●	●	●	●	●	●	●	●	●	●	●	●	●	● [1]	●	●

[1] siehe Erläuterungen

Für die möglichen Werte und deren Auswirkungen gilt prinzipiell das Gleiche wie schon zum Buchstabenabstand (siehe Abschnitt »Buchstabenabstand (letter-spacing)«) gesagt wurde.

Das Beispiel finden Sie in der Datei BSP/K04/word_spacing.html.

Listing 4.12: Verwendung der word-spacing-Eigenschaft

```
.normal    { word-spacing: normal }
.weiter    { word-spacing: 1em }
.enger     { word-spacing: -2pt }
.erweitert { word-spacing:1em; letter-spacing:0.5em }
```

Sie können mit Hilfe der `word-spacing`*-Eigenschaft auch dafür sorgen, dass iCab 2.9.x die Wortabstände exakt so anzeigt wie die anderen Browser, indem Sie mit der* `letter-spacing`*-Eigenschaft immer auch die* `word-spacing`*-Eigenschaft angeben. Das könnte dann wie in der letzten Zeile des Listings aussehen.*

Abbildung 4.16: Darstellung der Klassenstile im Opera-Browser

Zeilenhöhe (line-height)

Sie können mit der Eigenschaft `line-height` die Zeilenhöhe bestimmen. Sie legt im Prinzip fest, wie groß der Abstand zwischen zwei Zeilen ist.

CSS-Element: `line-height`	*Mögliche Werte: normal*, numerischer Wert mit Einheit, Zahl ohne Einheit, prozentualer Wert, `inherit`
CSS-Version: CSS 1, CSS 2, CSS 2.1, CSS 3, TV	*Zulässig für folgende (X)HTML-Elemente:* alle
Medium: Visual	*Vererbt:* Ja

Palm	NN	Mozilla		Internet Explorer							Opera			Safari		iCab	Kq	FF
2.x	4.x	1.x	1.8	4.x	5.0x (Win)	5.0 (Mac)	5.1+ (Mac)	5.5 (Win)	6	7	6	7	8	1.x	2.0	2.9+	3.x	1.0
●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●

Wenn `line-height` und `font-size` die gleiche Größe haben, gibt es keinen Abstand zwischen den Zeilen. In der Regel ist die Zeile aber höher als die Schriftgröße. In diesem Fall gibt es eine Differenz zwischen der Zeilenhöhe und der Schrifthöhe, die als Durchschuss bezeichnet wird. Normalerweise teilt ein Browser diese Differenz durch zwei und zeigt die Hälfte (Halbdurchschuss) oberhalb und die andere Hälfte unterhalb des Textes an. Der Halbdurchschuss zweier aufeinander folgender Zeilen ergibt dann den optischen Zeilenabstand, der genau genommen der Abstand zwischen zwei Texten ist.

Für den Fall, dass `line-height` einen Wert hat, der kleiner als die Schriftgröße ist, kann die Schrift die Grenzen der Zeilenbox (siehe Boxmodell in *Kapitel 5, »Größen, Abstände und Positionierung«*) verlassen und andere Zeilen überlappen.

Das Beispiel finden Sie in der Datei BSP/K04/line_height.html.

Mögliche Werte

Die verfügbaren Werte wirken sich unterschiedlich auf die Darstellung aus. In keinem Fall dürfen die Werte negativ sein.

- `normal`: In diesem Fall berechnet der Browser einen sinnvollen Wert für die Zeilenhöhe, der abhängig von der Schriftgröße ist. Der so berechnete Wert wird genauso verwendet, als wenn Sie ihn ohne eine Einheit als Wert für die `line-height`-Eigenschaft angeben. Der berechnete Wert sollte zwischen 1 und 1,2 liegen.
- numerischer Wert mit Einheit: Diese Angabe legt die Höhe der Zeile fest. Zulässig sind alle Einheiten, die für Längenwerte zulässig sind, ausgenommen Prozent.
- Zahl (ohne Einheit): Die angegebene Zahl wird vom Browser mit der Schriftgröße multipliziert und als Zeilenhöhe verwendet. Geben Sie also die Zahl 1,2 an und hat die Schrift eine Größe von 10 Pixel, dann hat die Zeile eine Höhe von 12 Pixel, so dass je ein Pixel oberhalb und unterhalb der Schrift verbleibt.

Vererbt wird nicht der berechnete Wert für die Zeilenhöhe, sondern die angegebene Zahl.

- prozentualer Wert: Hier wird der Prozentwert ebenfalls mit der Schriftgröße multipliziert und das Berechnungsergebnis als Zeilenhöhe verwendet.

Im folgenden Beispiel werden vier Absätze formatiert, die die Anwendung der verschiedenen Wertetypen zeigen. Zahlen und Prozentwerte kleiner als 1 bzw. 100% reduzieren den Zeilenabstand, Werte, die größer sind, erhöhen ihn. Gleiches gilt für Werte in em. Wenn Sie wie im Beispiel `1,5em` als Zeilenhöhe angeben, dann ist der Zeilenabstand 1,5 mal so groß wie die Schrift.

Listing 4.13: Klassenstile des Beispiels

```
.normal          { line-height: normal }
.numerischerWert { line-height: 1.5em }
.zahl            { line-height:2 }
.prozent         { line-height: 80% }
```

Abbildung 4.17: So sollte das Beispiel korrekt dargestellt werden.

4.3 Nicht/schlecht unterstützte Attribute

In dieser Rubrik finden Sie solche Attribute, die von vielen oder zumindest von sehr wichtigen Browsern nicht oder fehlerhaft unterstützt werden und die somit noch nicht praxistauglich sind.

Anführungszeichen (quotes)

Die Eigenschaft `quotes` definiert, welche Anführungszeichen für Zitate verwendet werden sollen.

CSS-Element: `quotes`	*Mögliche Werte:* `none`, Quotes-Wertepaar
CSS-Version: CSS 2, CSS 3	*Zulässig für folgende (X)HTML-Elemente:* alle
Medium: Visual	*Vererbt:* Ja

Palm	NN	Mozilla		Internet Explorer							Opera			Safari		iCab	Kq	FF
2.x	4.x	1.x	1.8	4.x	5.0x (Win)	5.0 (Mac)	5.1+ (Mac)	5.5 (Win)	6	7	6	7	8	1.x	2.0	2.9+	3.x	1.0
		● 1	● 1				⊙ 1				● 1	● 1	● 1		● 1	● 1	● 1	● 1

1 nur für das q-Element

Als Wertepaar wird eine Folge von Zeichenketten akzeptiert. Dabei gibt die erste das öffnende und die zweite das schließende Anführungszeichen an. Mit `quotes:"«" "»"` *werden also französische Anführungszeichen definiert. Alternativ können Sie anstelle der Zeichenfolge auch HTML-Codes (*`quotes:&#181 &#187`*) oder Unicode-Nummern (*`quotes:\00AB \00BB`*) angeben.*

Falls Sie normale doppelte, gerade Anführungszeichen verwenden möchten, wie sie in HTML und CSS auch zum Einfassen von Zeichenketten genutzt werden, fassen Sie die Zeichenketten einfach in Hochkommata ein, beispielsweise `quotes:'"' '"'`.

Sie finden das Beispiel auf der Buch-CD in der Datei BSP/K04/quotes.html.

Unter Mac OS X stellen alle drei Browser, unabhängig von den für `quotes` angegebenen Werten, die Anführungszeichen als doppelte, gerade Anführungszeichen dar. Offenbar bewirkt hier die Angabe von `quotes` nur, dass überhaupt Anführungszeichen erzeugt werden. Ab iCab Version 3.0 werden hingegen französische Anführungszeichen angezeigt und nicht die definierten.

Firefox stellt auch auf dem Mac die Anführungszeichen wie gewünscht korrekt dar, solange sie für ein q-Element angegeben wurden.

Schriftdehnung (font-stretch)

Die Schriftdehnung gibt an, wie stark die einzelnen Zeichen zusammengedrückt oder auseinander gezogen sind. Je weiter sie zusammengedrückt sind, desto schmaler sind die einzelnen Buchstaben und desto geringer der Abstand zwischen ihnen. Die Schriftdehnung bestimmen Sie über die CSS-Eigenschaft `font-stretch`.

CSS-Element: `font-stretch`	*Mögliche Werte: normal*, `wider`, `narrower`, `ultra-condensed`, `extra-condensed`, `condensed`, `semi-condensed`, `semi-expanded`, `expanded`, `extra-expanded`, `ultra-expanded`, `inherit`
CSS-Version: CSS 1, CSS 2, CSS 3	*Zulässig für folgende (X)HTML-Elemente:* alle
Medium: Visual	*Vererbt:* Ja

Palm	NN	Mozilla		Internet Explorer							Opera			Safari		iCab	Kq	FF
2.x	4.x	1.x	1.8	4.x	5.0x (Win)	5.0 (Mac)	5.1+ (Mac)	5.5 (Win)	6	7	6	7	8	1.x	2.0	2.9+	3.x	1.0
																●[1]		

[1] ab Version 3.0

Hier gilt wie bei den vorherigen Schrifteigenschaften, dass die Möglichkeiten der Darstellung von den verfügbaren Schriften der Schriftfamilie abhängen. Gibt es keine Condensed-Schrift, kann das entsprechende Schlüsselwort auch nicht dargestellt werden, unabhängig von den Fähigkeiten des Browsers.

Alle Schlüsselwörter bis auf `wider`, `narrower` *und* `inherit` *geben eine definierte Dehnung an, von* `ultra-condensed` *(sehr stark gestaucht) bis* `ultra-expanded` *(sehr stark gedehnt).* `Wider` *wählt den nächstmöglichen Wert, der eine stärkere Dehnung als beim vererbten Wert angibt,* `narrower` *staucht die Schrift mehr, als der vererbte Wert bestimmt.* `wider` *und* `narrower` *sind damit relative Werte. Folgendes Beispiel soll dies verdeutlichen.*

Das Beispiel finden Sie auf der Buch-CD als Datei BSP/K04/font_stretch.html.

Nehmen Sie an, Sie haben drei Klassenstile erstellt und legen darüber eine normale Schriftweite sowie die Werte `wider` und `narrower` fest. Außerdem haben Sie einen Absatz mit der Klasse `.normal`, der `span`-Elemente mit den anderen beiden CSS-Klassen enthält.

```
<style type="text/css">
<!--
    body, p   {
          font-family:Arial, Optima, Helvetica, sans-serif;
    }
    .normal   { font-stretch: normal }
    .wider    { font-stretch: wider }
```

Listing 4.14: Verwendung der Schlüsselwörter wider und narrower

```
    .narrower { font-stretch: narrower }
-->
</style>
...
<p class="normal">normaler Text, <span class="wider">mehr
   gedehnter(wider)</span> und <span class="narrower">mehr
   gestauchter(narrower)</span> Text</p>
```

In diesem Fall bewirkt das Schlüsselwort `wider` in der Klasse `.wider`, dass ausgehend von dem vererbten Wert `normal` die nächstgedehntere Variante, `semi-expanded` verwendet wird. Ausgehend vom Wert `normal` bewirkt `narrower`, dass das entsprechende `span`-Element in einer weiter gestauchten Schrift formatiert wird, dies wäre `semi-condensed`.

Abbildung 4.18: Korrekte Darstellung in iCab 3.0 Beta; da die Schriftart nur 3 Varianten bietet, werden alle condensed-Werte und alle expanded-Werte gleich dargestellt.

Textausrichtung der letzten Zeile (text-align-last)

Die Eigenschaft `text-align-last` legt die Ausrichtung der letzten Zeile eines Elements fest, wenn das Element selbst mit Blocksatz (`text-align:justify`) formatiert wurde.

CSS-Element: `text-align-last`	*Mögliche Werte: start*, `end`, `center`, `left`, `right`, `justify`, numerische Werte
CSS-Version: CSS 3	*Zulässig für folgende (X)HTML-Elemente:* alle
Medium: Visual	*Vererbt:* Ja

Palm	NN	Mozilla		Internet Explorer							Opera			Safari		iCab	Kq	FF
2.x	4.x	1.x	1.8	4.x	5.0x (Win)	5.0 (Mac)	5.1+ (Mac)	5.5 (Win)	6	7	6	7	8	1.x	2.0	2.9	3.x	1.0
									o[1]	o[1]								

[1] unterstützt werden die Werte `right`, `left`, `center`, `justify`.

Sie finden das Beispiel auf der Buch-CD in der Datei BSP/K04/text_align_last.html.

- `start` und `end`: richten den Text der letzten Zeile abhängig von der Richtung des Textflusses aus. Bei Ländereinstellungen, die von links nach rechts schreiben, wird der Text mit dem Wert `start` also links, in Ländern, bei denen von rechts nach links geschrieben wird, rechts ausgerichtet.
- `right`, `center` und `left`: richten die letzte Zeile immer links bzw. rechts aus, unabhängig von der Schreibrichtung.
- `justify`: verwendet auch für die letzte Zeile Blocksatz.
- Numerischer Wert: definiert einen Faktor, um den der Inhalt der letzten Zeile skaliert wird.

Das folgende Beispiel zeigt die Verwendung der einzelnen Werte in Klassenstilen. Damit die Eigenschaft überhaupt angewendet wird, werden alle Absätze mit Blocksatz formatiert und auf eine Breite von 150 Pixel beschränkt. Das ermöglicht auch bei kurzen Beispieltexten, den Blocksatz über mehrere Zeilen sichtbar zu machen.

Listing 4.15: Ausrichtung der letzten Zeile

```
<style type="text/css">
<!--
    p        { text-align:justify; width:150px; }
    .start   { text-align-last:start }
    .end     { text-align-last:end }
    .center  { text-align-last:center }
    .right   { text-align-last:right }
    .left    { text-align-last:left }
    .justify { text-align-last:justify }
    .numWert { text-align-last:2 }
-->
</style>
```

Derzeit ist der Internet Explorer 6 und 7 für Windows der einzige Browser, der die Eigenschaft zumindest in Teilen darstellt. Berücksichtigt werden die Werte `center`, `right`, `justify` und `left`.

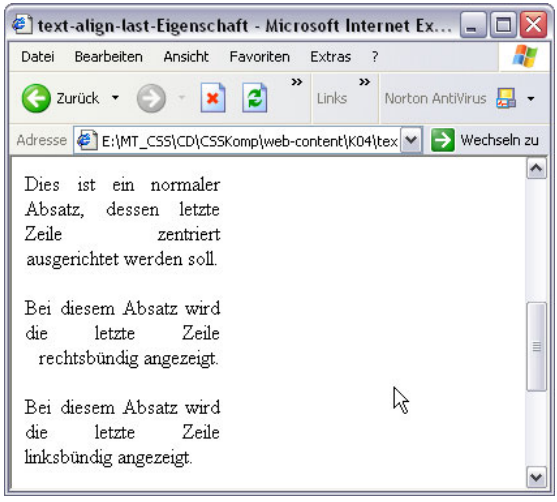


Abbildung 4.19: Darstellung von drei vom Internet Explorer unterstützten Werte left, right und center

Textschatten (text-shadow)

Mit der Eigenschaft `text-shadow` können Sie einen Text mit Schatten hinterlegen. Zumindest theoretisch. In der Praxis sieht das leider noch schlecht aus, da diese Eigenschaft bisher nur von Safari und Konqueror in Teilen angewendet wird.

Generell liegt der Schatten immer hinter dem Text, niemals darüber und er vergrößert niemals die Box des Elements, sondern ragt im Zweifelsfall über die Box hinaus.

CSS-Element: `text-shadow`	*Mögliche Werte:* `none`, Schattenwert
CSS-Version: CSS 2, CSS 3	*Zulässig für folgende (X)HTML-Elemente:* alle
Medium: Visual	*Vererbt:* Nein

Palm	NN	Mozilla		Internet Explorer							Opera			Safari		iCab	Kq	FF
2.x	4.x	1.x	1.8	4.x	5.0x (Win)	5.0 (Mac)	5.1+ (Mac)	5.5 (Win)	6	7	6	7	8	1.x	2.0	2.9	3.x	1.0
															○ [1]		○ [1]	

[1] Schatten wird nur angezeigt, wenn alle vier Werte für den Schattenwert angegeben werden, auch die optionalen

Ein Schattenwert setzt sich aus mehreren einzelnen Angaben zusammen. Die Syntax lautet: `text-shadow: x y Überblendradius Farbe`

`y` *und* `x` *geben dabei die Verschiebung des Schattens nach unten und nach rechts an und müssen angegeben werden. Negative Werte würden den Schatten nach links bzw. oben verschieben. Der Überblendradius bestimmt, die Stärke des Schattens, das heißt, wie breit er über den Text hinausragt. Für die Farbe dürfen Sie einen normalen Farbwert (siehe dazu Abschnitt »Vordergrundfarbe (color)«) angeben.*

Sie dürfen sowohl die Farbe als auch den Überblendradius weglassen. Bei fehlender Farbe bekommt der Schatten die gleiche Farbe wie der Text, bei fehlendem Überblendradius wird der Schatten gleichmäßig ausgeblendet.

Jeder Schatteneffekt muss einen Abstand für den Schatten festlegen und kann optional einen Überblendradius und eine Schattenfarbe angeben.

Damit wenigstens Safari und Konqueror den Schatten anzeigen, sollten Sie aber immer alle Angaben definieren.

Sie finden das Beispiel auf der Buch-CD in der Datei BSP/K04/text_shadow.html.

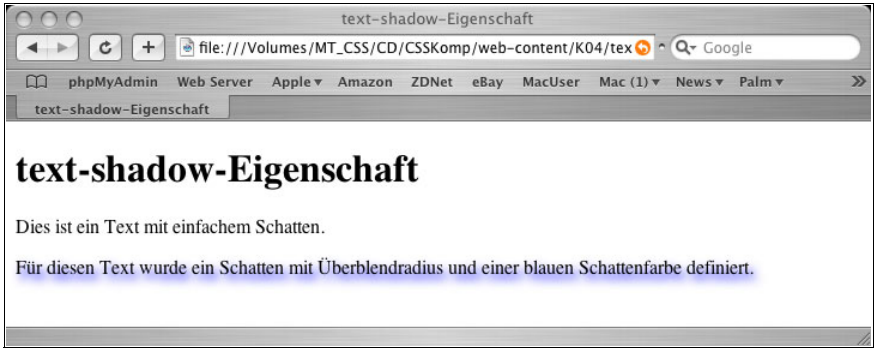


Abbildung 4.20: Safari zeigt zumindest die Schatten an, für die alle Werte angegeben wurden.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01//EN">

<html>
    <head>
        <title>text-shadow-Eigenschaft</title>
        <style type="text/css">
        <!--
            .schatten1 { text-shadow:4px 6px }
            .schatten2 { text-shadow:4px 6px 10px blue }
        -->
        </style>
    </head>
```

Listing 4.16: Beispiel für die text_shadow-Eigenschaft

```
    <body>
        <h1>text-shadow-Eigenschaft</h1>
        <p class="schatten1">Dies ist ein Text mit
            einfachem Schatten. </p>
        <p class="schatten2">F&uuml;r diesen Text
            wurde ein Schatten mit
            &Uuml;berblendradius und einer blauen
            Schattenfarbe definiert.</p>
    </body>

</html>
```

Typ der Textausrichtung (text-justify)

Die Eigenschaft `text-justify` legt fest, wie der Browser die optimale Ausrichtung von Wörtern und Zeichen für den Blocksatz berechnet.

CSS-Element: `text-justify`	*Mögliche Werte: auto*, `inter-word`, `inter-ideograph`, `distribute`, `distribute-all-lines`, `newspaper`
CSS-Version: CSS 3	*Zulässig für folgende (X)HTML-Elemente:* alle Blockelemente
Medium: Visual	*Vererbt:* Ja

Palm	NN	Mozilla		Internet Explorer							Opera			Safari		iCab	Kq	FF
2.x	4.x	1.x	1.8	4.x	5.0x (Win)	5.0 (Mac)	5.1+ (Mac)	5.5 (Win)	6	7	6	7	8	1.x	2.0	2.9	3.x	1.0
									●	●								

Sie finden das Beispiel auf der Buch-CD in der Datei BSP/K04/text_justify.html.

Die möglichen Werte haben folgende Auswirkungen:

- `auto`: Der Browser bestimmt, wie er den Blocksatz berechnet und anzeigt.
- `inter-word`: Die Abstände zwischen den Wörtern werden innerhalb einer Zeile so variiert, dass jede einzelne Zeile gefüllt ist. Dies ist die schnellste und einfachste Methode, die die meisten Browser auch beim Wert `auto` anwenden. Die letzte Zeile des Absatzes wird dabei nicht berücksichtigt.

Wenn Sie bestimmen möchten, wie die letzte Zeile ausgerichtet werden soll, verwenden Sie dazu die Eigenschaft `text-align-last`. *Näheres dazu finden Sie im Abschnitt »Textausrichtung der letzten Zeile (text-align-last)«.*

- `inter-ideograph`: Bei dieser Einstellung werden auch die Abstände zwischen den Zeichen variiert. Die Expansion der Zeichenabstände ist jedoch auf die breiteren Zeichen beschränkt. Auch hierbei wird die letzte Zeile nicht berücksichtigt.
- `distribute`: Bei dieser Methode werden sowohl die Wort- wie die Zeichenabstände bei Bedarf erweitert oder reduziert. Die letzte Zeile des Absatzes wird nicht berücksichtigt.
- `distribute-all-lines`: Dieser Typ funktioniert wie der Typ `distribute`, nur wird auch die letzte Zeile (abhängig von ihrer Länge) berücksichtigt. Kann die letzte Zeile nicht ausgerichtet werden, weil sie beispielsweise nur ein Bild oder ein einzelnes Zeichen enthält, wird der Inhalt der Zeile zentriert.
- `newspaper`: Bei dieser Methode werden sowohl die Wort- wie auch die Zeichenabstände variiert und das nicht nur innerhalb einer Zeile. Auch die vorstehenden und nachstehenden Zeilen werden berücksichtigt. Dabei kann der Browser entscheiden, wie viele Zeilen Berücksichtigung finden. Auch hier wird die letzte Zeile nicht ausgerichtet.

Wenn die Spalten, in denen Sie den Blocksatz verwenden, sehr klein sind, sollten Sie den Wert `newspaper` *verwenden. Dies ist zwar die langsamste Methode, aber dennoch die exakteste.*

Der Internet Explorer 6+ für Windows stellt die Eigenschaft korrekt dar, wenngleich beim Wert `inter-ideograph` kein Unterschied zu `inter-word` zu erkennen ist.

Abbildung 4.21: Die Darstellung der verschiedenen Blocksatz-Methoden im Internet Explorer

Schrifteffekt (font-effect)

Die Eigenschaft `font-effect` definiert zusätzliche Schrifteffekte.

CSS-Element: `font-effect`	*Mögliche Werte: none*, `emboss` (hervorgehoben), `engrave` (eingedrückt), `outline` (Kontur)
CSS-Version: CSS 3	*Zulässig für folgende (X)HTML-Elemente:* alle
Medium: Visual	*Vererbt:* Ja

Palm	NN	Mozilla		Internet Explorer							Opera			Safari		iCab	Kq	FF
2.x	4.x	1.x	1.8	4.x	5.0x (Win)	5.0 (Mac)	5.1+ (Mac)	5.5 (Win)	6	7	6	7	8	1.x	2.0	2.9	3.x	1.0

Sie finden das Beispiel auf der Buch-CD in der Datei BSP/K04/font_effekt.html.

Das folgende Listing zeigt die Anwendung der verschiedenen Werte in Form von Inline-CSS-Code. Eine Darstellung ist derzeit leider noch nicht möglich, weil kein Browser die Eigenschaft unterstützt.

```
<body>
    <h1>font-effect-Eigenschaft</h1>
    <p style="font-effect:none">Kein Schrifteffekt</p>
    <p style="font-effect:engraved">Eingedrückt</p>
    <p style="font-effect:embossed">Hervorgehoben</p>
    <p style="font-effect:outline">Kontur</p>
</body>
```

Listing 4.17: Anwendung der font-effect-Eigenschaft

@font-face-Regel

Mit der `@font-face`-Regel können Sie eigene Schriftfamilien definieren und dem Browser den Download von Schriften ermöglichen.

Die `@font-face`-Regel hat folgende Syntax:

```
@font-face { Beschreibung }
```

Dabei erfolgt die Beschreibung wie in normalen CSS-Stilen mit `Name:Wert`-Paaren.

CSS-Element: `@font-face{}`	*Mögliche Werte:* Schrifteigenschaften (`font-family`, `font-style`, `font-variant`, `font-weight`, `font-stretch`, `font-size`) und `src:url()`
CSS-Version: CSS 2	*Zulässig für folgende (X)HTML-Elemente:* alle
Medium: Visual	*Vererbt:* Nein

Palm	NN	Mozilla		Internet Explorer							Opera			Safari		iCab	Kq	FF
2.x	4.x	1.x	1.8	4.x	5.0x (Win)	5.0 (Mac)	5.1+ (Mac)	5.5 (Win)	6	7	6	7	8	1.x	2.0	2.9	3.x	1.0

Beispiel

Mit folgender `@font-face`-Regel können Sie eine Schriftfamilie definieren, die unter dem mit `src` angegebenen URL heruntergeladen werden kann.

Listing 4.18: Schriftartbeschreibung mit @font-face

```
@font-face {
    font-family: meineschrift;
    src: url("http://domain/ordner/meineschrift")
}
```

Die `@font-face`-Regel wird von keinem der großen Browser unterstützt. Daher ist sie derzeit noch nicht praxistauglich.

4.4 Praxisbeispiel

Das folgende Beispiel zeigt, wie Sie mit wenigen Absätzen in einem Dokument eine Willkommensseite gestalten können, bei der das Wort »Willkommen« in mehreren Sprachen und Formatierungen angezeigt wird. Ziel der Formatierung ist vor allem, eine Anordnung der Wörter zu erreichen, die zufällig wirkt, ohne jedoch von absoluter Positionierung Gebrauch zu machen.

Sie finden das Beispiel auf der Buch-CD in der Datei BSP/K04/praxisbeispiel.html.

Abbildung 4.22: Das Ergebnis in Firefox

Basis dieser Seite sind nur wenige Absätze, die die Wörter in verschiedener Reihenfolge enthalten. Jedes Wort umgeben Sie mit einem `span`-Element und weisen diesem, abhängig von der Sprache, eine CSS-Klasse zu. Für die Absätze, in denen die Wörter enthalten sind, legen Sie ebenfalls eine CSS-Klasse fest, die lediglich die Basisgröße der Schrift vorgibt.

```
<body>
    <p class="klein">
        <span class="englisch">welcome</span>
        <span class="franzoesisch">bienvenue</span>
        <span class="deutsch">willkommen</span>
        <span class="englisch">welcome</span>
```

Listing 4.19: Der Seiteninhalt

```
        <span class="franzoesisch">bienvenue</span>
        <span class="deutsch">willkommen</span>
    </p>
    <p>

        <span class="deutsch">willkommen</span>
        <span class="franzoesisch">bienvenue</span>
        <span class="englisch">welcome</span>
        <span class="deutsch">willkommen</span>
        <span class="franzoesisch">bienvenue</span>
    </p>
    <p class="gross">
        <span class="englisch">welcome</span>
        <span class="franzoesisch">bienvenue</span>
        <span class="deutsch">willkommen</span>
    </p>
    <p class="klein"><span class="englisch">welcome</span>
        <span class="franzoesisch">bienvenue</span>
        <span class="deutsch">willkommen</span>
        <span class="englisch">welcome</span>
        <span class="franzoesisch">bienvenue</span>
        <span class="deutsch">willkommen</span>
    </p>
</body>
```

Wichtig für den Zufallseffekt ist vor allem, dass Sie für die einzelnen Sprachen unterschiedliche relative Schriftgrößen festlegen. Das führt dann im Beispiel dazu, dass englische Wörter zwar immer kleiner als die deutschen angezeigt werden, dennoch wird die in den CSS-Klassen `gross` und `klein` definierte Schriftgröße berücksichtigt, indem die tatsächliche Schriftgröße relativ zu dieser Größe berechnet wird.

Der weitere Schritt zu einer zufälligen Anordnung besteht darin, für jede Sprache eine andere Schriftfarbe und Schriftauszeichnung zu verwenden. Für die englischen Texte wird beispielsweise über `font-style:italic` eine kursive Formatierung verwendet, deutsche Wörter werden unterstrichen dargestellt (`text-decoration:underline`) und französische ohne besondere Auszeichnung.

Zu guter Letzt müssen Sie noch dafür sorgen, dass die Wörter nicht alle in einer Reihe angezeigt werden, sondern versetzt zueinander. Auch die Abstände zwischen den einzelnen Wörtern sollten Sie variieren. Die dazu verwendeten Eigenschaften sind `text-indent` für einen Einrückung des Wortes, die einen größeren Abstand von links verursacht. Über die Eigenschaft `vertical-align` können Sie den Text, ausgehend von der Grundlinie des benachbarten Elements, nach unten oder oben verschieben. `padding-bottom` verschiebt das Inline-Element nach unten, indem der Innenabstand vergrößert wird.

Listing 4.20:
Die notwendigen Formatierungen

```
<style type="text/css">
    <!--
    p { font-weight:bold }
    .englisch  {
        color: #c06;
        font-size: 150%;
        font-family: Georgia, "Times New Roman", Times, serif;
        font-style: italic;
        letter-spacing: 80%;
        vertical-align: -150%
    }
    .deutsch  {
        color: #336;
        font-size: 200%;
        font-family: Verdana, Arial, Helvetica, sans-serif;
        line-height: 200%;
        font-stretch: expanded;
        text-decoration: underline;
        text-transform: uppercase;
        text-indent: 3em;
        vertical-align: -50%
    }
    .franzoesisch  {
        color: #396;
        font-family: "Courier New", Courier, Monaco, monospace;
        line-height: 170%;
        text-indent: 5em;
        word-spacing: 150%;
        padding-bottom: 3em;
        vertical-align: 120%
    }
    .gross { font-size:4em }
    .klein { font-size:2em }
    -->
</style>
```

5 Größen, Abstände und Positionierung

Wenn Sie komplexere Layouts mit CSS erstellen möchten, kommen Sie nicht umhin, auch die Größe und Position der Elemente auf der Webseite zu steuern. Hierfür benötigen Sie Kenntnisse über die Art und Weise, wie diese Positionierung funktioniert. Beschrieben wird das im CSS-Standard im Kapitel über das visuelle Formatierungsmodell und das Boxmodell. Beide werden nachfolgend kurz erläutert, damit die Beschreibungen der einzelnen CSS-Eigenschaften verständlich sind.

5.1 Einführung

Das visuelle Formatierungsmodell von CSS bestimmt, wie visuelle Elemente, z.B. für die Bildschirm- und Druckausgabe, formatiert werden. Es legt damit nicht nur fest, in welcher Reihenfolge die Elemente dargestellt werden, sondern auch ob und wie sie sich überlappen. Die Größe und der Inhalt eines Elements werden hingegen über das Boxmodell bestimmt. Es legt fest, wie Rahmenlinien, Innenabstände (`padding`) und Außenabstände (`margin`) sich auf die Größe des Elements und den Platz für seinen Inhalt auswirken.

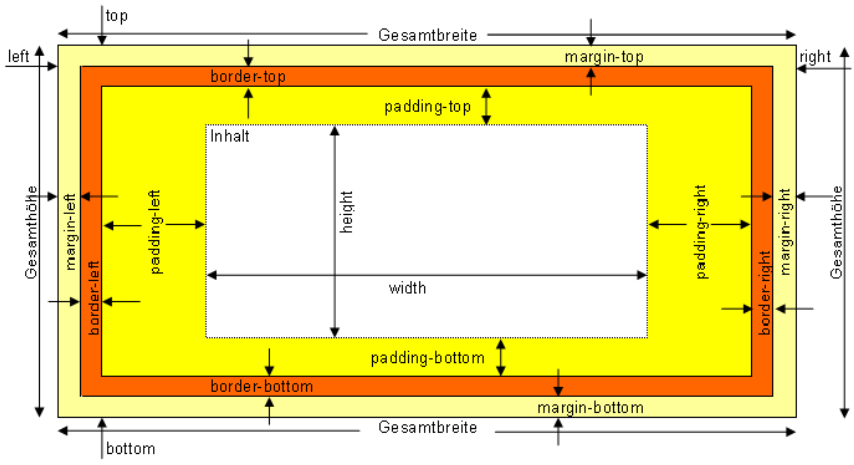
Das Boxmodell

Das Boxmodell heißt deshalb Boxmodell (auf Deutsch: Kasten-Modell), weil jedes Element einer Webseite durch einen rechteckigen Kasten beschrieben wird, in dem es angezeigt wird.

Für die Gestaltung von Webseiten ist neben der Position der Box auch die Größe des Inhaltsbereichs sehr wichtig. Sie hängt davon ab, ob für die Box eine Größe explizit festgelegt wurde oder nicht. Darüber hinaus spielt eine Rolle, ob die Box wiederum andere Boxen oder Tabellen enthält. Wenn die Größe einer Box nicht durch diese Faktoren bestimmt wurde, berechnet sie sich folgendermaßen:

Die Breite der Box berechnet sich aus der Summe der Außenabstände auf der linken und rechten Seite (`margin`) zuzüglich der Randstärke (`border`), der Breite des Inhalts und dem linken und rechten Innenabstand (`padding`). Die Formel lautet also:

Abbildung 5.1:
Das Boxmodell in der Übersicht



Breite = `margin-left` + `border-left-width` + `padding-left` + `width` (bzw. Breite des Inhalts) + `padding-right` + `border-right-width` + `margin-right`

Analog dazu wird die Höhe der Box nach der folgenden Formel berechnet:

Höhe = `margin-top` + `border-top-width` + `padding-top` + `height` (bzw. Höhe des Inhalts) + `padding-bottom` + `border-bottom-width` + `margin-bottom`

INFO

Auch wenn der Außenabstand (`margin`) mit zur Box gehört, wird er immer transparent dargestellt. Selbst wenn Sie also eine Hintergrundfarbe oder ein Hintergrundbild für das Element definieren, wird nur die Größe der Box abzüglich des Außenabstands mit dem Bild bzw. der Farbe gefüllt.

Das visuelle Formatierungsmodell von CSS

Webseiten, die auf Endlosmedien dargestellt werden, wie z.B. im Browser, sind in der Regel nicht komplett sichtbar. Wenn Sie beispielsweise eine Webseite erstellen und dafür CSS-Stile für die Bildschirmausgabe bestimmen, kann es abhängig vom Platzbedarf des Inhalts, der Fenstergröße des Browsers und der Bildschirmauflösung des Benutzers dazu kommen, dass nur Teile des HTML-Dokuments angezeigt werden. Dieser sichtbare Bereich ist der so genannte Viewport (auf Deutsch: Anzeigebereich).

Umschließender oder umgebender Block

Beim visuellen Formatierungsmodell erzeugt jedes Element entweder keinen Block, einen Block oder mehrere Blöcke, die wiederum Boxen gemäß dem Boxmodell enthalten. Sie werden innerhalb des Viewports dargestellt. Der Block der obersten Ebene ist der umschließende Block, der das gesamte (X)HTML-Dokument umfasst. Darüber hinaus hat aber jeder Block einen umgebenden bzw. umschließenden Block, nämlich der Block des Elements, dessen untergeordnetes Element es ist.

Ist die umschließende Box der obersten Ebene größer als der Viewport, soll der Browser gemäß CSS-Standard eine Möglichkeit zum Blättern zur Verfügung stellen. In nicht seitenorientierten Medien ist der Ausdruck »blättern« aber natürlich nicht so zu sehen, dass tatsächlich wie in einem Buch geblättert wird. Für die Bildschirmanzeige ermöglicht der Browser das Blättern in der Regel über seine Bildlaufleisten.

Der umschließende Block auf Dokumentebene, also der oberste Block der Hierarchie, wird auch als umschließender Ausgangsblock bezeichnet.

Die Größe des Viewports kann sich durch verschiedene Faktoren ändern, die Sie nicht zwingend per CSS oder durch Ihren HTML-Code beeinflussen können. Abhängig von der Definition Ihres HTML-Dokuments und Ihres CSS-Codes ändert der Browser bei Änderung des Viewports auch das Layout der Seite.

Der umschließende Ausgangsblock kann genau wie jeder andere Block untergeordnete Blöcke enthalten. Diese untergeordneten Blöcke werden auch als abgeleitete Blöcke oder abgeleitete Boxen bezeichnet.

Die Begriffe Block und Box stiften an dieser Stelle vielleicht etwas Verwirrung. Im Prinzip ist es aber gar nicht so schwer. Ein Block definiert den Inhalt der Box und deren Lage in der Hierarchie der Seite. Jeder Block definiert eine oder mehrere Boxen, die gemäß Boxmodell für die Darstellung des Blockinhalts sorgen. So gesehen können Sie beide Begriffe synonym verwenden, wenn Sie im Hinterkopf behalten, dass der Block noch nichts über die Darstellung der von ihm definierten Box aussagt.

Blöcke werden zudem nicht von jedem Element erzeugt, sondern nur von so genannten Blockelementen. Dazu gehören neben Überschriften auch Absätze. Inline-Elemente, wie `span` *oder* `a`*, erzeugen dagegen keinen Block, sondern nur eine zusätzliche Box innerhalb des umgebenden Blocks.*

Ein Beispiel soll dies verdeutlichen. Nehmen Sie an, Sie definieren mit `<h1>Überschrift</h1>` eine Überschrift erster Ordnung. Dann erzeugen Sie damit einen Block mit dem Inhalt »Überschrift«. Dieser Block wiederum definiert eine Box, die hier in Größe und Inhalt mit dem Block übereinstimmt. Grafisch lässt sich das wie in Abbildung 5.2 darstellen.

Durch Formatierung der Box können Sie es aber auch so einrichten, dass ein Block eine Box enthält, die nur Teile des Blocks ausfüllt oder auch aus dem Block herausläuft. Dieser Überhang wird als Überlauf bezeichnet.

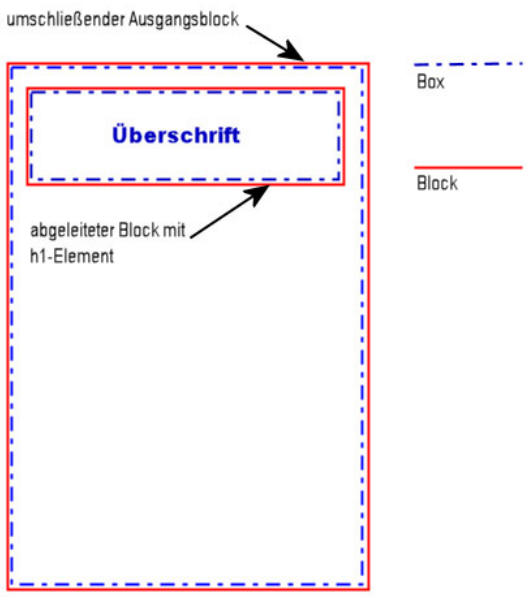


Abbildung 5.2: Verschachtelung von Blöcken und Boxen

Das könnte dann, bezogen auf das vorherige Beispiel, so aussehen:

```
<h1 style="border:1px solid silver;position:relative;left:-100px;
width:200px;height:50px">Überschrift</h1>
```

Die Überschrift wird hier relativ zu ihrem Block positioniert (`position:relative`) und dabei durch `left:-100px` nach links verschoben. Dadurch ragt sie nun links 100 Pixel aus dem umschließenden Block heraus. Da der Überlauf standardmäßig nicht sichtbar ist, sieht das Ergebnis im Browser wie in Abbildung 5.3 aus:

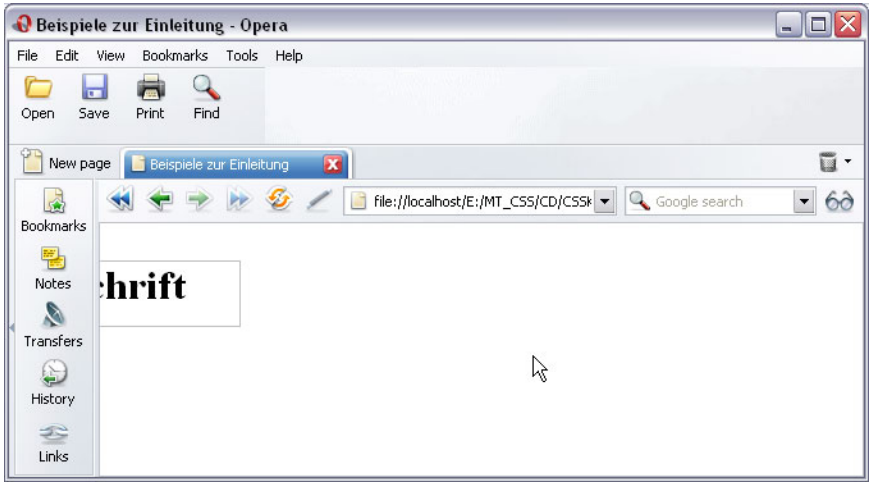


Abbildung 5.3: Die Box ragt nach links aus dem umgebenden Block heraus.

Der graue Rahmen, der oben mit `border:1px solid silver` *definiert wurde, dient hier nur dazu, die Größe und Position der Box kenntlich zu machen.*

Das Layout dieser Seite zeigt wiederum Abbildung 5.4. Sie sehen, dass die Größe der Box und deren Position eingehalten wird, jedoch nicht zwingend an die Lage und Größe des umgebenden Blocks gebunden ist. Die im Browser nicht mehr sichtbaren Teile der Box sind in der Grafik etwas verblasst dargestellt.

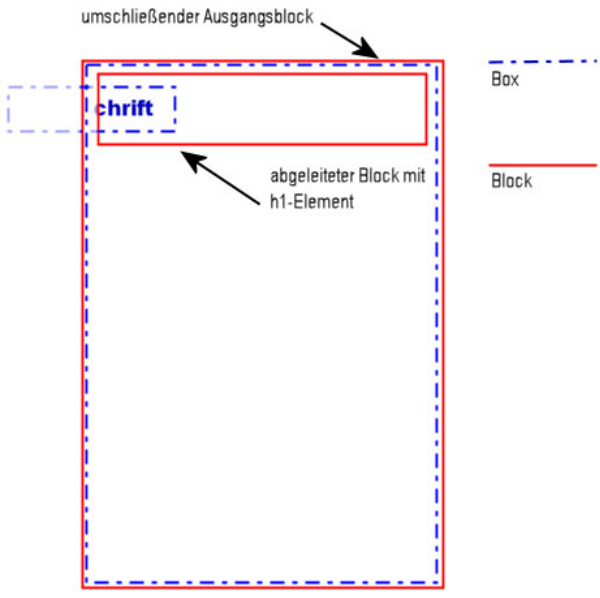


Abbildung 5.4: Hier ragt die Box aus dem umgebenden Block heraus.

Box-Typen

CSS kennt verschiedene Typen von Boxen. Abhängig vom Typ der Box verhält sich eine Box innerhalb des visuellen Formatierungsmodells unterschiedlich. Um welchen Typ Box es sich handelt, bestimmt die `display`-Eigenschaft.

Die Referenz zur `display`*-Eigenschaft finden Sie im Abschnitt »Anzeigeart (display)«.*

Elemente auf Blockebene

Als Elemente auf Blockebene werden alle (X)HTML-Blockelemente bezeichnet, die wie Absätze und Überschriften als eine visuelle Einheit angezeigt werden. Weitere Elemente können zu Blockelementen gemacht werden, indem Sie die `display`-Eigenschaft auf `block`, `list-item`, `compact` oder `run-in` setzen. Elemente auf Blockebene erzeugen eine Blockbox, die als Hauptblockbox bezeichnet wird. Sie definiert weitere abgeleitete Boxen für den Inhalt.

INFO

Blockelemente, die mehr als nur den definierten Inhalt enthalten, wie z.B Listeneinträge, bei denen zusätzlich auch ein Aufzählungszeichen erzeugt wird, erzeugen neben der Hauptblockbox zusätzliche Boxen, die relativ zur Hauptblockbox positioniert werden.

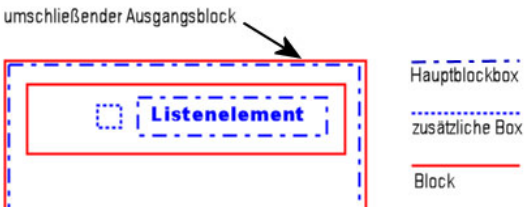


Abbildung 5.5: Aufbau von Blockelementen mit zusätzlichen Blöcken

Inline-Elemente

Als Inline-Elemente, einzeilige Elemente oder Elemente auf Inline-Ebene werden solche Elemente des (X)HTML-Dokuments bezeichnet und behandelt, die keinen neuen Block bilden, sondern lediglich eine Box innerhalb eines Blocks erzeugen. Dazu gehören beispielsweise Elemente für die Textauszeichnung wie das `b`-Element, aber auch Hyperlinks (`a`), Betonungen (`em`) oder Code (`code`) sowie Bilder. Weiterhin werden als Inline-Elemente solche Elemente behandelt, deren `display`-Eigenschaft die Werte `inline`, `inline-table`, `compact` oder `run-in` hat. Inline-Elemente erzeugen immer auch Inline-Boxen.

Wenn Sie z.B. folgenden Code in Ihrer (X)HTML-Seite definiert haben, erzeugen Sie damit zunächst mit dem Blockelement `h1` einen Block und einen weiteren Block mit dem `p`-Element. Da Blöcke immer einen Absatz bilden, werden sie untereinander angezeigt, wie Sie der Abbildung 5.6 entnehmen können.

```
<h1>Display-Eigenschaft</h1>
<p>Die Display-Eigenschaft bestimmt die Anzeigeart eines
   Elements.</p>.
```

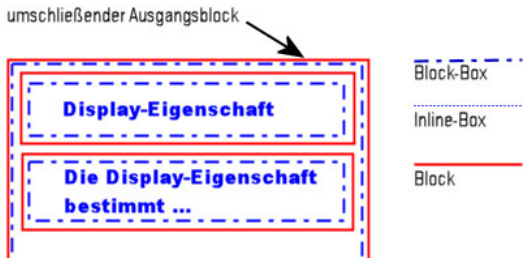


Abbildung 5.6: Anordnung von Blockboxen und Blockelementen im umgebenden Block

Anders sieht es hingegen aus, wenn Sie innerhalb der Blockelemente Inline-Elemente verwenden, indem Sie beispielsweise Teile des Inhalts in ein `code`-Element einfassen.

```
<h1><code>Display</code>-Eigenschaft</h1>
<p>Die <code>Display</code>-Eigenschaft bestimmt die Anzeigeart
eines Elements.</p>.
```

In diesem Fall werden die Inhalte beider Blockelemente in Inline-Boxen aufgeteilt, die dann nebeneinander und nicht wie Blockboxen untereinander dargestellt werden.

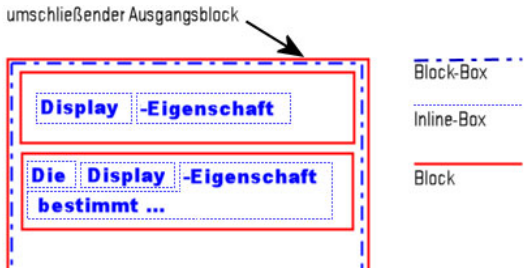


Abbildung 5.7: Aus einer einzigen Blockbox werden nun mehrere Inline-Boxen.

Im Beispiel entsteht durch die Formatierung mit dem `code`*-Element in jedem Blockelement eine benannte Inline-Box »code«, da sie durch das Inline-Element* `code` *erzeugt wird. Der umgebende Text des* `p`*- bzw.* `h1`*-Elements, für das der Autor kein Inline-Element spezifiziert hat, wird zu einer anonymen Inline-Box. Anonyme Inline-Boxen erben ihre Formatierungen von der übergeordneten Blockbox.*

Absolute und relative Positionierung

Reihenfolge und Schachtelung der Boxen legen fest, an welcher Stelle des Viewports ein Element angezeigt wird. Dabei gibt es zwei grundlegende Regeln:

- Blockboxen werden immer untereinander angezeigt und zwar in der Reihenfolge, in der sie im Code der (X)HTML-Seite definiert sind.
- Inline-Boxen werden hingegen immer nebeneinander angezeigt. Bei Bedarf und abhängig von der Formatierung stellt der Browser sie dann auch mehrzeilig dar.

Mit Hilfe von Angaben zur Positionierung von Elementen können Sie diese Reihenfolge in der Darstellung ändern und bei absoluter Positionierung auch ganz exakt die Position innerhalb des umgebenden Blocks festlegen, an der ein Element dargestellt wird.

Die Art der Positionierung legen Sie über die Eigenschaft `position` fest. Ihr Wert bestimmt unter anderem, wie die Werte der Eigenschaften `left`, `top`, `bottom` und `right` interpretiert werden. Sie legen die Position der oberen linken Ecke der Box fest.

Bei relativer Positionierung werden deren Werte relativ zur Position der Box interpretiert, die diese ohne Positionierung einnehmen würde. Bei absoluter Positionierung werden die Werte als absolute Positionsangaben ausgehend vom umgebenden Block interpretiert.

Position im normalen Fluss

Genau genommen passiert bei der relativen Positionierung Folgendes: Der Browser berechnet die normale Position des Elements, die sich aus der Reihenfolge der Elemente in der (X)HTML-Seite und aus deren Formatierungen wie Größe und Inhalte berechnet. Die so berechnete Position wird als Position im normalen Fluss bezeichnet.

Ausgehend von dieser Position wird das Element dann gemäß der Angaben zur relativen Positionierung verschoben. Wurde als linke Position beispielsweise der Wert 20px angegeben, wird die Box um 20 Pixel nach rechts verschoben. Alle nachfolgenden Elemente werden so positioniert, als wäre das relativ positionierte Element nicht verschoben worden.

5.2 Praxistaugliche Attribute

Nachfolgend finden Sie CSS-Attribute, die Sie problemlos oder mit nur geringen Einschränkungen in der Praxis einsetzen können. Entweder ist die Browserunterstützung sehr gut oder aber ein Nichtausführen der CSS-Anweisung wirkt sich nicht so negativ aus, dass die Seite nicht mehr bedienbar ist.

Anzeigeart (display)

Die `display`-Eigenschaft legt fest, wie das Element angezeigt werden soll (z.B. als Inline- oder Blockelement oder als Tabelle). Standardkonforme Browser dürfen die `display`-Eigenschaft ignorieren.

CSS-Element: `display`	*Mögliche Werte: inline*, `block`, `list-item`, `none`, `inherit`[1], `run-in`[2], `compact`[3+4], `marker`[5], `table`[2+4], `inline-table`[2+4], `table-row-group`[2+4], `table-header-group`[2+4], `table-footer-group`[2+4], `table-row`[2+4], `table-column-group`[2+4], `table-column`[2+4], `table-cell`[2+4], `table-caption`[2+4]
CSS-Version: CSS 1-3, Mobile, TV	*Zulässig für folgende (X)HTML-Elemente:* alle
Medium: alle	*Vererbt:* Nein

Palm	NN	Mozilla		Internet Explorer							Opera			Safari		iCab	Kq	FF
2.x	4.x	1.x	1.8	4.x	5.0x (Win)	5.0 (Mac)	5.1+ (Mac)	5.5 (Win)	6	7	6	7	8	1.x	2.0	2.9+	3.x	1.0
	○ 7	○ 7	○ 7	○ 7	○ 7	○ 7	○ 7	○ 7	○ 7	○ 7	○ 7	○ 7	○ 7	○ 7	○ 7	○ 7+6	○ 7	○ 7

1 nicht in Mobile-Spezifikation
2 nicht in Mobile- und TV-Spezifikation
3 nicht in CSS 2.1
4 erst ab CSS 2.x
5 nur in CSS 2.0
6 ab iCab 3.0
7 siehe nähere Erläuterungen im Anschluss

Sie finden das Beispiel auf der Buch-CD in der Datei BSP/K05/display.html.

Die verfügbaren Werte der Eigenschaften lassen sich grob in drei Gruppen einteilen: in allgemeine Werte, Werte für Elemente, die Inhalte erzeugen sowie tabellenspezifische Werte. In dieser Gruppierung werden die Werte nachfolgend auch erläutert.

Allgemeine Werte

none Mit dem Wert `none` können Sie festlegen, dass ein Element nicht angezeigt werden soll. Im Unterschied zur CSS-Eigenschaft `visibility:hidden` wird das Element aber nicht nur nicht angezeigt, es wird auch keine Box erzeugt, so dass kein Platz im Seitenlayout für das Element reserviert wird.

CSS-Element: `display:none`

CSS-Version: CSS 1-3, Mobile, TV

Zulässig für folgende (X)HTML-Elemente: alle

Medium: alle

Vererbt: Nein

Palm	NN	Mozilla		Internet Explorer							Opera			Safari		iCab	Kq	FF
2.x	4.x	1.x	1.8	4.x	5.0x (Win)	5.0 (Mac)	5.1+ (Mac)	5.5 (Win)	6	7	6	7	8	1.x	2.0	2.9+	3.x	1.0
●	○	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●

block Bei dem Wert `block` erzeugt der Browser für das Element eine Hauptblockbox.

CSS-Element: `display:block`

CSS-Version: CSS 1-3, Mobile, TV

Zulässig für folgende (X)HTML-Elemente: alle

Medium: alle

Vererbt: Nein

Palm	NN	Mozilla		Internet Explorer							Opera			Safari		iCab	Kq	FF
2.x	4.x	1.x	1.8	4.x	5.0x (Win)	5.0 (Mac)	5.1+ (Mac)	5.5 (Win)	6	7	6	7	8	1.x	2.0	2.9+	3.x	1.0
●	○	●	●		●	● [1]	● [1]	●	●	●	●	●	●	●	●	● [2]	●	●

[1] Führende Leerzeichen im folgenden Blockelement bleiben erhalten.
[2] ab Version 3.0

Demonstrieren lässt sich das beispielsweise an einem typischen Inline-Element, wie `b`. Definieren Sie folgenden Stil,

```
b { display:block }
```

erscheint eine damit formatierte Inline-Box wie ein Blockelement als eigener Absatz. Mit dem Code:

```
<div>Falls <code>display:block</code> ausgeführt wird, erscheint
ein <b>fett</b> formatierter Text als eigener Absatz, nicht als
Inline-Element</div>
```

wird dann die Ausgabe in Abbildung 5.8 erzeugt.

Abbildung 5.8: Korrekte Darstellung des Wertes block in Firefox

Der Netscape Navigator 4.7 erzeugt zwar einen Umbruch vor dem Element, nicht jedoch danach.

Abbildung 5.9: Im Netscape Navigator 4.x fehlt der Umbruch nach dem b-Element.


Mac

Der Internet Explorer für Macintosh zeigt das mit `display:block` formatierte Element im Beispiel zwar als Blockelement an. Führende Leerzeichen des nachfolgenden Textes werden aber, anders als bei anderen Browsern und der Windows-Version, nicht abgeschnitten.

inline

Mit diesem Wert können Sie dafür sorgen, dass ein Element als Inline-Element angezeigt wird. Angewendet auf ein typisches Blockelement wie `p` dürfte der Text des Elements nicht mehr als eigener Absatz angezeigt werden.

CSS-Element: `display:inline`

CSS-Version: CSS 1-3, Mobile, TV

Zulässig für folgende (X)HTML-Elemente: alle

Medium: alle

Vererbt: Nein

Palm	NN	Mozilla		Internet Explorer							Opera			Safari		iCab	Kq	FF
2.x	4.x	1.x	1.8	4.x	5.0x (Win)	5.0 (Mac)	5.1+ (Mac)	5.5 (Win)	6	7	6	7	8	1.x	2.0	2.9+	3.x	1.0
●		●	●		●	●	●	●	●	●	●	●	●	●	●	●[1]	●	●

[1] ab Version 3.0

Erstellen Sie beispielsweise den Stil

```
.inline { display:inline }
```

und wenden ihn mit

Listing 5.1: Beispiel für den Wert inline

```
<div>
    <p class="inline">Falls <code>display:inline</code>
    ausgeführt wird, </p>
    <p class="inline">sollte dieser Text als </p>
    <p class="inline">ein Absatz angezeigt werden.</p>
</div>
```

an, sollte der gesamte Inhalt des `div`-Elements als ein Absatz angezeigt werden, da jeder Absatz, der eigentlich ein Blockelement ist, als Inline-Element formatiert wird.

list-item

Das mit `display:list-item` formatierte Element wird wie ein Listenelement angezeigt. Das bedeutet, neben der Hauptblockbox wird eine zusätzliche Box für das Listenzeichen als Inline-Box erzeugt.

Abbildung 5.10: Firefox stellt den Wert inline korrekt dar.

CSS-Element: `display:list-item`

CSS-Version: CSS 1-3, Mobile, TV

Zulässig für folgende (X)HTML-Elemente: alle

Medium: alle

Vererbt: Nein

Palm	NN	Mozilla		Internet Explorer							Opera			Safari		iCab	Kq	FF
2.x	4.x	1.x	1.8	4.x	5.0x (Win)	5.0 (Mac)	5.1+ (Mac)	5.5 (Win)	6	7	6	7	8	1.x	2.0	2.9+	3.x	1.0
●		●	●		●	●[1]	●[1]	●	●	●	●	●	●	●	●	●[2]	●	●

1 Als Listenzeichen wird eine Zahl verwendet.
2 ab Version 3.0

Wenn Sie beispielsweise den Stil

```
.liste { display:list-item }
```

auf einen normalen Absatz anwenden, wird dieser mit einem Listenzeichen versehen.

Das Listenzeichen wird aber außerhalb der Box angezeigt. Da aber, anders als bei wirklichen Listeneinträgen, der automatische Einzug fehlt, kann es vorkommen, dass wie im Beispiel das Listenzeichen nicht mehr komplett sichtbar ist, weil es nach links über den Viewport hinausragt. Damit es sichtbar ist, müssen Sie für den Absatz zusätzlich einen größeren linken Außenabstand festlegen, beispielsweise indem Sie den Stil folgendermaßen definieren:

:-) TIPP

```
.liste { display:list-item; margin-left:20px; ...}
```

Mac

Abbildung 5.11: Während der Internet Explorer für Mac ein numerisches Listenzeichen verwendet, zeigen die anderen Browser einen Punkt als Listenzeichen an.

inherit

Beim Wert `inherit` erbt das Element den Wert der `diplay`-Eigenschaft des umgebenden Elements. Ist das umgebende Element ein Blockelement, müsste also auch ein Inline-Element wie `span` als Block dargestellt werden.

CSS-Element: `display:inherit`	
CSS-Version: CSS 1-3, TV	*Zulässig für folgende (X)HTML-Elemente:* alle
Medium: alle	*Vererbt:* Nein

Palm	NN	Mozilla		Internet Explorer							Opera			Safari		iCab	Kq	FF
2.x	4.x	1.x	1.8	4.x	5.0x (Win)	5.0 (Mac)	5.1+ (Mac)	5.5 (Win)	6	7	6	7	8	1.x	2.0	2.9+	3.x	1.0
●		●	●			●	●				●	●	●	●	●	●[1]	●	●

[1] ab Version 3.0

Den Wert können Sie mit folgendem Beispiel testen. Sie erstellen ein Blockelement wie `div` oder `p` und fügen darin neben dem Text des Blockelements ein Inline-Element wie `span` ein.

```
<div>Falls <code>display:inherit</code> <span
class="inherit">ausgeführt wird, sollte dieser Text in zwei
Absätzen angezeigt werden.</span></div>
```

Für dieses Element definieren Sie dann den Stil:

```
.inherit { display:inherit }
```

Bei korrekter Darstellung des Wertes müsste der Browser nun vor dem `span`-Element einen neuen Block erzeugen, da das `span`-Element nun `display:block` vom umgebenden Blockelement erbt.

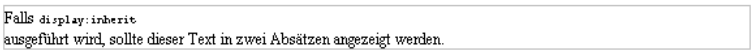


Abbildung 5.12: So sollte das korrekte Ergebnis aussehen.

Erzeugte Inhalte

Inhalte können Sie mit CSS mit verschiedenen CSS-Pseudo-Elementen (`:before` und `:after`) erzeugen. Sie erstellen damit Text oder andere Inhalte vor oder nach einem Element. Wie dieser erzeugte Inhalt angezeigt wird, regeln Sie über die Werte `marker`, `run-in` und `compact` für die `display`-Eigenschaft.

marker

Dieser Wert formatiert erzeugten Inhalt vor oder hinter einer Box als Markierung. Der Browser sollte den Text ähnlich wie ein Aufzählungszeichen darstellen.

CSS-Element: `display:marker`	
CSS-Version: CSS 2	*Zulässig für folgende (X)HTML-Elemente:* erzeugter Text innerhalb eines Blockelements
Medium: alle	*Vererbt:* Nein

Palm	NN	Mozilla		Internet Explorer							Opera			Safari		iCab	Kq	FF
2.x	4.x	1.x	1.8	4.x	5.0x (Win)	5.0 (Mac)	5.1+ (Mac)	5.5 (Win)	6	7	6	7	8	1.x	2.0	2.9+	3.x	1.0

Der folgende Stil wendet den Wert `marker` an. Dabei wird dem Element, das mit dem Stil `marker` formatiert wird, ein Pluszeichen vorangestellt, das dann wie ein Listenzeichen formatiert werden sollte.

```
.marker:before { display:marker; content:"+" }
```

run-in und compact

Beide Werte, `run-in` und `compact`, schieben den damit formatierten Inhalt als erste Zeile in das nächste Element. Allerdings passiert das nur dann, wenn dieses weder mit `float` noch mit `position:absolute` formatiert ist.

Der Wert `compact` *steht ausschließlich in CSS 2.0 und 3.0 zur Verfügung, nicht in CSS 2.1.*

CSS-Element: `display:compact` bzw. `display:run-in`

CSS-Version: CSS 2.0 + CSS 3.0

Zulässig für folgende (X)HTML-Elemente: alle

Medium: alle

Vererbt: Nein

Palm	NN	Mozilla		Internet Explorer							Opera			Safari		iCab	Kq	FF
2.x	4.x	1.x	1.8	4.x	5.0x (Win)	5.0 (Mac)	5.1+ (Mac)	5.5 (Win)	6	7	6	7	8	1.x	2.0	2.9+	3.x	1.0
○						● [1]	● [1]				● [1]	● [1]	● [1]		● [1]		● [1]	

[1] nur der Wert `run-in`

Um den Wert zu testen, benötigen Sie zunächst einen Stil, der den Wert einem Element zuweist:

```
.compact { display:compact }
```

bzw.

```
.runin { display:run-in }
```

Diesen Stil weisen Sie einem Element zu, dem beispielsweise ein `p`-Element folgt, das nicht positioniert ist.

Listing 5.2: Anwenden der Stile

```
<div>Falls <code>display:compact</code> ausgeführt wird, sollte der
   Text
    "TEST" als erste Zeile des folgenden Absatzes eingefügt werden. Zur
    Kontrolle wird der Folgeabsatz in roter Schrift formatiert.
    <p class="compact">TEST</p>
    <p style="color:red">Dies ist der Folgeabsatz.</p>
</div>
<div>Falls <code>display:run-in</code> ausgeführt wird, sollte der Text
    "TEST" als erste Zeile des folgenden Absatzes eingefügt werden. Zur
    Kontrolle wird der Folgeabsatz in roter Schrift formatiert.
    <p class="runin">TEST</p>
    <p style="color:red">Dies ist der Folgeabsatz.</p>
</div>
```

Der Palm-Browser wendet zwar beide Werte an, dafür aber fehlerhaft. Statt den Inhalt des formatierten Elements als erste Zeile des folgenden Absatzes einzufügen, wird er an das vorstehende Element angehängt.

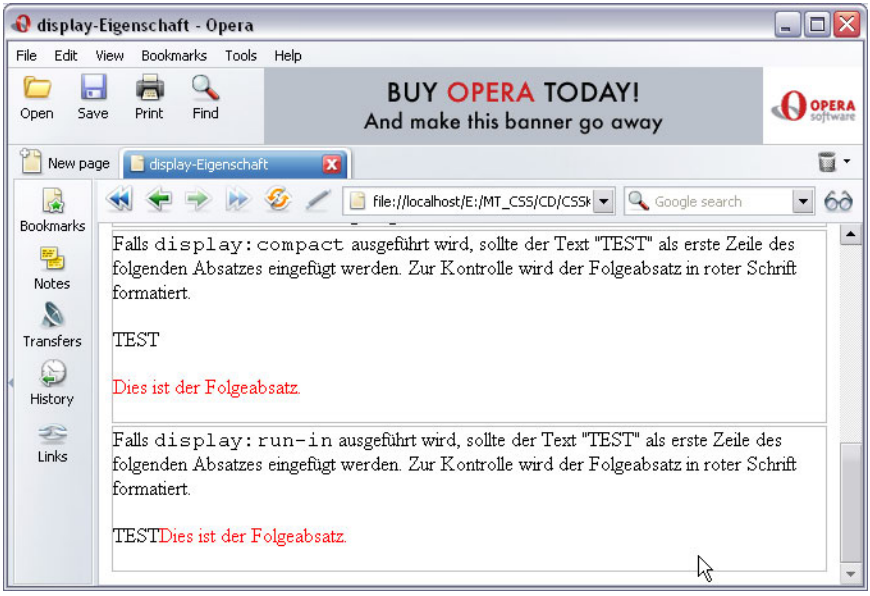


Abbildung 5.13: Korrekte Darstellung des Wertes run-in (unten) und fehlende Darstellung des Wertes compact

Werte für Tabellen

Die dritte Gruppe der Werte umfasst alle Werte, die mit `table` beginnen. Alle diese Werte sollten dazu führen, dass sich das damit formatierte Element wie das durch den Wert definierte Tabellenteil verhält und entsprechend formatiert wird. Geben Sie beispielsweise `table-cell` an, sollte das Element wie eine Tabellenzelle formatiert werden.

Sie finden die Beispiele zu den Tabellenwerten auf der Buch-CD in der Datei BSP/K05/displaytable.html.

table, table-row und table-cell

Ein mit dem Wert `table` formatiertes Element wird wie eine Tabelle dargestellt und behandelt; analog dazu stellen die Werte `table-cell` und `table-row` das Element wie eine Tabellenzelle bzw. eine Tabellenzeile dar.

Geben Sie für das umgebende Element den Wert `table` *und für dessen untergeordneten Elemente* `table-cell` *an, werden alle untergeordneten Elemente in einer Zeile dargestellt. Das passiert auch dann, wenn sich dazwischen Zeilenumbrüche mit* `br` *befinden. Für eine wirklich tabellarische Darstellung müssen Sie daher auch den Wert* `table-row` *verwenden und damit die Zeilen festlegen.*

CSS-Element: `display:table`

CSS-Version: CSS 2/3

Zulässig für folgende (X)HTML-Elemente: alle

Medium: alle

Vererbt: Nein

Palm	NN	Mozilla		Internet Explorer							Opera			Safari		iCab	Kq	FF
2.x	4.x	1.x	1.8	4.x	5.0x (Win)	5.0 (Mac)	5.1+ (Mac)	5.5 (Win)	6	7	6	7	8	1.x	2.0	2.9+	3.x	1.0
○		●	●			● [1]	● [1]				● [2]	● [2]	● [2]		●		●	●

[1] Die Darstellung des Wertes `table` ist zwar korrekt, die Werte `table-row` und `table-cell` werden jedoch nicht bzw. nicht korrekt angewendet, so dass auch der Wert `table` unbrauchbar ist.

[2] Verschiedene HTML-Strukturen können in Verbindung mit dem Wert `table` und dem Wert `table-row` zu Abstürzen des Browsers führen.

CSS-Element: `display:table-cell`

CSS-Version: CSS 2/3

Zulässig für folgende (X)HTML-Elemente: alle

Medium: alle

Vererbt: Nein

Palm	NN	Mozilla		Internet Explorer							Opera			Safari		iCab	Kq	FF
2.x	4.x	1.x	1.8	4.x	5.0x (Win)	5.0 (Mac)	5.1+ (Mac)	5.5 (Win)	6	7	6	7	8	1.x	2.0	2.9+	3.x	1.0
○		●	●								●	●	●		●		●	●

CSS-Element: `display:table-row`

CSS-Version: CSS 2/3

Zulässig für folgende (X)HTML-Elemente: alle

Medium: alle

Vererbt: Nein

Palm	NN	Mozilla		Internet Explorer							Opera			Safari		iCab	Kq	FF
2.x	4.x	1.x	1.8	4.x	5.0x (Win)	5.0 (Mac)	5.1+ (Mac)	5.5 (Win)	6	7	6	7	8	1.x	2.0	2.9+	3.x	1.0
○		●	●								●[1]	●[1]	●[1]		●		●	●

[1] Verschiedene HTML-Strukturen können in Verbindung mit dem Wert `table` und dem Wert `table-row` zu Abstürzen des Browsers führen.

Der Palm-Browser stellt den Wert gleich mehrfach falsch dar. Definieren Sie lediglich den Stil `div {display:table; border: 1px solid silver}` und erzeugen wie im folgenden Beispiel dann ein `div`-Element, das Absätze mit `span`-Element enthält, ignoriert der Browser die Absätze komplett, zeigt alle Werte in einer Zeile an und behandelt das `div`-Element wie ein Inline-Element. Kombiniert man `table` mit `table-row` und `table-cell`, werden teilweise die Absätze wieder korrekt angezeigt, dafür aber der umgebende Rahmen für das `div`-Element nicht mehr. Im Prinzip ist die Darstellung in diesem Browser damit absolut praxisuntauglich.

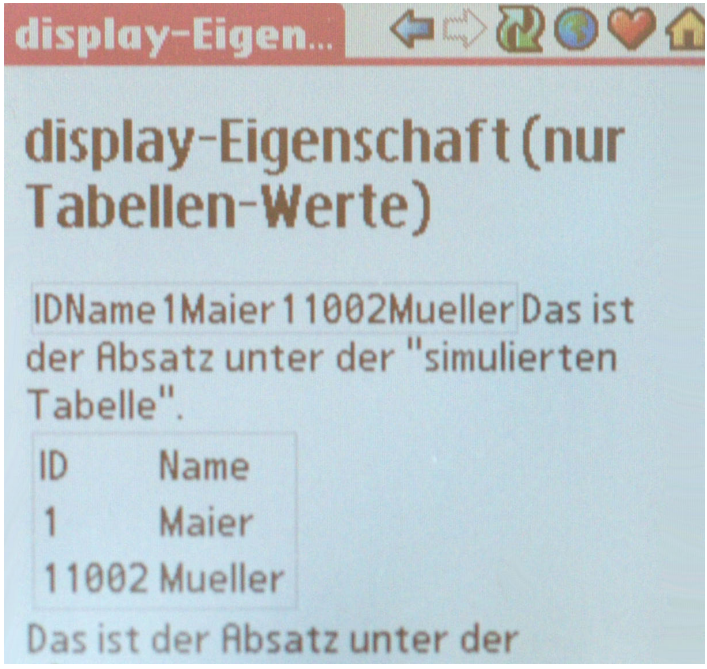


Abbildung 5.14: Fehlerhafte Darstellung der Beispielseite im Palm-Browser

Am besten lassen sich die Werte `table`, `table-row` und `table-cell` gemeinsam darstellen. Der grundlegende Aufbau der Beispielseite sieht dazu folgendermaßen aus. Sie enthält zunächst ein `div`-Element, dem die CSS-Klasse `tabelle` zugeordnet ist. Dieses Element soll mittels CSS so angezeigt werden, als wäre es eine Tabelle. Das heißt, die darin enthaltenen Daten sollten in Zeilen und Spalten angeordnet werden. Der HTML-Code definiert die späteren Tabellenzeilen als Absätze mit `p`-Elementen und die darin enthaltenen späteren Zellinhalte werden in `span`-Elemente gesetzt.

Nach dem `div`-Element folgt ein normaler Absatz, der zeigen soll, ob das `div`-Element als Block- oder Inline-Element angezeigt wird. Darunter wird zum Vergleich eine Tabelle mit den gleichen Daten und ebenfalls einem folgenden Absatz definiert.

Listing 5.3: Aufbau der HTML-Seite

```
<body>
    <h1>display-Eigenschaft (nur Tabellen-Werte)</h1>
    <div class="tabelle">
        <p><span>ID</span><span>Name</span></p>
        <p><span>1</span><span>Maier</span></p>
        <p><span>11002</span><span>Mueller</span></p>
    </div>
    <p>Das ist der Absatz unter der "simulierten Tabelle".</p>
    <table>
        <tr><td>ID</td><td>Name</td></tr>
        <tr><td>1</td><td>Maier</td></tr>
        <tr><td>11002</td><td>Mueller</td></tr>
    </table>
    <p>Das ist der Absatz unter der Vergleichstabelle.</p>
</body>
```

Damit nun der obere `div`-Tag wie eine Tabelle angezeigt wird, sind natürlich noch ein paar Stile erforderlich: Zunächst legt der erste Selektor die Umrandung und den Abstand für das `div`- und `table`-Element fest. Die Formatierungen sind für das Beispiel nicht erforderlich, aber ganz nützlich, damit Sie sehen, wo `div`- und `table`-Elemente beginnen und enden.

Der Klassenstil `.tabelle` definiert, dass das umgebende `div`-Element als Tabelle angezeigt werden soll. Mit dem Elementstil für die `span`-Elemente bestimmen Sie, dass diese als Tabellenzellen angezeigt werden. Um einen Umbruch in mehrere Zeilen zu erreichen, wird im Stil für die enthaltenen `p`-Elemente `display:table-row` angegeben. Jeder Absatz stellt damit eine Tabellenzeile dar.

Da im Beispiel außerhalb des `div`*-Elements ebenfalls* `p`*-Elemente stehen, wird hier der Selektor* `.tabelle p` *verwendet, der nur die* `p`*-Elemente auswählt, die sich innerhalb des Elements mit der CSS-Klasse* `.tabelle` *befinden.*

```
<style type="text/css">
<!--
    div,table {
         border: 1px solid silver;
         margin:2px
    }
    .tabelle   { display:table }
    span       { display:table-cell; width:50% }
    .tabelle p { display:table-row }
-->
</style>
```

Listing 5.4: Die erforderlichen Stile

Abbildung 5.15: In etwa so sollte der Browser das Beispiel darstellen.

Für den Opera-Browser ist es offenbar ausgesprochen wichtig, dass Sie als Elemente, die Sie mit `display:table-row` definieren, nur solche Elemente auswählen, die im HTML-Code die als Zellen formatierten Elemente umschließen. Würden Sie beispielsweise die `p`-Elemente aus dem Code entfernen und zwischen den `span`-Elementen `br`-Elemente einfügen, um das Zeilenende zu markieren, würde das mit dem entsprechenden Stil für das `br`-Element dazu führen, dass der Opera-Browser bei der Anzeige der Seite abstürzt.

Sicherlich muss man dazu sagen, dass es auch logisch ist, dass nur ein umschließendes Element als Tabellenzeile formatiert werden kann. Dennoch ist sowohl der HTML- wie der CSS-Code korrekt. Offenbar ist also nur die Kombination das Problem.

Listing 5.5: Dieser Code führt zu einem Absturz des Opera-Browsers.

```
<style type="text/css">
<!--
    div,table { border: 1px solid silver; margin:2px }
    .tabelle  { display:table }
    span      { display:table-cell; width:50% }
    br        { display:table-row }
-->
</style>
</head>
<body>
    <h1>display-Eigenschaft (nur Tabellen-Werte)</h1>
    ...
    <!-- der folgende Code lässt Opera abstürzen -->
    <div class="tabelle">
        <span>ID</span><span>Name</span><br>
        <span>1</span><span>Maier</span><br>
        <span>11002</span><span>Mueller</span>
    </div>
...
```

table-column

Der Wert `table-column` bewirkt, dass das Element dargestellt wird wie die mit dem Element `col` definierten Spalten einer Tabelle. Die Browserunterstützung ist allerdings eher mangelhaft.

CSS-Element: `display:table-column`

CSS-Version: CSS 2/3 — *Zulässig für folgende (X)HTML-Elemente:* alle

Medium: alle — *Vererbt:* Nein

Palm	NN	Mozilla		Internet Explorer							Opera			Safari		iCab	Kq	FF
2.x	4.x	1.x	1.8	4.x	5.0x (Win)	5.0 (Mac)	5.1+ (Mac)	5.5 (Win)	6	7	6	7	8	1.x	2.0	2.9+	3.x	1.0
●																		

table-caption, und table-header-group, table-footer-group

Analog zu den vorstehend genannten Werten funktionieren auch die Werte `table-caption`, `table-footer-group` und `table-header-group`. Elemente, die Sie mit diesen Werten formatieren, werden als Tabellenbeschriftung (`table-caption`), Spaltenüberschriften (`table-header-group`) und Spaltenunterschriften (`table-footer-group`) angezeigt.

table-caption

Die Tabellenbeschriftung muss gemäß HTML-Standard entweder über oder unter der Tabelle angezeigt werden. Standardmäßig zeigen die meisten Browser die mit dem `caption`-Element definierte Tabellenbeschriftung über der Tabelle an, also auch oberhalb der Spaltenüberschriften. Sie erstreckt sich dabei immer über die volle Breite der Tabelle.

CSS-Element: `display:table-caption`

CSS-Version: CSS 2/3

Zulässig für folgende (X)HTML-Elemente: alle

Medium: alle

Vererbt: Nein

Palm	NN	Mozilla		Internet Explorer							Opera			Safari		iCab	Kq	FF
2.x	4.x	1.x	1.8	4.x	5.0x (Win)	5.0 (Mac)	5.1+ (Mac)	5.5 (Win)	6	7	6	7	8	1.x	2.0	2.9+	3.x	1.0
○		●	●								●	●	●		●		●	●

Beim folgenden HTML-Code, in dem die Texte für Spaltenüberschriften, Zellinhalte und Tabellenbeschriftung bewusst völlig durcheinander definiert wurden, müsste ein korrekt arbeitender Browser die Tabellenbeschriftung in dem Absatz mit der CSS-Klasse `.titel` also oberhalb der Spaltenbeschriftungen anzeigen.

Listing 5.6: Aufbau des div-Elements zum Testen der Werte

```
<div class="tabelle2">
    <p><span>1</span><span>Maier</span></p>
    <p class="titel">Dieser Titel sollte über den
        Spaltenüberschriften angezeigt werden.</p>
    <p><span>11002</span><span>Mueller</span></p>
    <p class="spaltentitel"><span>ID</span><span>Name</span></p>
</div>
<p>Das ist der Absatz unter der Tabelle</p>
```

Um dafür zu sorgen, dass der Absatz mit der CSS-Klasse `titel` als Tabellenüberschrift angezeigt wird, benötigen Sie zwei Stile. Zunächst müssen Sie angeben, dass das `div`-Element mit der CSS-Klasse `tabelle2` als Tabelle angezeigt werden soll. Dann können Sie mit dem zweiten Stil definieren, dass die CSS-Klasse `.titel` den Tabellentitel darstellt.

```
.tabelle2        { display:table; width:200px }
.tabelle2 .titel { display:table-caption;
                   text-align:center }
```

Die meisten Browser zeigen eine Tabellenüberschrift in der Regel mit zentrierter Textausrichtung an. Das passiert nicht automatisch, wenn Sie ein Element mit `display:table-caption` *definieren. Möchten Sie die Tabellenbeschriftung also ebenfalls zentrieren, müssen Sie das separat mit* `text-align:center` *definieren.*

table-header-group

Die mit diesem Wert formatierten Elemente werden wie **thead**-Elemente angezeigt. Das heißt, sie stehen vor der ersten normalen Tabellenzeile, aber gegebenenfalls nach der Tabellenüberschrift.

CSS-Element:
`display:table-header-group`

CSS-Version: CSS 2/3

Zulässig für folgende (X)HTML-Elemente: alle

Medium: alle

Vererbt: Nein

Palm	NN	Mozilla		Internet Explorer							Opera			Safari		iCab	Kq	FF
2.x	4.x	1.x	1.8	4.x	5.0x (Win)	5.0 (Mac)	5.1+ (Mac)	5.5 (Win)	6	7	6	7	8	1.x	2.0	2.9+	3.x	1.0
○		●	●								●	●	●		●		●	●

Wenn Sie im obigen HTML-Code (siehe Listing 5.6) das Element mit der CSS-Klasse `.spaltentitel` als Spaltenbeschriftungen festlegen möchten, benötigen Sie dazu die folgenden Stile. Auch hier gilt, dass der Wert für die `display`-Eigenschaft nur die korrekte Position und die erzeugten Boxen bestimmt. Wenn Sie, wie bei `th`-Elementen üblich, auch eine fette und zentrierte Formatierung der Spaltenüberschriften erreichen möchten, müssen Sie das wieder explizit angeben.

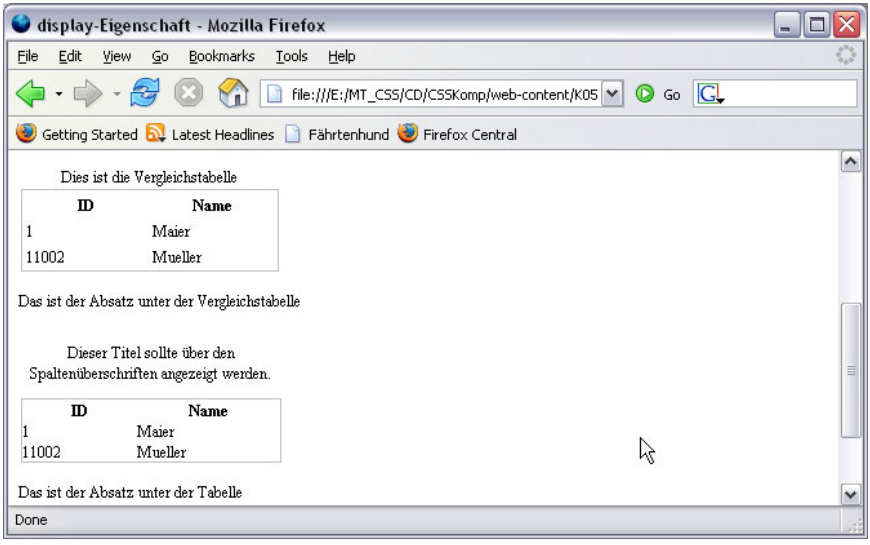


Abbildung 5.16: Oben wird eine normale, per HTML aufgebaute Tabelle zum Vergleich angezeigt, unten die korrekt per CSS simulierte Tabelle.

```
.tabelle2 .spaltentitel {
    display:table-header-group;
    text-align:center;
    font-weight:bold
}
.tabelle2 span {
    display:table-cell
}
```

Listing 5.7: Notwendige Stile zum Formatieren der Spaltenüberschriften

table-footer-group

Die mit diesem Wert formatierten Elemente werden wie **tfoot**-Elemente angezeigt. Das heißt, sie stehen nach der letzten normalen Tabellenzeile, aber gegebenenfalls über der Tabellenunterschrift.

CSS-Element:
`display:table-footer-group`

CSS-Version: CSS 2/3

Zulässig für folgende (X)HTML-Elemente: alle

Medium: alle

Vererbt: Nein

Palm	NN	Mozilla		Internet Explorer							Opera			Safari		iCab	Kq	FF
2.x	4.x	1.x	1.8	4.x	5.0x (Win)	5.0 (Mac)	5.1+ (Mac)	5.5 (Win)	6	7	6	7	8	1.x	2.0	2.9+	3.x	1.0
○		●	●								●	●	●		●		●	●

table-row-group und table-column-group

Beide Werte, `table-row-group` und `table-column-group`, sollten zur gleichen Anzeige führen wie die HTML-Elemente `tbody` und `colgroup`. Beide werden jedoch derzeit von keinem Browser korrekt ausgeführt.

CSS-Element: `display:table-row-group`
und `display:table-column-group`

CSS-Version: CSS 2/3

Zulässig für folgende (X)HTML-Elemente: alle

Medium: alle

Vererbt: Nein

Palm	NN	Mozilla		Internet Explorer							Opera			Safari		iCab	Kq	FF
2.x	4.x	1.x	1.8	4.x	5.0x (Win)	5.0 (Mac)	5.1+ (Mac)	5.5 (Win)	6	7	6	7	8	1.x	2.0	2.9+	3.x	1.0
		●	●				○											

inline-table Mit dem Wert `inline-table` können Sie eine Inline-Tabelle simulieren. Das bedeutet, dass zwar der Inhalt mit Hilfe weiterer Elemente tabellarisch dargestellt werden kann, das Element jedoch keinen eigenen Block, sondern eine abgeleitete Box erzeugt.

CSS-Element: `display:inline-table`	
CSS-Version: CSS 2/3	*Zulässig für folgende (X)HTML-Elemente:* alle
Medium: alle	*Vererbt:* Nein

Palm	NN	Mozilla		Internet Explorer							Opera			Safari		iCab	Kq	FF
2.x	4.x	1.x	1.8	4.x	5.0x (Win)	5.0 (Mac)	5.1+ (Mac)	5.5 (Win)	6	7	6	7	8	1.x	2.0	2.9+	3.x	1.0
												●	●		●		●	

Ausschnitt (clip)

Mit der `clip`-Eigenschaft können Sie einen Ausschneidebereich definieren. Er bestimmt den Teil eines Elements, der sichtbar sein soll. Normalerweise entspricht der Ausschneidebereich dem Inhalt des Elements.

CSS-Element: `clip`	*Mögliche Werte: auto*, `inherit`, Shape-Wert
CSS-Version: CSS 2.0, 2.1[1], CSS 3, TV	*Zulässig für folgende (X)HTML-Elemente:* alle Blockelemente und alle ersetzten Elemente, deren `overflow`-Eigenschaft einen Wert ungleich `visible` hat. In CSS 2.1 nur positionierte Elemente.
Medium: Visual	*Vererbt:* Nein

Palm	NN	Mozilla		Internet Explorer							Opera			Safari		iCab	Kq	FF
2.x	4.x	1.x	1.8	4.x	5.0x (Win)	5.0 (Mac)	5.1+ (Mac)	5.5 (Win)	6	7	6	7	8	1.x	2.0	2.9+	3.x	1.0
	○	●[2]	●[2]		●[3]	●[2]	●[2]	●[2]	●[2]	●[2]		●[2]	●[2]	●[2]	●[2]	●[2+4]	●[2]	●[2]

[1] wesentliche Änderungen in CSS 2.1
[2] gemäß CSS 2.1
[3] gemäß CSS 2.0
[4] ab Version 3.0

Mögliche Werte

Der Wert `auto` besagt, dass der Ausschnitt die gleiche Größe wie das Element hat. Damit ist der gesamte Inhalt des Elements sichtbar. Dies ist auch der Standardwert.

Ein Shape-Wert ist die Definition einer rechteckigen Fläche. Dabei gilt folgende Syntax: `rect(`*`oben rechts unten links`*`)`. Die numerischen Werte *`oben`*, *`rechts`*, *`unten`* und *`links`* sind dabei die Abstände von der jeweiligen Kante der Box.

!! STOP

So sagt es jedenfalls der CSS-Standard 2.0. In CSS 2.1 wurde die Definition jedoch geändert. Hier werden die Werte `oben` und `unten` als Abstände von der oberen Kante und `rechts` und `links` als Abstände von der linken Kante interpretiert. Abbildung 5.17 verdeutlicht dies anhand eines rechteckigen Ausschnitts von 80 x 80 Pixel, der sich mittig in einer Box von 100 x 100 Pixel befindet. Auch die Reihenfolge hat sich geändert.

Für ein 100 x 100 Pixel großes Element sind daher die folgenden beiden Angaben äquivalent:

`rect(10px 10px 10px 10px)` *gemäß CSS 2.0*
`rect(10px 90px 90px 10px)` *gemäß CSS 2.1*

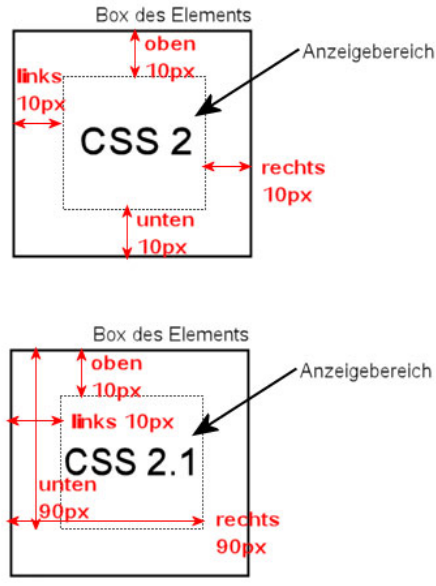
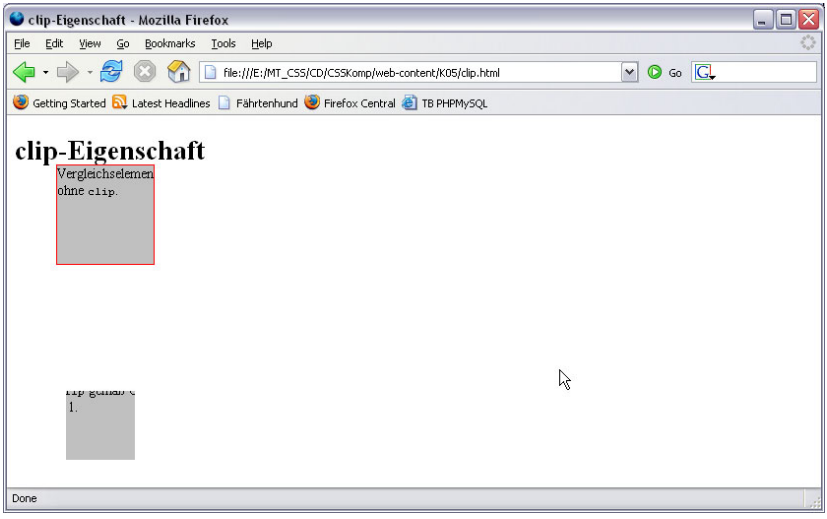


Abbildung 5.17: Vergleich der Werte in CSS 2 (oben) und 2.1 (unten)

Die einzelnen numerischen Werte müssen Sie (außer bei 0) mit einer Einheit versehen. Sie können einzelne Abstände auch mit dem Wert `auto` angeben, der dann der Länge 0 entspricht.

Sie finden das Beispiel auf der Buch-CD in der Datei BSP/K05/clip.html.

Abbildung 5.18: Korrekte Darstellung gemäß CSS 2.1 in Firefox



iCab 2.9.x und der Palm-Browser führen die `clip`-Eigenschaft nicht aus. Sie stellen aber zumindest die Inhalte der Elemente dar, so dass sie lesbar sind. Ab der Version 3.0 unterstützt iCab die `clip`-Eigenschaft fehlerfrei.

Außenabstand (margin, margin-top, margin-bottom, margin-left, margin-right)

Der Außenabstand bestimmt den Abstand eines Elements zwischen dessen Rahmenlinie (falls vorhanden) und dem umgebenden bzw. benachbarten Element. Der Außenabstand wird immer transparent dargestellt, d.h. nicht mit der Hintergrundfarbe bzw. dem Hintergrundbild gefüllt.

Wenn Sie für alle Seiten des Elements einen gleich großen Außenabstand festlegen möchten, verwenden Sie dazu die `margin`-Eigenschaft. Wenn Sie unterschiedliche Werte für die einzelnen Seiten des Elements festlegen möchten, können Sie dazu die Eigenschaften

- `margin-top`, oberer Außenabstand,
- `margin-bottom`, unterer Außenabstand,
- `margin-left`, linker Außenabstand, und
- `margin-right`, rechter Außenabstand

verwenden.

CSS-Element: `margin`, `margin-top`, `margin-bottom`, `margin-left`, `margin-right`	*Mögliche Werte:* `inherit`, numerischer Wert mit Maßangabe, *0*
CSS-Version: CSS 1-3, TV, Mobile	*Zulässig für folgende (X)HTML-Elemente:* alle; in CSS 2.1 alle, außer Elemente, deren `display`-Eigenschaft einen anderen Tabellenwert als `table` oder `inline-table` aufweist
Medium: Visual	*Vererbt:* Nein

Palm	NN	Mozilla		Internet Explorer							Opera			Safari		iCab	Kq	FF
2.x	4.x	1.x	1.8	4.x	5.0x (Win)	5.0 (Mac)	5.1+ (Mac)	5.5 (Win)	6	7	6	7	8	1.x	2.0	2.9+	3.x	1.0
⊙	○	●	●	● [1]	● [1]	●	●	● [1]	●	●	●	●	●	●	●	● [3]	●	●

1 Bei Verwendung von `margin` für Inline-Elemente können Probleme auftreten.
2 Kurzformen mit zwei bzw. vier Werten werden fehlerhaft dargestellt.
3 ab iCab 3.0 fehlerfrei

Sie finden das Beispiel auf der Buch-CD in der Datei BSP/K05/margin.html.

Um den Außenabstand von Elementen sichtbar zu machen, fasst das Beispiel die `div`-Elemente in übergeordnete `div`-Elemente ein. Diese umgebenden Elemente haben einen schwarzen Rahmen, einen Innenabstand (`padding`) von 0 und eine graue Füllfarbe.

Die `div`-Elemente, die mit verschiedenen Außenabständen formatiert wurden, haben hingegen einen gepunkteten roten Rahmen und eine weiße Füllfarbe. Daraus ergibt sich dann, dass der Außenabstand der graue Bereich zwischen dem gepunkteten Rahmen und dem Außenrahmen ist.

Listing 5.8: Die Beispielseite zum Testen der margin-Eigenschaft

```
<html>
    <head>
        <title>margin-Eigenschaft</title>
        <style type="text/css">
        <!--
            div {
                border:1px dotted red;
                background-color:white;
            }
            .umgebung {
                background-color:silver;
                padding:0;
                width:150px;
```

```
                height:150px;
                margin:10px;
                border:1px solid black
            }
            .margin0 {
                margin:0;
                height:148px;
                width:148px
            }
            .margingleich {
                margin:10px;
                height:130px;
                width:130px
            }
            .marginungleich {
                 margin-top:10px;
                 margin-bottom:20px;
                 margin-left:5px;
                 margin-right:15px;
                 height:120px;
                 width:130px
            }
        -->
        </style>
    </head>

    <body>
        <h1>margin-Eigenschaft</h1>
        <div class="umgebung">
            <div class="margin0">
                Vergleichselement ohne <code>margin</code>.
            </div>
        </div>
        <div class="umgebung">
            <div class="margingleich">
                Vergleichselement mit einheitlich 10px Abstand an
                allen Seiten.
            </div>
        </div>
        <div class="umgebung">
            <div class="marginungleich">
                Vergleichselement mit ungleichen Außenabständen.
            </div>
        </div>
    </body>
</html>
```

Bei iCab 2.9.x führt der Einsatz von `margin` – insbesondere zusammen mit Elementen, die eine feste Größe haben – zu einem absolut unbrauchbaren Ergebnis. iCab berücksichtigt die Außenabstände nicht, zumindest nicht bei der einheitlichen Angabe mit `margin:10px` für alle Seiten. Zudem hält iCab die Größe der Elemente nicht ein. Dadurch entsteht überfließender Text, der außerhalb des Elements angezeigt wird. Ab iCab 3.0 erhalten Sie eine brauchbare und fehlerfreie Anzeige.

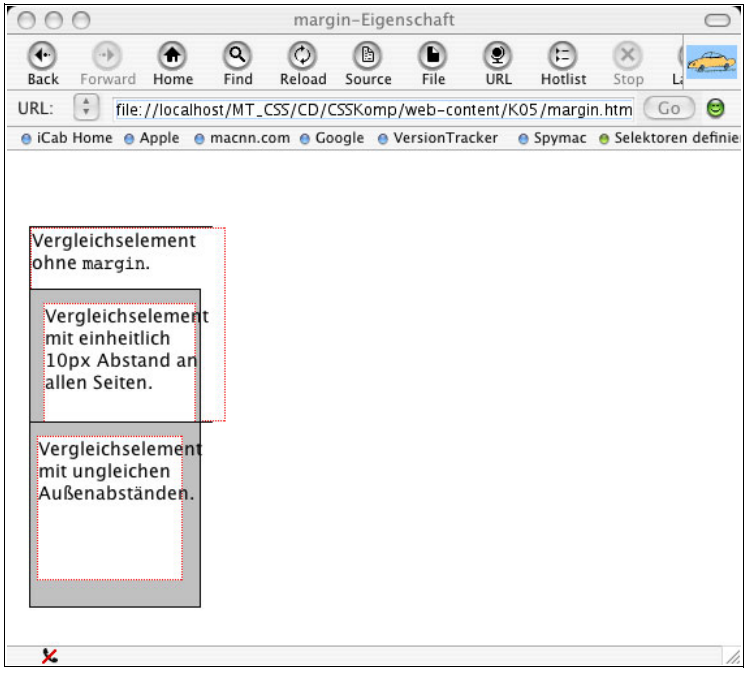


Abbildung 5.19: Das Ergebnis in iCab 2.9.x

Der Netscape Navigator 4 hält die Außenabstände auch nicht korrekt ein, das Ergebnis ist aber zumindest in vielen Fällen noch annehmbar.

Die vertikalen Außenabstände untereinander liegender Elemente werden zusammengefasst. Hat also beispielsweise ein Absatz einen unteren Außenabstand von 10px und der folgende Absatz einen oberen Außenabstand von 10px, wird kein Abstand von 20px, sondern nur ein Abstand von 10px angezeigt. Dies gilt allerdings nur für Elemente, die nicht positioniert sind und wenn es sich um Nicht-Floating-Elemente handelt. Sie können das in der Abbildung 5.20 sehen. Obwohl für alle übergeordneten Elemente ein Außenabstand von 10 Pixel definiert ist, beträgt er zwischen den vertikal angeordneten Elementen nur 10 und nicht 20 Pixel

Der Netscape Navigator 4 fasst vertikale Außenabstände nicht zusammen, stellt also sowohl den oberen wie den unteren Abstand beider Elemente dar.

Mehr zur Positionierung und zu den Floating-Elementen finden Sie in den Abschnitten »Positionierungsart (position)« und »Textumfluss (float)«.

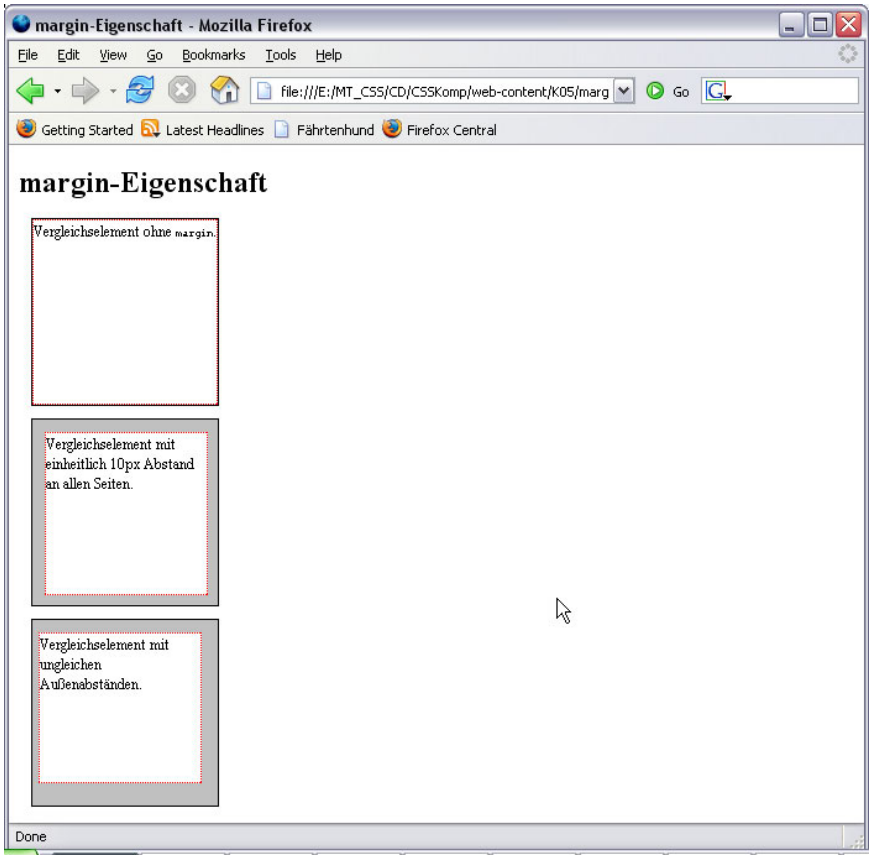


Abbildung 5.20: Korrekte Darstellung der margin-Eigenschaft in Firefox

Kurzform

Die `margin`-Eigenschaft stellt auch eine Kurzform bereit, wenn Sie unterschiedliche Werte für die Außenabstände der einzelnen Seiten definieren möchten. In diesem Fall lautet die Syntax:

```
margin: oben rechts unten links
```

Alternativ können Sie auch nur zwei Werte angeben. Dann lautet die Syntax:

```
margin: obenunten rechtslinks
```

Breite (width)

Die `width`-Eigenschaft legt die Breite eines Elements fest. Sie wird gemäß Boxmodell wie folgt berechnet:

Breite = `margin-left` + `border-left-width` + `padding-left` + `width` (bzw. Breite des Inhalts) + `padding-right` + `border-right-width` + `margin-right`

CSS-Element: `width`	*Mögliche Werte: auto*, `inherit`, positiver numerischer Wert mit Einheit, Prozentwert[1]
CSS-Version: CSS 1, CSS 2, CSS 3, TV, Mobile	*Zulässig für folgende (X)HTML-Elemente:* alle, außer nicht positionierte Inline-Elemente, Tabellenzeilen und Zeilengruppen
Medium: Visual	*Vererbt:* Nein

Palm	NN	Mozilla		Internet Explorer							Opera			Safari		iCab	Kq	FF
2.x	4.x	1.x	1.8	4.x	5.0x (Win)	5.0 (Mac)	5.1+ (Mac)	5.5 (Win)	6	7	6	7	8	1.x	2.0	2.9+	3.x	1.0
●	◎ 3+4	●	●	⊙ 2	●	●	●	●	●	●	●	●	●	◎ 4	◎ 4	● 4	◎ 4	●

1 erst ab CSS 2.0.
2 keine prozentualen Werte
3 Maße werden nicht immer exakt eingehalten, das Ergebnis ist aber durchaus brauchbar. Abweichende Darstellung vor allem bei Prozentwerten.
4 Prozentuale Werte bis zur Version 2.9.x werden zu klein dargestellt.

Prozentuale Werte beziehen sich auch hier auf die Breite des umgebenden Elements.

iCab 2.9.x, Safari und Konqueror stellen die `width`-Eigenschaft nur bei prozentualen Werten nicht korrekt dar. In allen drei Browsern füllt die Box bei einer Breite von 25% etwas weniger als ¼ des verfügbaren Platzes der umgebenden Box.

CSS TV-Standard sieht vor, dass die Browser für die TV-Wiedergabe eine maximale Breite von 100% intern festlegen können. Es kann daher sein, dass Werte für die `width`*-Eigenschaft, die größer 100% sind, bzw. absolute Werte, die über die Breite des Viewports hinausgehen, nicht dargestellt werden.*

Sie finden das Beispiel auf der Buch-CD in der Datei BSP/K05/width.html.

Wenn Sie beispielsweise für ein `div`-Element die Breite auf `300px` festlegen, sollte die daraus resultierende Box also eine Gesamtbreite von 300 Pixel haben, die auch die Außenabstände (`margin`) beinhaltet.

```
.breite { width:300px; }
```

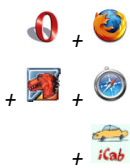
Die Höhe eines Elements können Sie mit der `height`*-Eigenschaft festlegen. Mehr dazu finden Sie weiter unten im Abschnitt »Höhe (height)«. Über die Eigenschaften* `min-width` *und* `max-width` *können Sie auch die minimale und maximale Höhe festlegen.*

Sie können die Eigenschaften `min-width`*,* `max-width` *und* `width` *auch gleichzeitig anwenden. Problematisch können dabei widersprüchliche Angaben sein. Folgendes Beispiel soll dies verdeutlichen.*

```
.widerspruch { font-size:10px; max-width:100px;
               min-width:250px; width:150px; }
```

Definieren Sie für ein Element eine Breite, die sowohl der Angabe für `min-width` als auch der für `max-width` entspricht, verhalten sich die Browser bei der Lösung dieses Widerspruchs im Gegensatz zur Höhe (`height`) recht einheitlich.

Kein Problem haben alle Browser, die mit den Eigenschaften `min-width` und `max-width` nichts anfangen können. Sie wenden einfach nur die mit `width` festgelegte Breite an. Für iCab gilt das nur bis zur Version 2.9.x einschließlich.



Bei der Breite verhalten sich im Gegensatz zur Höhe alle anderen Browser (incl. iCab ab Version 3.0) einheitlich. Sie richten sich im Zweifelsfall nach der Mindestbreite des Elements und lassen `width` und `max-width` außer Acht. Damit hat also `min-width` die Priorität vor den anderen Angaben.

Folgendes Beispiel zeigt die Anwendung der Eigenschaften `width`, `min-width` und `max-width`.

Die Beispielseite definiert verschiedene CSS-Klassen, die eine exakte Breite in Pixel (`.breite`) eine minimale (`.min`) und eine maximale (`.max`) Breite definieren. Die Klasse `.widerspruch` legt widersprüchliche Angaben fest und `.prozent` definiert die Breite in Prozent.

Listing 5.9: Definierte CSS-Klassen

```
div           { border:1px solid silver }
.breite       { width:300px }
.min          { min-width:300px }
.max          { max-width:100px }
.widerspruch  { font-size:10px; max-width:100px;
                min-width:250px; width:150px; }
.prozent      { width:25% }
```

Jede dieser CSS-Klassen wird dann auf ein `div`-Element angewendet.

Listing 5.10: Anwenden der Stile

```
<body>
    <h1>width-, max-width und min-width-Eigenschaft</h1>
    <div class="breite">Diese Box sollte eine Breite von 300 Pixel
    haben.</div>
```

```
    <div class="min">Diese Box sollte mindestens 300 Pixel Breite haben,
    also größer oder gleich breit wie die vorstehende sein.</div>
    <div class="max">Diese Box sollte maximal 100 Pixel breit sein.</div>
    <div class="widerspruch">Für diese Box wurde eine Breite (150px)
    definiert, die die Mindestbreite (200px) unterschreitet und die
    maximale Breite (100px) überschreitet. </div>
    <div class="prozent">Diese Box sollte 25% der Seitenbreite
    haben.</div>
</body>
```

Im Ergebnis sollten damit der erste Absatz eine Breite von 300 Pixel, der zweite mindestens die gleiche Breite und der dritte 100 Pixel Breite haben. Der unterste Absatz muss ¼ der Seitenbreite (abzüglich des Innenabstands für das `body`-Element) haben.

Abbildung 5.21: Korrekte Darstellung der Beispielseite im Opera-Browser

Flusssteuerung (clear)

Definieren Sie mehrere Floating-Boxen (siehe *Abschnitt »Textumfluss (float)«*), können Sie mit Hilfe der `clear`-Eigenschaft festlegen, auf welcher Seite einer Floating-Box keine andere Floating-Box stehen darf.

Mehr zu Floating-Boxen finden Sie im Abschnitt »Textumfluss (float)«.

CSS-Element: `clear`	*Mögliche Werte: none*, `left`, `right`, `both`, `inherit`
CSS-Version: CSS 1-3, TV, Mobile	*Zulässig für folgende (X)HTML-Elemente:* alle Blockelemente, auch Floating-Boxen
Medium: Visual	*Vererbt:* Nein

Palm	NN	Mozilla		Internet Explorer							Opera			Safari		iCab	Kq	FF
2.x	4.x	1.x	1.8	4.x	5.0x (Win)	5.0 (Mac)	5.1+ (Mac)	5.5 (Win)	6	7	6	7	8	1.x	2.0	2.9+	3.x	1.0
●	● [1]	●	●	● [1]	●	●	●	●	●	●	●	●	●	●	●	● [2]	●	●

[1] Da die `float`-Eigenschaft nicht korrekt funktioniert, ist der Einsatz von `clear` ebenfalls nicht brauchbar.
[2] ab Version 3.0

Mögliche Werte

Die Werte der Eigenschaft haben eine unterschiedliche Bedeutung, abhängig davon, ob Sie sie auf eine Floating-Box oder eine normale Box anwenden.

- `both`: In diesem Fall wird der Textumfluss außer Kraft gesetzt. Das Element wird unterhalb der letzten Floating-Box positioniert.
- `none`: Alle Floating-Boxen verhalten sich wie vorgesehen. Es gibt keine Beschränkungen.
- `left`: Für die Box des Elements wird der obere Rand vergrößert, bis die Box unterhalb der letzten Floating-Box steht, die mit `float:left` formatiert ist.
- `right`: Für die Box des Elements wird der obere Rand vergrößert, bis die Box unterhalb der letzten Floating-Box steht, die mit `float:right` formatiert ist.

Bei Anwendung der Werte auf eine Floating-Box wird diese ebenfalls verschoben, mit dem Ergebnis, das dies unter Umständen auch Einfluss auf andere Floating-Boxen hat, da diese dann den Platz der verschobenen Box einnehmen oder sich der Textumfluss der Seite ändert.

Sie finden das Beispiel auf der Buch-CD in der Datei BSP/K05/clear.html.

Enthält die Seite zwei Floating-Boxen, von der die erste links, die zweite rechts ausgerichtet ist, und zusätzlich einen normalen Absatz, umfließt der Absatz die Boxen.

Abbildung 5.22: Dies ist der normale Textumfluss um die Floating-Boxen.

Listing 5.11: Aufbau der Beispielseite

```
<div style="float:left;width:120px;
    border:1px solid red;margin:5px;
    padding:10px">Dieser Kasten sollte links ausgerichtet
      sein.</div>
<div style="float:right; width:120px;
    border:1px solid gray;margin:5px;
    padding:10px">Dieser Kasten sollte rechts ausgerichtet
      sein.</div>
<p>Dieser Text umfließt die beiden Kästchen, dieser Text umfließt die
    beiden Kästchen, dieser Text umfließt die beiden Kästchen, dieser
    Text umfließt die beiden Kästchen, dieser Text umfließt die beiden
    Kästchen, dieser Text umfließt die beiden Kästchen, dieser Text
    umfließt die beiden Kästchen, dieser Text umfließt die beiden
    Kästchen.</p>
```

Wenn Sie nun den einleitenden `p`-Tag um `style="clear:left"` ergänzen, wird der Absatz unter die beiden Boxen verschoben. Der Grund ist, dass der Absatz dann unterhalb der linken Box angezeigt wird. An der Position der rechten Box ändert sich nichts.

Geben Sie für den Absatz `clear:none` an, dafür aber für das zweite `div`-Element `clear:left`, wird die rechte Floating-Box unter den Absatz gesetzt, der die linke Floating-Box umfließt:

```
    <div style="float:right; clear:left;
                width:120px; border:1px solid gray;
                margin:5px; padding:10px">
```

Abbildung 5.23: Das ist das Ergebnis von clear:left für den Absatz.

!! STOP

Es ist unbedingt erforderlich, dass Sie die `clear`*-Eigenschaft nach der* `float`*-Eigenschaft eingeben. Ansonsten würde durch das Setzen der* `float`*-Eigenschaft der Wert der* `clear`*-Eigenschaft überschrieben. Zumindest gilt dies für den Internet Explorer für Windows. Allen anderen Browsern ist die Reihenfolge egal.*

Wenn sie einmal die falsche Reihenfolge der Anweisungen definiert haben, korrigiert der Internet Explorer 6 die Anzeige nicht mehr, auch nicht nach mehrfachem Reload der Seite.

Höhe (height)

Die Eigenschaft `height` definiert die Höhe eines Elements. Sie kann generell nur positive Werte annehmen. Prozentuale Werte sind erst ab CSS 2.0 möglich und beziehen sich immer auf die Höhe des umgebenden Elements.

CSS-Element: `height`	*Mögliche Werte: auto*, `inherit`, positiver numerischer Wert mit Einheit, Prozentwert[1]
CSS-Version: CSS 1, CSS 2, CSS 3, TV, Mobile	*Zulässig für folgende (X)HTML-Elemente:* alle außer nicht positionierte Inline-Elemente, Tabellen und Tabellenzellen
Medium: Visual	*Vererbt:* Nein

Palm	NN	Mozilla		Internet Explorer							Opera			Safari		iCab	Kq	FF
2.x	4.x	1.x	1.8	4.x	5.0x (Win)	5.0 (Mac)	5.1+ (Mac)	5.5 (Win)	6	7	6	7	8	1.x	2.0	2.9+	3.x	1.0
		○ 2	○ 2		○ 2	●	●	○ 2	○ 2	○ 2	○ 2	○ 2	○ 2	○ 2	○ 2	○ 3	○ 2	○ 2

1 erst ab CSS 2.0
2 keine prozentualen Werte
3 ab iCab 3.0

Sie finden das Beispiel auf der Buch-CD in der Datei BSP/K05/height.html.

Wenn Sie beispielsweise für ein `div`-Element die Höhe auf `3em` festlegen, sollte die daraus resultierende Box dreimal so hoch wie die Schrift in der Box sein.

```
.hoch3em { height:3em; }
```

Die Breite eines Elements können Sie mit der `width`*-Eigenschaft festlegen. Mehr dazu finden Sie weiter unten im Abschnitt »Breite (width)«. Über die Eigenschaften* `min-height` *und* `max-height` *können Sie auch die minimale und maximale Höhe festlegen.*

Sie können die Eigenschaften `min-height`*,* `max-height` *und* `height` *auch gleichzeitig anwenden. Bei widersprüchlichen Angaben verfahren die Browser jedoch unterschiedlich. Folgendes Beispiel soll dies verdeutlichen.*

Mit dem Stil

```
.widerspruch {
    font-size:8px;
    min-height:25px;
    max-height:10px;
    height:15px;
}
```

wird eine Höhe für das Element festgelegt, die größer ist als notwendig, da die Schrift lediglich 8 Pixel hoch ist. Gleichzeitig ist die minimale Höhe größer als die definierte Höhe und die maximale Höhe kleiner als die Höhe. Sicherlich ist eine solche Konstruktion nicht sinnvoll, dennoch können Sie an einem solchen Beispiel sehr schön sehen, welchem Wert die Browser die höchste Priorität geben.

Der Internet Explorer (bis einschließlich Version 6) beherrscht die Eigenschaften `max-height` und `min-height` nicht. Daher stellt sich hier die Frage nach der Priorität gar nicht. Genau wie alle anderen Browser, die ausschließlich die `height`-Eigenschaft unterstützen, werden auch hier die Angaben für `max-height` und `min-height` einfach ignoriert.

Der Opera-Browser berücksichtigt in dieser Situation die Einstellung für `max-height`. Der Wert für diese Eigenschaft überschreibt also den Wert für `height`. Der Wert für `min-height` wird ignoriert, falls er dem von `max-height` widerspricht.

Genau anders verfahren Mozilla, Firefox, iCab 3.0 und Netscape 6/7+. Diese Browser ignorieren den Wert von `max-height` und verwenden den Wert von `min-height`.

Beachten Sie, dass eine Angabe der Höhe für ein Element die Position der nachfolgenden Elemente im normalen Fluss beeinflusst. Das heißt, bei Angabe einer Höhe, die niedriger ist als zur Darstellung des Inhalts notwendig, läuft der Inhalt über die Box hinaus. Das bewirkt, dass der Inhalt und der Inhalt der folgenden Box übereinander liegen. Das zeigt auch die korrekte Darstellung der Beispielseite, bei der sich die unteren Absätze überlappen

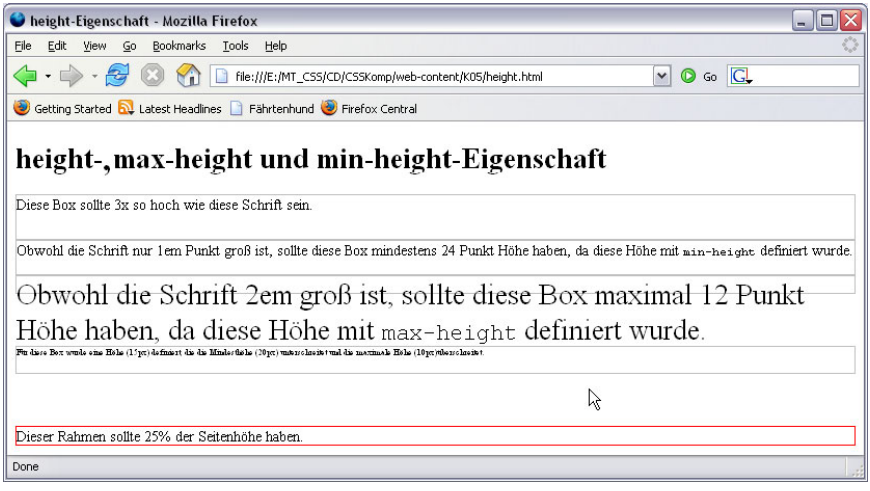


Abbildung 5.24: Fast korrekte Darstellung der Beispielseite in Firefox, nur die Höhe von 25% für den letzten Absatz wird nicht berücksichtigt.

Innenabstand (padding, padding-top, padding-bottom, padding-left, padding-right)

Die Eigenschaft `padding` legt den Innenabstand eines Elements fest. Das ist der Abstand zwischen dem Inhalt und dem umgebenden Rahmen (falls vorhanden). Der Innenabstand wird immer in der gleichen Formatierung (Füllfarbe/Hintergrund) dargestellt wie der Inhalt des Elements.

Wenn Sie für alle Seiten des Elements einen gleich großen Innenabstand festlegen möchten, verwenden Sie dazu die `padding`-Eigenschaft. Wenn Sie unterschiedliche Werte für die einzelnen Seiten des Elements festlegen möchten, können Sie dazu die Eigenschaften

- `padding-top`: oberer Innenabstand,
- `padding-bottom`: unterer Innenabstand,
- `padding-left`: linker Innenabstand und
- `padding-right`: rechter Innenabstand

verwenden.

:-) TIPP

Genau wie bei `margin`*, gibt es auch bei* `padding` *Kurzformen. Sie haben die gleiche Syntax wie bei* `margin`*, also:*

```
padding: oben rechts unten links
```

und:

```
padding: obenunten rechtslinks
```

CSS-Element: `padding`, `padding-top`, `padding-bottom`, `padding-left`, `padding-right`	*Mögliche Werte:* `inherit`, Prozentwert, numerischer Wert mit Maßangabe, *0*
CSS-Version: CSS 1-3, TV, Mobile	*Zulässig für folgende (X)HTML-Elemente:* alle, in CSS 2.0 alle, außer Elemente deren `display`-Eigenschaft einen anderen Tabellenwert als `table`, `inline-table` oder `table-cell` aufweist
Medium: Visual	*Vererbt:* Nein

Palm	NN	Mozilla		Internet Explorer							Opera			Safari		iCab	Kq	FF
2.x	4.x	1.x	1.8	4.x	5.0x (Win)	5.0 (Mac)	5.1+ (Mac)	5.5 (Win)	6	7	6	7	8	1.x	2.0	2.9	3.x	1.0
⊙ 2	○ 1	●	●	● 1	● 1	○	○	● 1	●	●	●	●	●	●	●	●	●	●

1 Bei Verwendung von `padding` können Probleme für Inline-Elemente auftreten.
2 Kurzform mit zwei/vier Werten wird fehlerhaft angezeigt.

Geben Sie einen prozentualen Wert an, bezieht sich dieser immer auf die Breite des umgebenden Elements.

Sie finden das Beispiel auf der Buch-CD in der Datei BSP/K05/padding.html.

Beispiel

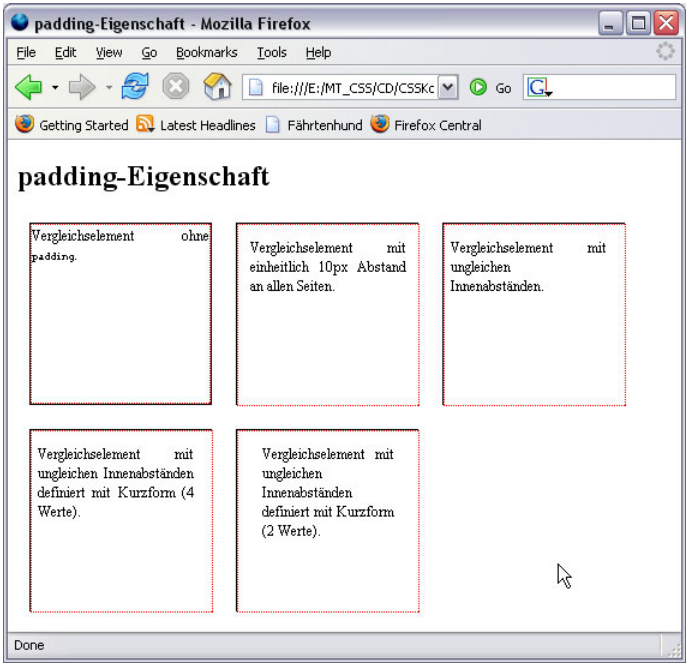


Abbildung 5.25: So sollte die Beispielseite korrekt dargestellt werden.

Listing 5.12: Stile zur Demonstration der padding-Eigenschaft

```
div {
    border:1px dotted red;
    background-color:white;
    text-align:justify
}
```

```
.umgebung {
    float:left;
    background-color:silver;
    padding:0;
    width:150px;
    height:150px;
    margin:10px;
    border:1px solid black
}
.padding0        { padding:0; height:148px; width:148px }
.paddinggleich   { padding:10px; height:130px; width:130px }
.paddingungleich { padding-top:10px; padding-bottom:20px;
                   padding-left:5px; padding-right:15px;
                   height:120px;width:130px }
.kurzform1       { padding:10px 15px 20px 5px;
                   height:120px; width:130px }
.kurzform2       { padding:10px 20px;
                    height:130px; width:110px }
```

Mac

Die Mac-Version des Internet Explorers stellt die `padding`-Eigenschaft nicht korrekt dar. Er stellt das Boxmodell fehlerhaft dar, da er den Innenabstand außerhalb des Rahmens anzeigt und in die angegebene Breite und Höhe mit einrechnet. Bei richtiger Darstellung bestimmen aber `width` und `height` die Breite und Höhe des Inhaltbereichs. Der Innenabstand müsste hinzuaddiert und um das Ganze der Rahmen gezeichnet werden.

:-) TIPP

Wenn Sie, wie im Beispiel, ein übergeordnetes Element definieren, dessen Breite und Höhe Sie festlegen, können Sie das Problem umgehen, indem Sie nur für den Internet Explorer für Macintosh die Größe der inneren Elemente auf 100% festlegen: `width:100%;height:100%`. *Dann stellt der Browser die Seite korrekt dar. Achten Sie dann aber unbedingt darauf, dass auch wirklich nur der Internet Explorer für Mac die Stile ausführt, sonst zeigen die anderen Browser ein fehlerhaftes Ergebnis.*

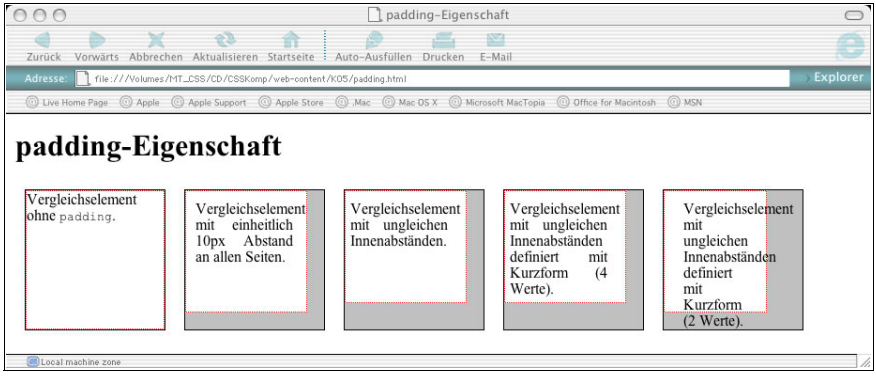


Abbildung 5.26: Fehlerhafte Darstellung im Internet Explorer für Mac

Mehr zum Boxmodell finden Sie am Anfang dieses Kapitels im Abschnitt »Das Boxmodell«. Erläuterungen zu CSS-Browserweichen enthält Kapitel 3, »Browseroptimierung«.

Linke Position (left)

Mit der `left`-Eigenschaft legen Sie fest, wie weit die linke Inhaltskante des Blocks von der linken Kante des umschließenden Blocks nach rechts verschoben wird. Damit definieren Sie also die linke Position des Elements. Genau wie die Eigenschaften `right`, `top` und `bottom` können Sie die Eigenschaft nur auf positionierte Elemente anwenden.

Näheres zur Positionierung finden Sie auch im Abschnitt über die `position`*-Eigenschaft.*

CSS-Element: `left`	*Mögliche Werte: auto*, `inherit`, Prozentwert, numerischer Wert mit Einheit
CSS-Version: CSS 2, CSS 3, TV	*Zulässig für folgende (X)HTML-Elemente:* alle positionierten Elemente
Medium: Visual	*Vererbt:* Nein

Palm	NN	Mozilla		Internet Explorer							Opera			Safari		iCab	Kq	FF
2.x	4.x	1.x	1.8	4.x	5.0x (Win)	5.0 (Mac)	5.1+ (Mac)	5.5 (Win)	6	7	6	7	8	1.x	2.0	2.9+	3.x	1.0
●	● [2]	●	●			●	○ [1]	●	●	●	●	●	●	●	●	● [3]	●	●

1 Bei Angabe von `right` und `left` werden beide Werte berücksichtigt, aber nicht korrekt dargestellt.
2 Bei Angabe von `right` und `left` wird `right` ignoriert.
3 ab Version 3.0

Sie finden das Beispiel auf der Buch-CD in der Datei BSP/K05/left.html.

Bei absoluter Positionierung können Sie mit folgendem Klassenstil ein Element am linken Rand ausrichten:

```
.links { position:absolute; left:0px }
```

Bei relativer Positionierung wird die linke Position des Blocks ausgehend von seiner normalen Position im Textfluss verschoben. Positive Werte verschieben den Block nach rechts, negative verschieben ihn nach links. Im folgenden Fall wird der Block also um `5em` nach rechts eingerückt.

```
.eingerueckt { position:relative; left:5em }
```

Legen Sie die Position gleichzeitig mit `left` und `right` fest, ergibt sich aus der Differenz beider Werte die Breite des Elements.

```
.leftright { position:absolute;
             left:10em; right:10em;
             top:10em; border:1px solid red }
```

Die Mac-Version des Internet Explorers zeigt bei Angabe von `right` und `left` eine horizontale Scrollleiste an, obwohl dies für die Anzeige des Seiteninhalts nicht erforderlich ist, da der entsprechende Block auch verkleinert werden könnte, wenn keine Breite definiert ist.

Mac

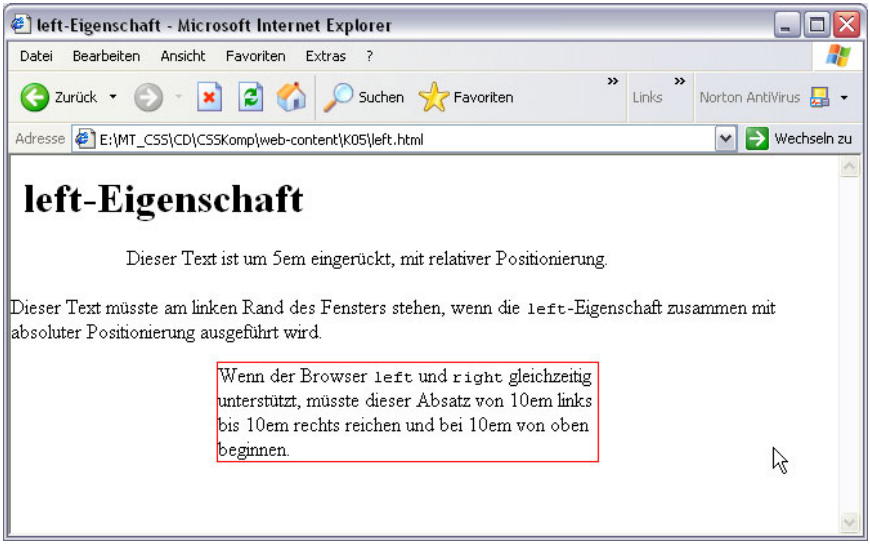


Abbildung 5.27: Die left-Eigenschaft funktioniert in der Windows-Version des Internet Explorers ohne Einschränkungen.

Der Palm-Browser stellt die `left`-Eigenschaft korrekt dar. Die Beispielseite wird aber dennoch nicht fehlerfrei angezeigt, weil der Browser die `top`-Eigenschaft nicht unterstützt. Diese ist allerdings erforderlich, damit der absolut positionierte zweite Absatz der Seite nicht über der Seitenüberschrift liegt.

Obere Position (top)

Mit der `top`-Eigenschaft definieren Sie, wie weit die oberste Inhaltskante der Box von der oberen Kante des umschließenden Blocks entfernt dargestellt wird.

CSS-Element: `top`	*Mögliche Werte: auto*, `inherit`, Prozentwert, numerischer Wert mit Einheit
CSS-Version: CSS 2, CSS 3, TV	*Zulässig für folgende (X)HTML-Elemente:* alle positionierten Elemente
Medium: Visual	*Vererbt:* Nein

Palm	NN	Mozilla		Internet Explorer							Opera			Safari		iCab	Kq	FF
2.x	4.x	1.x	1.8	4.x	5.0x (Win)	5.0 (Mac)	5.1+ (Mac)	5.5 (Win)	6	7	6	7	8	1.x	2.0	2.9+	3.x	1.0
		●	●			● [1]	● [1]	● [1]	● [1]	● [1]	●	●	●	●	●	● [2]	●	●

1 Bei Angabe von `top` und `bottom` wird der Wert für `bottom` ignoriert.
2 ab Version 3.0

Wenn Sie einen prozentualen Wert für `top` *eingeben, bezieht sich dieser auf die Höhe des umgebenden Blocks.*

Sie finden das Beispiel auf der Buch-CD in der Datei BSP/K05/top.html.

Folgendes Beispiel soll die Verwendung der `top`-Eigenschaft demonstrieren. In der Seite werden neben der Seitenüberschrift drei Absätze definiert. Der erste wird mit der CSS-Klasse `oben` am oberen Seitenrand positioniert.

```
.oben { position:absolute; top:0px }
```

Dazu wird zunächst die `position`-Eigenschaft auf `absolute` gesetzt und anschließend die `top`-Eigenschaft auf 0. Dadurch beginnt der obere Rand der Box am oberen Rand des Inhalts des umgebenden Blocks. Hier ist dies der Block des `body`-Elements.

Der zweite Absatz wird mit der CSS-Klasse `.relativoben` formatiert.

```
.relativoben { position:relative; top:50% }
```

Mit der Angabe `position:relative` wird der Absatz nun relativ positioniert. In diesem Fall bedeutet dies, dass der Absatz normalerweise unterhalb des `h1`-Elements stehen würde, da absolut positionierte Elemente wie der erste Absatz aus dem normalen Fluss der Seite herausfallen. Das zweite `p`-Element hätte also einen ganz normal großen Abstand zur Überschrift. Durch die Angabe `top:50%` wird der untere Rand des Absatzes nun aber um `50%` nach unten verschoben. Dadurch wird der Absatz fast ohne Abstand unterhalb der Box der Überschrift angezeigt.

Für den dritten Absatz, der ebenfalls absolut positioniert wird, wird sowohl ein Wert für `top` als auch für `bottom` angeben und damit die Position von unten und von oben festgelegt. Aus der Differenz ergibt sich gleichzeitig die Höhe des Elements.

```
.topbottom { position:absolute;
             bottom:10em; top:10em;
             border: 1px solid red }
```

Zur Kennzeichnung der Elementgröße sorgt `border:1px solid red` für einen roten Rahmen um den Absatz.

Listing 5.13: Aufbau des Beispiels

```
    <head>
...
        <style type="text/css">
        <!--
           .oben        { position:absolute; top:0px }
           .relativoben { position:relative; top:50% }
           body         { border: 1px solid gray }
           .topbottom   { position:absolute;
                          bottom:10em; top:10em;
                          border: 1px solid red }
        -->
        </style>
    </head>

    <body>
        <h1>top-Eigenschaft</h1>
        <p class="oben">Dieser Text müsste am oberen Rand des Fensters
        stehen, wenn die <span><code>top</code></span>-Eigenschaft
        zusammen mit absoluter Positionierung ausgeführt wird.</p>
        <p class="relativoben">Falls der Browser die <code>top</code>-
        Eigenschaft zusammen mit relativer Positionierung
        unterst&uuml;tzt sollte dieser Text 50% unterhalb der
        &Uuml;berschrift stehen.</p>
        <p class="topbottom">Wenn der Browser <code>bottom</code> und
        <code>top</code> gleichzeitig unterst&uuml;tzt, m&uuml;sste
        dieser Absatz von 10em oben, bis 10em unten reichen.</p>
    </body>
```

Näheres zu Positionierung finden Sie auch im Abschnitt über die `position`*-Eigenschaft.*

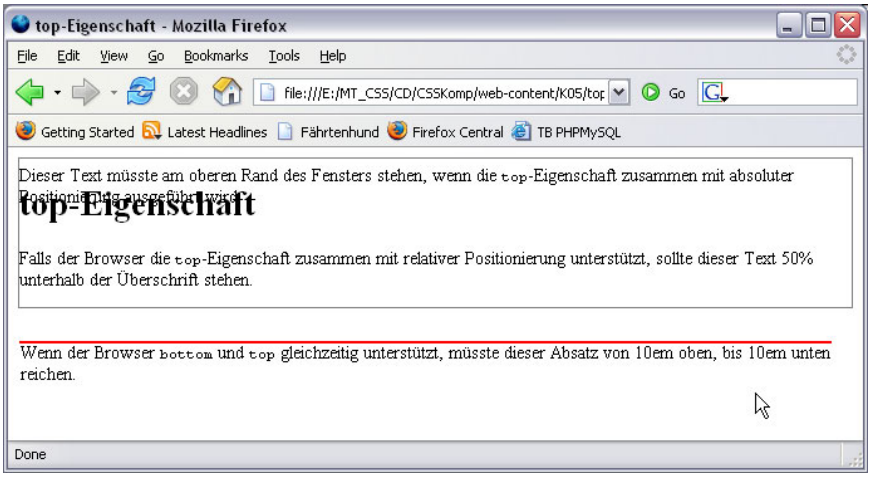


Abbildung 5.28: Korrekte Darstellung der Beispielseite

Positionierungsart (position)

CSS kennt zwei Eigenschaften, mit denen Sie die Art der Positionierung bestimmen können: `position` und `float`. Dahinter verbergen sich einfach nur zwei verschiedene Berechnungsarten für die Position eines Elements und zur Steuerung des Textumflusses für das Element.

`float` ist bereits Bestandteil des CSS 1.0-Standards, `position` wurde erst mit CSS 2.0 eingeführt.

Näheres zur Eigenschaft `float` *finden Sie im Abschnitt »Textumfluss (float)«.*

CSS-Element: `position`	*Mögliche Werte: static*, `relative`, `absolute`, `fixed`[1], `inherit`
CSS-Version: CSS 2, CSS 3, TV	*Zulässig für folgende (X)HTML-Elemente:* alle Elemente, nicht auf erzeugten Inhalt; in CSS 2.1 auch auf erzeugen Inhalt
Medium: Visual	*Vererbt:* Nein

Palm	NN	Mozilla		Internet Explorer							Opera			Safari		iCab	Kq	FF
2.x	4.x	1.x	1.8	4.x	5.0x (Win)	5.0 (Mac)	5.1+ (Mac)	5.5 (Win)	6	7	6	7	8	1.x	2.0	2.9+	3.x	1.0
⊙		●	●	◎ 2+3	◎ 2	⊙	●	◎ 2	◎ 2	⊙ 2	●	●	●	●	●	● 4	●	●

1 nicht in CSS TV-Standard
2 `fixed` wird nicht unterstützt
3 Der Wert `absolute` wird nur für `span`- und `div`-Elemente unterstützt.
4 ab Version 3.0

Der gewählte Wert für die `position`-Eigenschaft wirkt sich nicht nur auf die Position eines Elements aus, sondern auch auf die Anwendbarkeit anderer Eigenschaften. Bestimmte Eigenschaften, wie beispielsweise `top`, `left`, `bottom` und `right`, können nur auf positionierte Elemente angewendet werden.

Als positioniert gilt ein Element dann, wenn seine `position`*-Eigenschaft einen Wert ungleich* `static` *hat.*

Außerdem hat der Wert der Eigenschaft auch Einfluss auf die nachfolgenden und umgebenden Elemente und die Stapelreihenfolge, die Sie explizit mit `z-index` festlegen können.

Mehr zur Stapelreihenfolge im Abschnitt »Stapelreihenfolge (z-index)«.

Sie finden das Beispiel auf der Buch-CD in der Datei BSP/K05/position.html.

Verfügbare Werte

- `static`: Bei diesem Wert wird das Element gemäß seiner Position im normalen Fluss der Seite positioniert. Es wird also nicht verschoben. `static` ist der Standardwert. Bei Angabe von `static` werden Angaben zur Position (`top`, `left`, `bottom` und `right` ignoriert).
- `relative`: Geben Sie den Wert `relative` an, wird zunächst die Position des Elements gemäß dem normalen Fluss berechnet. Ausgehend von dieser Position wird das Element dann um die Werte der Eigenschaften `top`, `left`, `bottom` und `right` verschoben. Ein relativ positioniertes Element hat keinen Einfluss auf die umliegenden und nachfolgenden Elemente, da es den normalen Fluss nicht verändert. Definieren Sie beispielsweise zwei Absätze und verschieben den ersten durch relative Positionierung ein wenig nach unten und den zweiten gar nicht (`position:static`), überlappen sich beide, weil der erste versetzt wurde, der zweite jedoch an seiner ursprünglichen Stelle angezeigt wird.

```
.relative { position:relative;
            top:30px; left:30px;
            border:1px solid black }
<div class="relative">Dieser Absatz wurde um 30px nach rechts unten
   verschoben.</div>
<p>Dieser Absatz folgt nach dem relativ positionierten.</p>
```

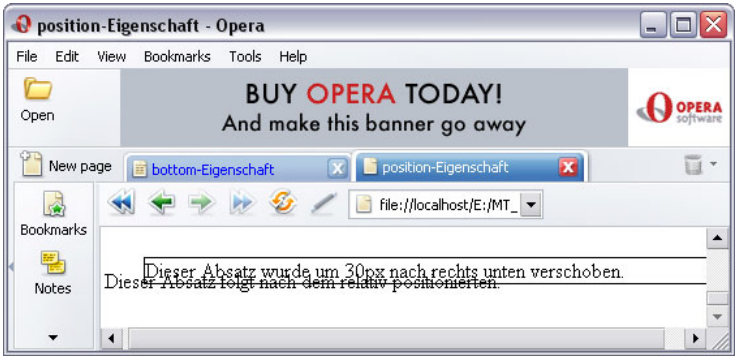


Abbildung 5.29: Der relativ positionierte Absatz überdeckt den nicht positionierten Absatz.

- `absolute`: Dieser Wert legt fest, dass das Element absolut positioniert wird. In diesem Fall geben die Werte `left`, `right`, `top` und `bottom` die Abstände des Elements zum umgebenden Element an.

INFO

Absolut positionierte Elemente beeinflussen nicht die Position nachfolgender Elemente, da sie aus dem normalen Fluss herausfallen.

- `fixed`: Positionieren Sie Elemente mit `fixed`, wird deren Position generell wie bei `absolute` berechnet. Im Unterschied zu einfachen absolut positionierten Elementen behält ein solches Element beim Scrollen seine Position in Bezug auf den Viewport bei. Es bleibt also stehen, wenn der Benutzer nach unten scrollt.

Folgendes Beispiel demonstriert die Verwendung der beiden Werte `fixed` und `absolute`. Es definiert zunächst zwei Stile, die jeweils ein Element absolut und fix positionieren. Beide Elemente bekommen eine Breite von 150 Pixel und einen Rahmen. Der absolut positionierte Stil erhält einen schwarzen, der andere einen roten Rahmen:

Listing 5.14: Die erforderlichen Stile zum Positionieren der Kästen

```
.absolute {
    position:absolute;
    top:10px;
    right:120px;
    width:100px;
    border: 1px solid black
}
```

```
.fixed {
    position:fixed;
    top:10px;
    right:10px;
    width:100px;
    border: 1px solid red
}
```

Wenden Sie diese Stile nun auf zwei `div`-Elemente an, sollten diese oben rechts in der Seite (mit je 10 Pixel Abstand von oben und rechts und zueinander) angezeigt werden.

```
<div class="absolute">Dies ist ein absolut positionierter Kasten.</
   div>
<div class="fixed">Dies ist ein fix positionierter
Kasten und sollte beim Scrollen stehen
bleiben.</div>
```

Abbildung 5.30: So sollten die beiden Kästchen dargestellt werden.

Auf den ersten Blick sehen beide `div`-Elemente aus, als wären sie einfach nur absolut positioniert. Der Unterschied wird beim Scrollen deutlich: Der rechte, rot umrandete Kasten bleibt oben rechts, egal welcher Teil der Seite sichtbar ist. Der schwarz umrandete, absolut positionierte Kasten verschwindet genau wie unpositionierte und relativ positionierte Elemente.

Der Palm-Browser stellt den mit `fixed` positionierten Kasten zwar dar und fixiert ihn auch im Viewport. Die Position liegt allerdings um ca. 10 Pixel zu tief.

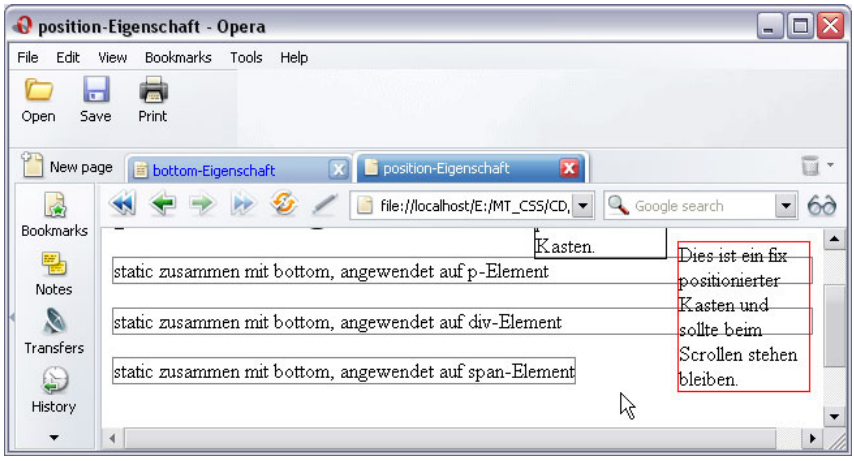


Abbildung 5.31: Nach dem Scrollen ist der Unterschied sichtbar.

Mac In früheren Versionen des Internet Explorers für Mac wurden Elemente nicht korrekt dargestellt, die mit `position:fixed` formatiert waren und einen Hyperlink enthielten. Dieses Problem ist in der Version 5.2 des Browsers behoben.

Der Internet Explorer 7 positioniert die mit `position:fixed` positionierte Box fehlerhaft. Auch hier scrollt diese Box mit dem Inhalt der Seite.

Rechte Position (right)

Mit der Eigenschaft `right` können Sie, analog zu `left`, die rechte Position eines Elements bestimmen. Die Eigenschaft legt fest, wie weit die rechte Inhaltskante des Elements von der rechten Kante des umgebenden Elements entfernt angezeigt wird.

REF *Näheres zu Positionierung finden Sie auch im Abschnitt über die* `position`*-Eigenschaft.*

CSS-Element: `right`	*Mögliche Werte: auto*, `inherit`, Prozentwert, numerischer Wert mit Einheit
CSS-Version: CSS 2, CSS 3, TV	*Zulässig für folgende (X)HTML-Elemente:* alle positionierten Elemente
Medium: Visual	*Vererbt:* Nein

Palm	NN	Mozilla		Internet Explorer							Opera			Safari		iCab	Kq	FF
2.x	4.x	1.x	1.8	4.x	5.0x (Win)	5.0 (Mac)	5.1+ (Mac)	5.5 (Win)	6	7	6	7	8	1.x	2.0	2.9+	3.x	1.0
●		●	●			●	○[1]	●	●	●	●	●	●	●	●	●[2]	●	●

1 Bei Angabe von `right` und `left` werden beide Werte berücksichtigt, aber nicht korrekt dargestellt.
2 ab Version 3.0

Sie finden das Beispiel auf der Buch-CD in der Datei BSP/K05/right.html.

Bei relativer Positionierung bezeichnet der angegebene Wert die Änderung gegenüber der normalen rechten Position. Geben Sie beispielsweise einen positiven Wert an, wird das Element um den entsprechenden Wert nach links verrückt.

Das bedeutet, dass das Element über den linken Rand des Viewports hinausragt, falls es die volle Breite des Viewports einnimmt. Folgender Stil zeigt dies. Er legt fest, dass der damit formatierte Absatz um 20 Pixel nach links gerückt wird. Da er die volle Breite einnimmt, sind die ersten 20 Pixel des Absatzes nicht sichtbar.

```
.eingerueckt { position:relative;
               right:20px; top:10em }
```

Abbildung 5.32: Der Opera-Browser 8 stellt die right-Eigenschaft korrekt dar.

Mit Hilfe von absoluter Positionierung können Sie die `right`- und `left`-Eigenschaft beispielsweise für eine zweispaltige Darstellung nutzen. Basis dafür sind zwei CSS-Klassen, `Spalte1` und `Spalte2`. Die erste wurde 1% von links (`left:1%`), die zweite 1% von rechts (`right:1%`) positioniert. Da beide Spalten eine Breite von 48% haben, entsteht zwischen den Spalten eine Lücke von 2%. Der hier definierte Rahmen (`border:1px solid silver`) dient lediglich dazu, die Spalten besser sichtbar zu machen.

Listing 5.15: Zwei Spalten mit absoluter Positionierung definieren

```
.spalte1 {
    position:absolute;
    left:1%;
    width:48%;
    border:1px solid silver;
    top:7em
}
.spalte2 {
    position:absolute;
    right:1%;
    width:48%;
    border:1px solid silver;
    top:7em
}
```

Wie bei der `left`*-Eigenschaft beziehen sich prozentuale Angaben auf die Breite des umgebenden Elements.*

Sichtbarkeit (visibility)

Mit dem Wert der Eigenschaft `visibility` können Sie steuern, ob und wie ein Element angezeigt werden soll.

Auch mit der `display`*-Eigenschaft können Sie Elemente ausblenden. Der Unterschied besteht aber darin, dass die mit* `visibility` *ausgeblendeten Boxen ihre Größe und Position behalten und damit auch Einfluss auf die Positionierung der übrigen Seitenelemente haben. Bei* `display:none` *würde die Box gar nicht erzeugt werden.*

Details zur `display`*-Eigenschaft finden Sie im Abschnitt »Anzeigeart (display)«.*

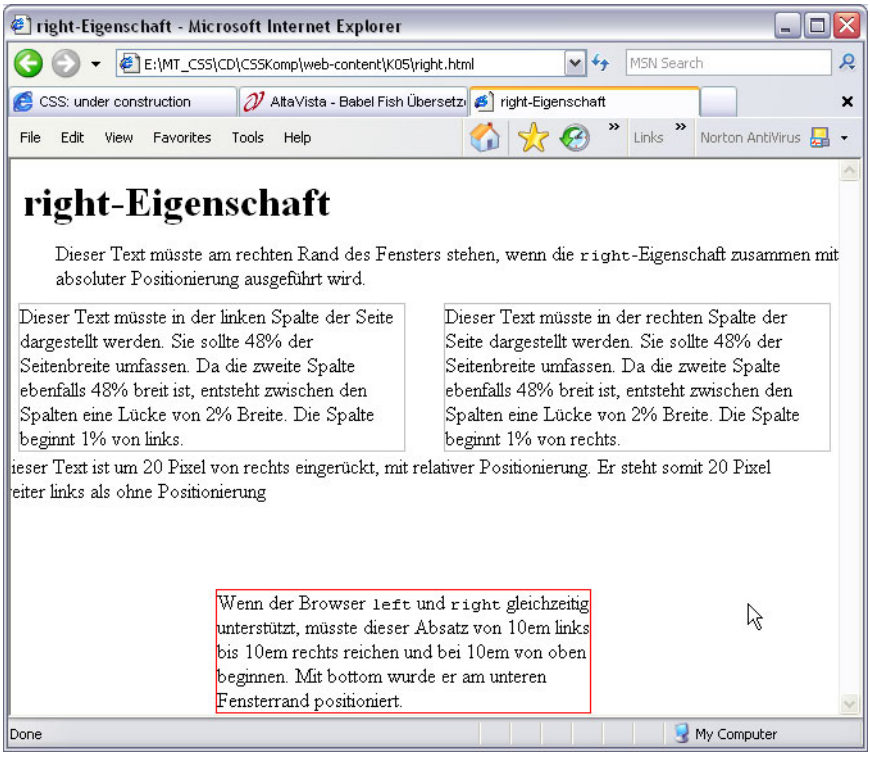


Abbildung 5.33: Auch der Internet Explorer 6 stellt die Testseite korrekt dar.

CSS-Element: `visibility`	*Mögliche Werte: inherit*, `visible`, `hidden`, `collapse`[1]
CSS-Version: CSS 2, CSS 3, TV, Mobile	*Zulässig für folgende (X)HTML-Elemente:* alle
Medium: Visual	*Vererbt:* Nein; in CSS 2.1 wird die Eigenschaft jedoch vererbt

Palm	NN	Mozilla		Internet Explorer							Opera			Safari		iCab	Kq	FF
2.x	4.x	1.x	1.8	4.x	5.0x (Win)	5.0 (Mac)	5.1+ (Mac)	5.5 (Win)	6	7	6	7	8	1.x	2.0	2.9+	3.x	1.0
○ 3	○ 2+3	●	●	○ 3	○ 3	○ 3	○ 3	○ 3	○ 3	○ 3	○ 3	○ 3	○ 3	○ 3	○ 3	○ 3+4	○ 3	●

1 nur auf Tabellenzeilen oder Tabellenspalten anwendbar
2 nur bei absolut positionierten Elementen
3 ohne den Wert `collapse`
4 ab Version 3.0

Mögliche Werte

- `visible`: In diesem Fall ist die Box sichtbar, das Element wird also angezeigt.
- `hidden`: Die Box ist unsichtbar. Das heißt, sie wird vollständig transparent angezeigt. Mit der folgenden Anweisung können Sie somit einen Absatz ausblenden.

```
<p style="visibility:hidden">Falls dieser Text sichtbar ist, wird
der Wert <code>hidden</code> nicht für unpositionierte Absätze
unterstützt.</p>
```

- `collapse`: blendet Tabellenzeilen oder Spalten aus. Dadurch wird nicht nur der Zellbereich nicht dargestellt, sondern auch der Platz für den Zellbereich eingespart. Die nachfolgende Spalte/Zeile wird also entsprechend aufgerückt.

INFO

Wenden Sie `visibility:collapse` *auf andere Elemente als Zeilen oder Spalten an, wirkt sich das wie* `visibility:hidden` *aus.*

CD

Sie finden das Beispiel auf der Buch-CD in der Datei BSP/K05/visibility.html.

Abbildung 5.34: So sollte die Beispielseite von standardkonformen Browsern angezeigt werden.

Insbesondere beim Wert `collapse` verhalten sich die Browser einheitlich falsch bzw. unzureichend.

Die Opera-Browser 6-8, Safari 1.0+, der Palm-Browser, iCab 3.0 und die Mac-Version des Internet Explorers stellen `visibility:collapse`, angewendet auf das `tr`-Element, so dar wie der Wert `hidden`. Das heißt, die Zeile ist nicht sichtbar, der dafür notwendige Platz bleibt aber frei. Alle diese Browser stellen `visibility:collapse`, angewendet auf Tabellenspalten (`col`-Element), gar nicht dar.

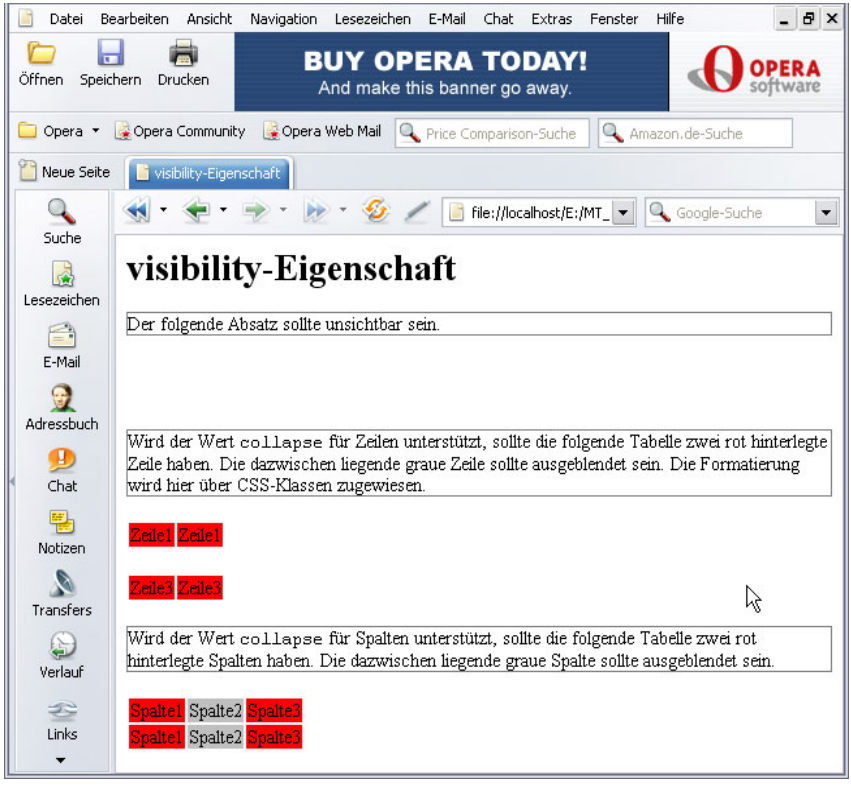


Abbildung 5.35: Fehlerhafte Darstellung der collapse-Eigenschaft

Beim Palm-Browser kann man davon ausgehen, dass die fehlende Darstellung von `visibility:collapse` in Tabellenspalten daran liegt, dass der Browser das `col`-Element nicht unterstützt.

Beispiel

Das folgende Beispiel soll die Verwendung des Wertes `collapse` zeigen. Es definiert zwei Tabellen. In der ersten werden die beiden CSS-Klassen auf die Zeilen der Tabelle angewendet, wodurch die mittlere ausgeblendet werden sollte.

Bei der zweiten Tabelle wird mithilfe der Elemente `colgroup` und `col` vor der eigentlichen Tabelle der Spaltenaufbau definiert und dort der mittleren Spalte ebenfalls die CSS-Klasse `grauunsichtbar` zugewiesen.

Listing 5.16: Beispielcode für den Wert collapse

```
        ...
        <style type="text/css">
        <!--
            p  {
               border:1px solid gray
            }
            .rotsichtbar {
                background-color:red;
                visibility:visible
            }
            .grauunsichtbar {
                background-color:silver;
                visibility:collapse
            }
        -->
        </style>
    </head>
    <body>
    <h1>visibility-Eigenschaft</h1>
    <p>Wird der Wert <code>collapse</code> für Zeilen unterst&uuml;tzt,
    sollte die folgende Tabelle zwei rot hinterlegte Zeile haben. Die
    dazwischen liegende graue Zeile sollte ausgeblendet sein. Die
    Formatierung wird hier über CSS-Klassen zugewiesen.</p>
    <table>
        <tr class="rotsichtbar"><td>Zeile1</td><td>Zeile1</td></tr>
        <tr class="grauunsichtbar">
            <td>Zeile2</td><td>Zeile2</td>
        </tr>
        <tr class="rotsichtbar"><td>Zeile3</td><td>Zeile3</td></tr>
    </table>
    <p>Wird der Wert <code>collapse</code> f&uuml;r Spalten
    unterst&uuml;tzt, sollte die folgende Tabelle zwei rot hinterlegte
    Spalten haben. Die dazwischen liegende graue Spalte sollte
    ausgeblendet sein.</p>
    <table>
        <colgroup>
            <col class="rotsichtbar">
            <col class="grauunsichtbar">
            <col class="rotsichtbar">
        </colgroup>
        <tr>
            <td>Spalte1</td>
            <td>Spalte2</td>
            <td>Spalte3</td>
        </tr>
        <tr>
            <td>Spalte1</td>
            <td>Spalte2</td>
            <td>Spalte3</td>
        </tr>
    </table>
</body>
```

Stapelreihenfolge (z-index)

Die Stapelreihenfolge legt fest, in welcher Reihenfolge sich Elemente überlappen, die Sie so positioniert haben, dass sie übereinander liegen.

CSS-Element: `z-index`	*Mögliche Werte: auto*, numerischer Wert, `inherit`
CSS-Version: CSS 2, CSS 3, TV	*Zulässig für folgende (X)HTML-Elemente:* alle positionierten Elemente
Medium: Visual	*Vererbt:* Nein

Palm	NN	Mozilla		Internet Explorer							Opera			Safari		iCab	Kq	FF
2.x	4.x	1.x	1.8	4.x	5.0x (Win)	5.0 (Mac)	5.1+ (Mac)	5.5 (Win)	6	7	6	7	8	1.x	2.0	2.9+	3.x	1.0
◎[1]	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●[2]	●	●

[1] nur bei absoluter Positionierung
[2] ab Version 3.0

Sie finden das Beispiel auf der Buch-CD in der Datei BSP/K05/z_index.html.

Standardmäßig hat die `z-index`-Eigenschaft den Wert `auto`. In diesem Fall werden die Elemente in der Reihenfolge gestapelt, in der sie im Dokument definiert sind. Je früher Sie ein Element angeben, desto weiter unten liegt die Ebene in der Stapelreihenfolge. Folgendes Beispiel soll dies demonstrieren. Es definiert zwei CSS-Klassen, die zwei `div`-Elementen zugewiesen werden. Mit diesen CSS-Klassen werden die Elemente ausschließlich positioniert und ihre Hintergrundfarbe (`background-color`) festgelegt.

Listing 5.17: Die CSS-Klassen zur Positionierung

```
.ebene1 {
    position:absolute;
    top:150px;
    left:10px;
    width:200px;
    height:200px;
    background-color:rgb(220,220,220)
}
.ebene2 {
    position:absolute;
    top:250px;
    left:20px;
    width:200px;
    height:200px;
```

```
    background-color:silver
}
```

Die Hintergrundfarbe ist ausschließlich deshalb notwendig, damit die Stapelreihenfolge sichtbar wird. Elemente ohne Hintergrund werden nämlich transparent dargestellt. Sie können dann schlecht erkennen, welches Element im Hintergrund und welches im Vordergrund liegt.

Listing 5.18: Anwenden der Klassen auf die HTML-Elemente

```
<body>
    <h1>z-index-Eigenschaft</h1>
    <div class="ebene1">Dieser Kasten wurde zuerst definiert.</div>
    <div class="ebene2">Dieser Kasten ist der zweite.</div>
    <p>Dieser Text wurde nicht positioniert.</p>
</body>
```

Durch die Standard-Stapelreihenfolge bewirkt diese Formatierung, dass das zuerst definierte `div`-Element unter dem danach definierten Element liegt.

Abbildung 5.36: Darstellung der Überlappung im Internet Explorer

Über die `z-index`-Eigenschaft können Sie diese Stapelreihenfolge ändern. So legen Sie fest, auf welcher Ebene ein Element angezeigt werden soll. Zulässig sind ganze Zahlen als Werte.

Je höher der Wert der `z-index`-Eigenschaft ist, desto weiter oben in der Stapelreihenfolge liegt das Element. Ein Element mit dem Wert 2 überlagert daher ein Element mit dem Wert 1, auch wenn es zuerst definiert wurde. Ergänzen Sie beispielsweise die beiden CSS-Klassen aus dem vorherigen Bei-

spiel um die z-index-Angabe, werden die beiden div-Elemente in umgekehrter Reihenfolge dargestellt.

```
.ebene1 { z-index:5; position:absolute;… }
.ebene2 { z-index:3; position:absolute;… }
```

Abbildung 5.37: Nun überlagert der erste Kasten den zweiten.

Natürlich können Sie die z-index-Eigenschaft auch auf relativ positionierte Elemente anwenden.

INFO

Der Palm-Browser stellt die z-index-Eigenschaft nur für absolut positionierte Elemente korrekt dar. Bei relativer Positionierung wird sie ignoriert.

Die z-index-Eigenschaft bestimmt die Stapelreihenfolge eines Elements immer innerhalb des aktuellen Stapelkontextes. Sobald Sie die z-index-Eigenschaft anwenden, wird dieser Stapelkontext erzeugt. Das bedeutet, dass das umgebende Element den z-index-Wert 0 erhält. Dadurch kann es durchaus passieren, dass Elemente mit niedrigeren z-index-Werten andere Elemente mit höheren Werten überlagern. Das passiert nämlich dann, wenn sie sich in anderen Stapelkontexten befinden. Folgendes Beispiel soll das demonstrieren.

INFO

Sie finden das Beispiel auf der Buch-CD in der Datei BSP/K05/z_index2.html.

Im Unterschied zum vorherigen Beispiel umgibt dieser Code die absolut und relativ positionierten `div`-Elemente mit je einem weiteren `div`-Element, das absolut positioniert ist. Die zuerst definierte Ebene erhält über die CSS-Klasse `ebeneB` den `z-index`-Wert 50. Die später definierte Ebene erhält den `z-index`-Wert 100. Die erste Ebene liegt also unter der zweiten. Dies entspricht der normalen Stapelreihenfolge.

Innerhalb der beiden Ebenen befinden sich die absolut bzw. relativ positionierten Elemente. Diese haben im Element mit der Klasse `ebeneA` einen niedrigeren Wert als in der Ebene mit der Klasse `ebeneB`. Dennoch überlagern die untergeordneten `div`-Elemente mit den niedrigeren `z-index`-Werten die beiden Elemente mit den höheren. Das liegt daran, dass die `z-index`-Eigenschaft nur die Stapelreihenfolge zwischen gleichrangigen Elementen regelt.

Gleichrangig sind Elemente immer dann, wenn sie sich innerhalb des gleichen übergeordneten Elements befinden.

Der `z-index`-Wert wird immer relativ zur Ebene des übergeordneten Elements berechnet. Hat dieses wie im Beispiel die Ebene 100 und das untergeordnete Element die Ebene 1, müsste das Element den Wert 101 bekommen, wenn Sie auf die übergeordneten Container verzichten würden. Die `z-index`-Werte von 50 und 100 im Beispiel regeln also die Stapelreihenfolge der Container. Dadurch liegt der zweite Container über dem ersten. Innerhalb des Containers spielt die Reihenfolge der Container dann keine Rolle mehr.

Listing 5.19: Durch die Positionierung übergeordneter Container können Elemente mit niedrigem z-index-Wert die Elemente mit höheren Werten überlagern.

```
<head>
    <title>z-index-Eigenschaft
        (Stapelkontext)</title>
    <style type="text/css">
    <!--
        .ebeneA {
            position:absolute;
            top:50;
            left:0;
            z-index:100;
            border:1px solid black;
            height:500px;
            width:500px;
        }
        .ebeneB {
            position:absolute;
            top:50;
            left:50;
            z-index:50;
            border:1px solid black;
```

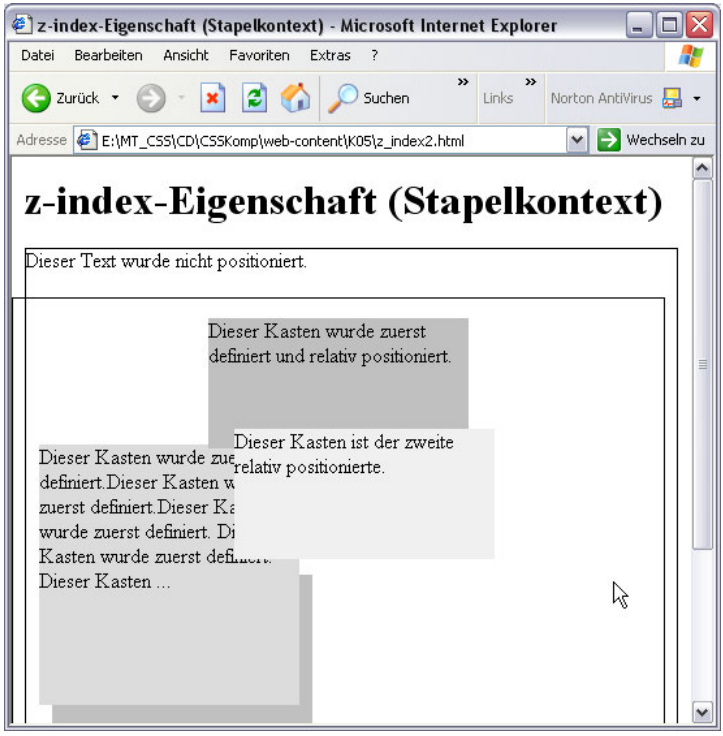


Abbildung 5.38: Das Ergebnis des Beispiels

```
    height:500px;
    width:500px;
}
.ebene1 {
    z-index:5;
    position:absolute;
    top:150px;
    ...
}
.ebene2 {
    z-index:3;
    position:absolute;
    top:250px;
    ...
}
.ebene3 {
    position:relative;
    z-index:1;
    width:200px;
    ...
}
.ebene4 {
    position:relative;
    z-index:2;
    width:200px;
    ...
```

```
        }
    -->
    </style>
</head>

<body>
    <h1>z-index-Eigenschaft (Stapelkontext)</h1>
    <div class="ebeneB">
        <div class="ebene1">Dieser Kasten wurde
        ...
        definiert. Dieser Kasten ...</div>
        <div class="ebene2">Dieser Kasten ist der
        zweite.</div>
    </div>
    <p>Dieser Text wurde nicht positioniert.</p>
    <div class="ebeneA">
        <div class="ebene3">Dieser Kasten wurde
        zuerst definiert und relativ
        positioniert.</div>
        <div class="ebene4">Dieser Kasten ist der
        zweite relativ positionierte.</div>
    </div>
</body>
```

Textumfluss (float)

Mit der `float`-Eigenschaft können Sie ebenfalls Elemente positionieren, und zwar indem Sie festlegen, ob und wie die folgenden Inhalte das Element umfließen sollen.

CSS-Element: `float`	*Mögliche Werte: none*, `left`, `right`, `inherit`
CSS-Version: CSS 1-3, TV, Mobile	*Zulässig für folgende (X)HTML-Elemente:* alle, außer absolut positionierte
Medium: Visual	*Vererbt:* Nein

Palm	NN	Mozilla		Internet Explorer							Opera			Safari		iCab	Kq	FF
2.x	4.x	1.x	1.8	4.x	5.0x (Win)	5.0 (Mac)	5.1+ (Mac)	5.5 (Win)	6	7	6	7	8	1.x	2.0	2.9+	3.x	1.0
○		●	●	○	○	● [1]	● [1]	○	●	●	● [1]	● [1]	●	● [2]	● [2]	● [3]	● [2]	●

1 Floating-Boxen werden weiter oben dargestellt als das umfließende Element.
2 Floating-Boxen werden eine Idee zu weit unten dargestellt.
3 ab Version 3.0

Diese Eigenschaft gibt an, ob eine Box nach links, rechts oder überhaupt nicht gleiten soll. Sie kann für alle Elemente gesetzt werden, die Boxen erzeugen, die nicht absolut positioniert sind. Die Werte dieser Eigenschaft haben die folgenden Bedeutungen:

Die Werte `left` und `right` bewirken, dass die durch das Element erzeugte Box links (`left`) oder rechts (`right`) ausgerichtet wird und die folgenden Elemente auf der jeweils anderen Seite um die Box laufen.

Solche Boxen werden als Floating-Boxen bezeichnet.

`none` ist der Standardwert, der bewirkt, dass das Element nicht umflossen wird.

Sie finden das Beispiel auf der Buch-CD in der Datei BSP/K05/float.html.

Definieren Sie mit

Listing 5.20: Beispielcode zur Demonstration der float-Eigenschaft

```
<body>
    <h1>float-Eigenschaft</h1>
    <div style="float:left;width:120px;
        border:1px solid gray;margin:5px;
        padding:10px">
         Dieser Kasten sollte links ausgerichtet
         sein.</div>
    <div style="float:right;width:120px;
        border:1px solid red;margin:5px;
        padding:10px">Dieser
         Kasten sollte rechts ausgerichtet
         sein.</div>
    <p>Dieser Text umfließt die beiden Kästchen,
    dieser Text umfließt die beiden Kästchen, dieser
    Text umfließt die beiden Kästchen, dieser Text
    umfließt die beiden Kästchen, dieser Text
    umfließt die beiden Kästchen, dieser Text
    umfließt die beiden Kästchen, dieser Text
    umfließt die beiden Kästchen.</p>
</body>
```

zwei `div`-Elemente und ein `p`-Element. Beide `div`-Elemente werden mit `float` ausgerichtet, sodass der Inhalt des Absatzes beide `div`-Elemente in der Mitte umfließt.

Generell gilt, dass Floating-Boxen so hoch wie möglich positioniert werden sollen. Das ist auch der Grund, warum beide Floating-Boxen im Beispiel auf

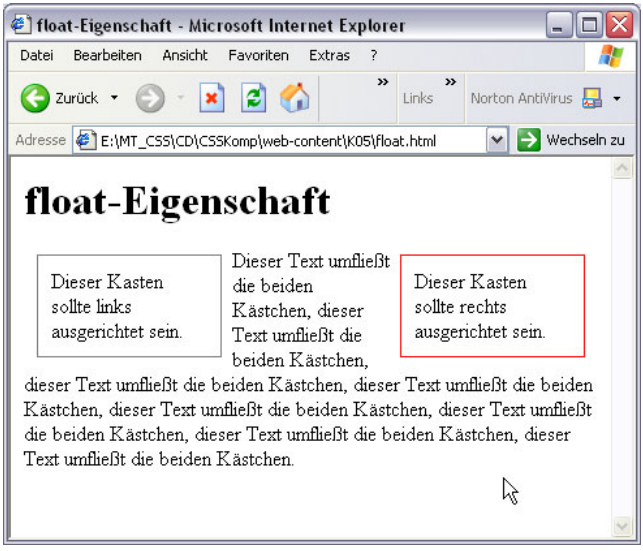


Abbildung 5.39: Als Ergebnis erhalten Sie zwei Boxen mit umfließenden Text.

gleicher Höhe angezeigt werden, obwohl sie nacheinander im Quellcode angegeben sind.

Den Abstand zum umfließenden Text können Sie mit Hilfe der `margin`-Eigenschaft festlegen.

:-) TIPP

Wenn Sie mehrere Floating-Boxen erstellen, die Sie alle links oder alle rechts ausrichten, werden diese alle nebeneinander dargestellt. Das ist die perfekte Methode, um Links als Navigationsbuttons nebeneinander anzuordnen.

REF

Ein Beispiel dazu finden Sie weiter unten im Abschnitt »Praxisbeispiel«.

palm

Der Palm-Browser stellt Floating-Boxen korrekt dar, solange der Platz neben den Boxen ausreicht, um den umfließenden Inhalt darzustellen. Sollte das nicht der Fall sein, wird der Text in die Floating-Box geschoben.

Der Internet Explorer 4 und 5 für Windows ignoriert innerhalb von Floating-Boxen die `Padding`-Eigenschaft.

\+ Mac

Der Opera-Browser in der Version 7.x und niedriger sowie der Internet Explorer für Mac stellen die Floating-Boxen 1-2 Pixel zu hoch dar. Die

Inhalte der Floating-Boxen stehen damit 1-2 Pixel höher als der umfließende Inhalt.

Abbildung 5.40: Die gepunktete gelbe Linie zeigt den Höhenunterschied.

!! STOP

Der Internet Explorer stellt die `float`*-Eigenschaft im Prinzip dar. In seltenen Fällen kann es jedoch in Verbindung mit der* `clear`*-Eigenschaft zu Darstellungsfehlern kommen und zwar immer dann, wenn Elemente mit* `clear:left` *und* `float:left` *untereinander positioniert werden und das umgebende Blockelement eine bestimmte Mindestgröße unterschreitet. Diese Mindestgröße kann man allerdings nicht nachvollziehen, weil die Breite immer noch ausreichend ist, um die Elemente vollständig darzustellen.*

Untere Position (bottom)

Die Eigenschaft `bottom` definiert, an welcher Position von unten ein Element dargestellt wird. Dabei gibt der Wert an, wie weit die untere Kante des Inhalts über der unteren Kante der umgebenden Box positioniert wird. Ist die umgebende Box die umgebende Ausgangsbox, wird der Wert von `bottom` relativ zum unteren Rand des Viewports berechnet. Ob der Wert von `bottom` als absolute Position oder relativ zu anderen Boxen interpretiert wird, hängt vom Wert der Eigenschaft `position` ab.

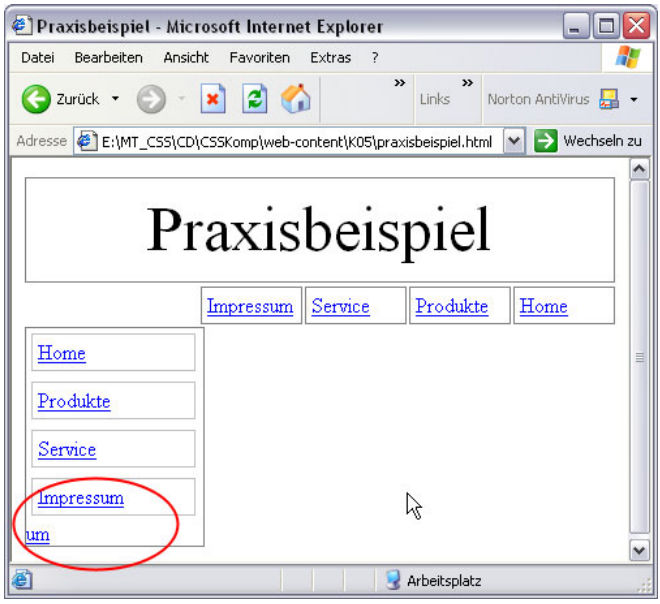


Abbildung 5.41: Darstellungsfehler. Der Internet Explorer zeigt den letzten Teil der Floating-Box unterhalb der Box zusätzlich an.

INFO

Die Eigenschaft darf nur für positionierte Elemente angewendet werden. Als positioniert gilt ein Element dann, wenn seine `position`*-Eigenschaft einen Wert ungleich* `static` *hat.*

REF

Näheres zur Positionierung finden Sie im Abschnitt über die `position`*-Eigenschaft.*

CSS-Element: `bottom`	*Mögliche Werte: auto*, `inherit`, Prozentwert, numerischer Wert mit Einheit
CSS-Version: CSS 2, CSS 3, TV	*Zulässig für folgende (X)HTML-Elemente:* alle positionierten Elemente
Medium: Visual	*Vererbt:* Nein

Palm	NN	Mozilla		Internet Explorer							Opera			Safari		iCab	Kq	FF
2.x	4.x	1.x	1.8	4.x	5.0x (Win)	5.0 (Mac)	5.1+ (Mac)	5.5 (Win)	6	7	6	7	8	1.x	2.0	2.9+	3.x	1.0
		●	●			● [1]	● [1]	● [1]	● [1]	● [1]	●	●	●	●	●	● [2]	●	●

[1] Bei Angabe von `top` und `bottom` wird der Wert für `bottom` ignoriert.
[2] ab Version 3.0

Die vertikale Position einer Box können Sie auch mit Hilfe der `top`*-Eigenschaft definieren. Sie legt die Position der Box von oben fest. Grundsätzlich ist es möglich, dass Sie gleichzeitig* `bottom` *und* `top` *angeben. Allerdings sollten Sie dann für die Höhe des Elements keinen Wert vorgeben. Generell ist es ratsam, nur einen Wert anzugeben, weil beispielsweise der Internet Explorer 6 und der Internet Explorer 5 (Mac) den Wert* `bottom` *ignorieren, wenn Sie* `top` *gleichzeitig angeben.*

Geben Sie als Wert für die `bottom`*-Eigenschaft einen prozentualen Wert an, bezieht sich der immer auf die Höhe des umgebenden Elements.*

Folgendes Beispiel soll die Verwendung der `bottom`-Eigenschaft demonstrieren. In der Seite werden neben der Seitenüberschrift drei Absätze definiert. Der erste wird mit der CSS-Klasse `.unten` am unteren Seitenrand positioniert.

```
.unten { position:absolute; bottom:0px }
```

Dazu wird zunächst die `position`-Eigenschaft auf `absolute` gesetzt und anschließend die `bottom`-Eigenschaft auf 0 gesetzt. Dadurch beginnt der untere Rand der Box am unteren Rand des Inhalts des umgebenden Blocks. Hier ist dies der Block des `body`-Elements.

Der zweite Absatz wird mit der CSS-Klasse `.weiterunten` formatiert.

```
.weiterunten { position:relative; bottom:2em }
```

Mit der Angabe `position:relative` wird der Absatz nun relativ positioniert. In diesem Fall bedeutet dies, dass der Absatz normalerweise unterhalb des `h1`-Elements stehen würde, da absolut positionierte Elemente wie der erste Absatz aus dem normalen Fluss der Seite herausfallen. Das zweite `p`-Element hätte also einen ganz normal großen Abstand zur Überschrift. Durch die Angabe `bottom: 2em` wird der untere Rand des Absatzes nun aber um `2em` nach oben verschoben. Dadurch wird der Absatz fast ohne Abstand unterhalb der Überschrift angezeigt.

Für den dritten Absatz, der ebenfalls absolut positioniert wird, wird sowohl ein Wert für `top` als auch für `bottom` angeben und damit die Position von unten und von oben festgelegt. Aus der Differenz ergibt sich damit gleichzeitig die Höhe des Elements.

```
.topbottom { position:absolute;
             bottom:5em; top:5em;
             border: 1px solid red }
```

Zur Kennzeichnung der Elementgröße sorgt `border:1px solid red` für einen roten Rahmen um den Absatz.

Um kenntlich zu machen, wie groß die eigentliche Seite, das heißt der Block des `body`*-Elements ist, wird auch das* `body`*-Element mit einem Rahmen umgeben. Dafür sorgt der Stil:*

```
body { border: 1px solid gray }
```

Sie finden das Beispiel auf der Buch-CD in der Datei BSP/K05/bottom.html.

Der Internet Explorer 5+ für Mac sowie die Internet Explorer 5.5 und 6 für Windows ignorieren den Wert für `bottom`, wenn Sie sowohl `bottom` als auch `top` angeben.

Abbildung 5.42: Korrekt sollte die Seite wie hier in Netscape 7.1 aussehen.

Listing 5.21: Aufbau der Beispielseite

```
<body>
    <h1>bottom-Eigenschaft</h1>
    <p class="unten">Dieser Text müsste am unteren Rand des
    Fensters stehen, wenn die <code>bottom</code>-Eigenschaft
    zusammen mit absoluter Positionierung ausgeführt wird.</p>
    <p class="weiterunten">Falls der Browser die <code>bottom</code>-
    Eigenschaft zusammen mit relativer Positionierung unterstützt,
    sollte dieser Text knapp unterhalb der Überschrift stehen.</p>
    <p class="topbottom">Wenn der Browser <code>bottom</code> und
    <code>top</code> gleichzeitig unterstützt, müsste dieser Absatz
    von 5em oben, bis 5em unten reichen.</p>
</body>
```

Überlauf (overflow)

Mit dieser Eigenschaft können Sie festlegen, wie mit dem Inhalt einer Box umgegangen werden soll, der über die Grenzen der sie umschließenden Box hinausgeht.

CSS-Element: `overflow`	*Mögliche Werte: visible*, `hidden`, `scroll`, `auto`, `inherit`
CSS-Version: CSS 2, CSS 3	*Zulässig für folgende (X)HTML-Elemente:* alle Elemente auf Blockebene und alle ersetzten Elemente; in CSS 2.1 nicht für verschobene Blockelemente, Tabellenzellen und Inline-Blockelemente
Medium: Visual	*Vererbt:* Nein

Palm	NN	Mozilla		Internet Explorer							Opera			Safari		iCab	Kq	FF
2.x	4.x	1.x	1.8	4.x	5.0x (Win)	5.0 (Mac)	5.1+ (Mac)	5.5 (Win)	6	7	6	7	8	1.x	2.0	2.9+	3.x	1.0
○ 3		●	●	● 1	● 1	⊙ 2	⊙ 2	● 1	● 1	● 1	●	●	●	●	●	● 4	●	●

1 Beim Wert `visible` wird die umgebende Box so vergrößert, dass der ganze Inhalt sichtbar ist.
2 Der Wert `visible` wird fehlerhaft dargestellt.
3 kein `scroll` und kein `auto`
4 ab Version 3.0

mögliche Werte

- `visible`: bestimmt, dass der Inhalt nicht abgeschnitten, sondern komplett angezeigt wird. Das ist die Standardeinstellung.
- `hidden`: Der überlaufende Inhalt wird abgeschnitten, ohne dass der Benutzer diesen beispielsweise durch Scrollen anzeigen kann. Um zu bestimmen, wie groß der Bereich sein soll, der abgeschnitten wird, verwenden Sie die `clip`-Eigenschaft.

Mehr zur Eigenschaft `clip` *finden Sie im Abschnitt »Ausschnitt (clip)«.*

- `scroll`: Der Inhalt wird abgeschnitten und der Browser kann (muss aber nicht) Möglichkeiten zur Verfügung stellen, um den nicht sichtbaren Bereich einzusehen. Dazu kann beispielsweise eine Scrollleiste angezeigt werden. Falls das Zielmedium für die visuelle Ausgabe ein `print` ist, wird bei diesem Wert der abgeschnittene Inhalt gedruckt.

- `auto`: Der Browser entscheidet, wie er mit dem überlaufenden Inhalt umgeht. Er sollte dabei jedoch einen Mechanismus zum Scrollen zur Verfügung stellen.

Beispiel

Um den Überlauf zu testen, benötigen Sie ein Blockelement, für das Sie eine feste Größe definiert haben, und ein darin enthaltenes Blockelement, dessen Inhalt größer ist, als in der umgebenden Box dargestellt werden kann.

Sie finden das Beispiel auf der Buch-CD in der Datei BSP/K05/overflow.html.

Ein solcher HTML-Code könnte wie folgt aussehen:

Listing 5.22: Aufbau einer Inhalts-Box zum Testen der overflow-Eigenschaft

```
<div class="umgebung">
    <div class="kastenauto">
    Dieser Kasten wurde mit <code>overflow:auto</code> formatiert!
    Der abgeschnittene Inhalte sollte zug&auml;nglich sein.
    </div>
</div>
```

Das innere `div`-Element muss nun nur noch soviel Inhalt haben, dass die Größe des äußeren `div`-Elements nicht ausreicht, um ihn darzustellen. Das erreichen Sie bei wenig Text am einfachsten, indem Sie die Schriftgröße erhöhen. Damit die beiden Boxen sichtbar sind, definieren die Stile in der Beispielseite zusätzlich einen grauen Rahmen für das äußere und einen roten Rahmen für das innere `div`-Element.

Sie sollten keinesfalls die Größe der inneren Box über `width` *und* `height` *größer festlegen als der Innenraum der umgebenden Box. Dann erzwingen alle Opera- und Mozilla-Browser die Größe der inneren Box und zeigen den Überlauf an, auch wenn Sie mit* `overflow` *eine andere Einstellung festlegen. Einen Überlauf gibt es dann nämlich unter Umständen nicht. Eine Größe für die innere Box müssen Sie aber angeben. Lediglich der Internet Explorer 6 für Windows zeigt die innere Box auch dann korrekt gemäß der Einstellungen für* `overflow` *an, wenn Sie deren Größe zu groß festlegen.*

Listing 5.23: Stile zum Testen der einzelnen Werte

```
<style type="text/css">
<!--
    .umgebung {
        float:left;
        width:100px; height:100px;
        border:1px solid gray;
        font-size:1.5em;
        padding:1px;
        margin:10px;
    }
    .kastenscroll {
        border:1px solid red;
```

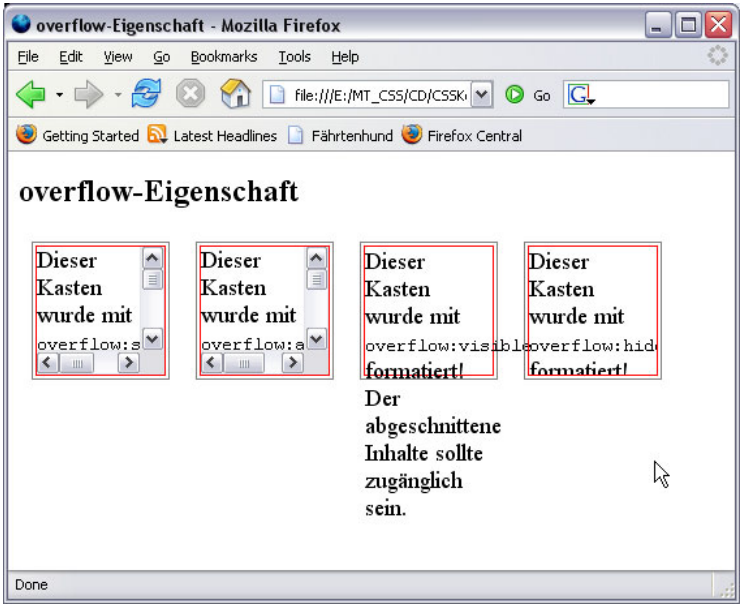


Abbildung 5.43: Ein mögliches, korrektes Ergebnis, hier in Firefox

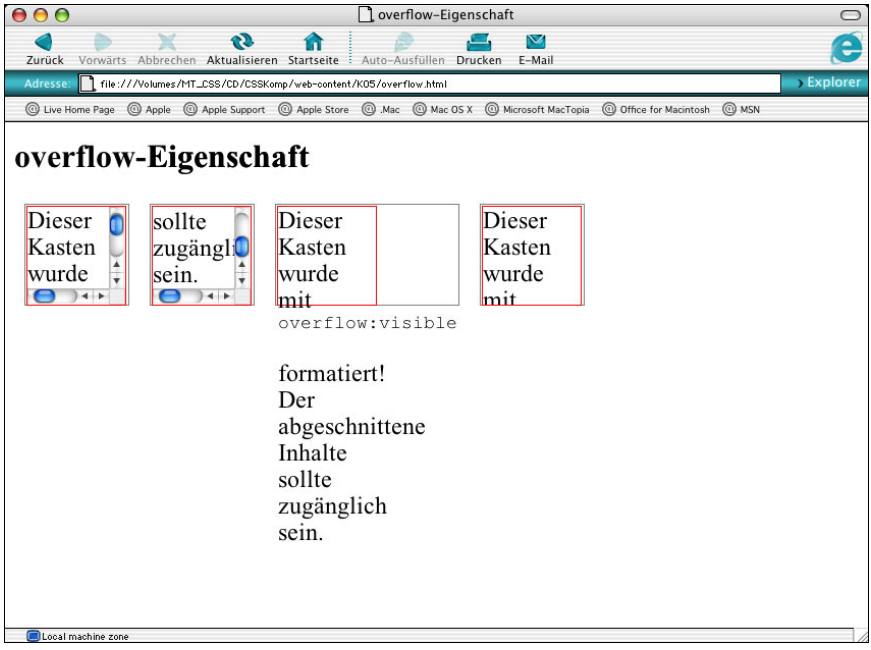
```
        overflow:scroll;
        width:98px; height:98px;
    }
    .kastenauto {
        border:1px solid red;
        overflow:auto;
        width:98px; height:98px;
    }
    .kastenhidden {
        border:1px solid red;
        overflow:hidden;
        width:98px; height:98px;
    }
    .kastenvisible {
        border:1px solid red;
        overflow:visible;
        width:98px; height:98px;
    }
-->
</style>
```

Alle Browser stellen das Element beim Wert `auto` genauso dar wie beim Wert `scroll`, d.h. mit horizontalen und vertikalen Bildlaufleisten.

Mac

Die Mac-Version des Internet Explorers zeigt gerade beim Wert `visible` ein sehr merkwürdiges Verhalten. Einerseits vergrößert sie wie die Windows-Version die umgebende Box. Allerdings nur in der Breite. Die innere Box behält hingegen ihre definierte Größe. Der Inhalt der inneren Box läuft dennoch über die innere Box und, nach unten, auch über die äußere Box hinaus.

Abbildung 5.44: Fehlerhafte Darstellung des Wertes visible



Der Palm-Browser unterstützt nur die Werte `visible` und `hidden`. Alle anderen Werte werden wie `hidden` dargestellt.

5.3 Nicht/schlecht unterstützte Attribute

In dieser Rubrik finden Sie solche Attribute, die von vielen oder zumindest von sehr wichtigen Browsern nicht oder nur fehlerhaft unterstützt werden und die somit noch nicht praxistauglich sind.

Maximale Breite (max-width)

Die Eigenschaft `max-width` definiert die maximale Breite eines Elements. Sie können `max-width` zusätzlich zur `width`-Eigenschaft oder zusätzlich zur `min-width`-Eigenschaft (minimale Breite) angeben. Prozentuale Werte beziehen sich generell auf die Breite des umgebenden Elements.

CSS-Element: `max-width`	*Mögliche Werte: none*, `inherit`, positiver numerischer Wert mit Einheit, Prozentwert
CSS-Version: CSS 2, CSS 3	*Zulässig für folgende (X)HTML-Elemente:* alle, außer nicht positionierte Inline-Elemente, Tabellenzeilen und Zeilengruppen
Medium: Visual	*Vererbt:* Nein

Palm	NN	Mozilla		Internet Explorer							Opera			Safari		iCab	Kq	FF
2.x	4.x	1.x	1.8	4.x	5.0x (Win)	5.0 (Mac)	5.1+ (Mac)	5.5 (Win)	6	7	6	7	8	1.x	2.0	2.9+	3.x	1.0
		●	●								●	●	●		●	●[1]	●	●

[1] ab Version 3.0

Sie finden das Beispiel auf der Buch-CD in der Datei BSP/K05/width.html.

Wenn Sie Schriften mit der Einheit `em` definieren, können Sie zusätzlich mit der `max-width`-Eigenschaft festlegen, dass die Box unabhängig von der Größe, die durch ihren Inhalt bestimmt wird, eine maximale Größe nicht überschreitet.

```
.max { max-width:100px }
```

Die exakte Breite eines Elements können Sie mit der Eigenschaft `width` *festlegen. Mehr dazu finden Sie im Abschnitt »Breite (width)«.*

Maximale Höhe (max-height)

Die Eigenschaft `max-height` definiert die maximale Höhe eines Elements. Sie können `max-height` zusätzlich zur `height`-Eigenschaft oder zusätzlich zur `min-height`-Eigenschaft (minimale Höhe) angeben. Prozentuale Werte beziehen sich generell auf die Höhe des umgebenden Elements.

CSS-Element: `max-height`	*Mögliche Werte: none*, `inherit`, positiver numerischer Wert mit Einheit, Prozentwert
CSS-Version: CSS 2, CSS 3	*Zulässig für folgende (X)HTML-Elemente:* alle, außer nicht positionierte Inline-Elemente, Tabellen und Tabellenzellen
Medium: Visual	*Vererbt:* Nein

Palm	NN	Mozilla		Internet Explorer							Opera			Safari		iCab	Kq	FF
2.x	4.x	1.x	1.8	4.x	5.0x (Win)	5.0 (Mac)	5.1+ (Mac)	5.5 (Win)	6	7	6	7	8	1.x	2.0	2.9+	3.x	1.0
		○ [2]	○ [2]									○ [2]	○ [2]		○ [2]	○ 2+1	○ [2]	○ [2]

[1] ab Version 3.0
[2] keine prozentualen Werte

Sie finden das Beispiel auf der Buch-CD in der Datei BSP/K05/height.html.

Wenn Sie Schriftgrößen mit der Einheit `em` definieren, können Sie zusätzlich mit der `max-height`-Eigenschaft festlegen, dass die Box unabhängig von der Größe, die durch ihren Inhalt bestimmt wird, eine maximale Größe nicht überschreitet.

```
.max { font-size:2em; max-height:12pt }
```

Die exakte Höhe eines Elements können Sie mit der Eigenschaft `height` *festlegen. Mehr dazu finden Sie im Abschnitt »Höhe (height)«.*

Minimale Breite (min-width)

Die Eigenschaft `min-width` definiert die Mindestbreite eines Elements. Sie können `min-width` zusätzlich zur `width`- oder zur `max-width`-Eigenschaft (maximale Breite) angeben.

CSS-Element: `min-width`	*Mögliche Werte: auto*[2], `inherit`, positiver numerischer Wert mit Einheit, Prozentwert
CSS-Version: CSS 2, CSS 3	*Zulässig für folgende (X)HTML-Elemente:* alle, außer nicht positionierte Inline-Elemente, Tabellenzeilen und Zeilengruppen
Medium: Visual	*Vererbt:* Nein

Palm	NN	Mozilla		Internet Explorer							Opera			Safari		iCab	Kq	FF
2.x	4.x	1.x	1.8	4.x	5.0x (Win)	5.0 (Mac)	5.1+ (Mac)	5.5 (Win)	6	7	6	7	8	1.x	2.0	2.9+	3.x	1.0
		●	●								●	●	●		●	●[1]	●	●

[1] ab Version 3.0
[2] In CSS 2.1 ist der Standardwert 0.

Sie finden das Beispiel auf der Buch-CD in der Datei BSP/K05/width.html.

Der Wert von `min-width` *hat in allen Browsern, die die Eigenschaft unterstützen, Vorrang vor* `max-width` *und* `width`*. Prozentuale Werte beziehen sich generell auf die Breite des umgebenden Elements.*

Die exakte Breite eines Elements können Sie mit der Eigenschaft `width` *festlegen. Mehr dazu finden Sie im Abschnitt »Breite (width)«. Dort finden Sie auch ein Beispiel zur* `min-width`*-Eigenschaft.*

Minimale Höhe (min-height)

Die Eigenschaft `min-height` definiert die Höhe eines Elements, die das Element mindestens haben soll. Sie können `min-height` zusätzlich zur `height`- oder zur `max-height`-Eigenschaft (maximale Höhe) angeben.

CSS-Element: `min-height`	*Mögliche Werte: auto*[2], `inherit`, positiver numerischer Wert mit Einheit, Prozentwert
CSS-Version: CSS 2, CSS 3	*Zulässig für folgende (X)HTML-Elemente:* alle, außer nicht positionierte Inline-Elemente, Tabellen und Tabellenzellen
Medium: Visual	*Vererbt:* Nein

Palm	NN	Mozilla		Internet Explorer							Opera			Safari		iCab	Kq	FF
2.x	4.x	1.x	1.8	4.x	5.0x (Win)	5.0 (Mac)	5.1+ (Mac)	5.5 (Win)	6	7	6	7	8	1.x	2.0	2.9+	3.x	1.0
		●	●									●	●	●	●	●[1]	●	●

[1] ab Version 3.0
[2] In CSS 2.1 ist der Standardwert 0.

Geben Sie prozentuale Werte an, beziehen diese sich immer auf die Höhe des umgebenden Elements.

Sie finden das Beispiel auf der Buch-CD in der Datei BSP/K05/height.html.

Sinnvoll einsetzen lässt sich die Eigenschaft vor allem, wenn Sie Schriften mit der Einheit `em` definieren. Dann können Sie zusätzlich mit der `min-height`-Eigenschaft festlegen, dass die Box unabhängig von der Größe, die durch ihren Inhalt bestimmt wird, eine minimale Größe hat.

```
.min { font-size:1em; min-height:24pt }
```

Die exakte Höhe eines Elements können Sie mit der Eigenschaft `height` *festlegen. Mehr dazu finden Sie im Abschnitt »Höhe (height)«.*

5.4 Praxisbeispiel

Sowohl absolut wie relativ positionierte Elemente und Floating-Boxen eignen sich sehr gut zum Layout von Seiten. Dieses Praxisbeispiel soll Ihnen zeigen, wie Sie diese CSS-Eigenschaften sinnvoll einsetzen können, um ein Layout für eine Webseite zu erstellen.

Zur Darstellung der Positionen einzelner Elemente wird jedes Element mit einem grauen Rahmen umgeben. Weitere Formatierungen, außer die zur Positionierung notwendigen, werden nicht gemacht. Sie können dem Beispielcode aber selbstverständlich auch noch Hintergrundfarben und Bilder, andere Rahmenstile etc. hinzufügen.

Sie finden das Beispiel auf der Buch-CD in der Datei BSP/K05/praxisbeispiel.html *und in der CSS-Datei.* formate.css.

Seitenbanner

Das Seitenbanner ist das einzige Element der Seite, das nicht zwingend positioniert werden muss, da es auch im normalen Fluss ganz oben auf der Seite erscheint. Es reicht also aus, dafür ein `p`-Element am Anfang des `body`-Elements einzufügen und mit einer eindeutigen ID zu versehen, damit Sie dafür die CSS-Formatierungen festlegen können.

Die Höhe von `1em` stellt sicher, dass der Text unabhängig von der Schriftgröße immer Platz hat. Damit aber dennoch etwas Abstand nach oben und unten bleibt, wird ein oberer und unterer Innenabstand mit `padding-top` und `padding-bottom` von `0.25em` festgelegt. Insgesamt hat das Element dann eine Höhe von 1,5em zuzüglich einem 1 Pixel starken Rahmen.

Listing 5.24: des Seitenbanners

```
<html>
    <head>
        <meta http-equiv="content-type" content="text/html;
        charset=iso-8859-1">
        <title>Praxisbeispiel</title>
        <style type="text/css">
        <!--
            #banner {
                border: 1px solid gray;
                height:1em;
                font-size:3em;
                padding-top:0.25em;
                padding-bottom:0.25em;
                text-align:center
            }
        -->
        </style>
    </head>
    <body>
        <p id="banner">Praxisbeispiel</p>
    </body>
</html>
```

Horizontale Navigationsleiste

Die horizontale Navigationsleiste wird üblicherweise unterhalb des Seitenbanners angezeigt. Sie kann beispielsweise als Floating-Box formatiert werden, indem Sie jedes `a`-Element als Floating-Box links ausrichten und in einem `div`-Element zusammenfassen. Letzteres gibt Ihnen bei Bedarf die Möglichkeit, die Navigationsleiste absolut oder relativ zu positionieren und etwas nach oben oder unten zu rücken. Im Beispiel wird darauf jedoch verzichtet. Zudem können Sie so ganz einfach nur Formatierungen für die Links der horizontalen Navigation definieren, ohne dass dies andere Links der Seite beeinflusst.

Listing 5.25: Benötigter HTML-Code für die horizontale Navigationsleiste

```
<p id="banner">Praxisbeispiel</p>
<div id="hornavigation">
    <a href="index.html">Home</a>
    <a href="produkte.html">Produkte</a>
    <a href="service.html">Service</a>
    <a href="impressum.html">Impressum</a>
</div>
```

Zur Ausrichtung der Elemente brauchen Sie nur zwei Stile, einen für alle Links der Navigationsleiste und einen für die Navigationsleiste selbst. Diese richten Sie mit `float:right` rechtsbündig aus und setzen sie auf eine Breite von `68%`. Mit `padding:0` löschen Sie den Standard-Innenabstand.

Der zweite Stil richtet die Hyperlinks der Navigationsleiste aus. Mit `float:left` richten Sie die Links linksbündig innerhalb des umgebenden `div`-Elements aus und definieren einen rechten Außenabstand von `0.5%` der Breite des umgebenden Elements. Den Abstand von oben legen Sie mit `margin-top` fest und definieren für oben und unten einen Innenabstand von `0.5em` und für links und rechts einen Innenabstand von `0.5%`, so dass zwischen dem mit `border:1px solid gray` definierten Rahmen und dem Text etwas Abstand verbleibt. Dieser ist in der Horizontalen umso größer, je größer die Fensterbreite ist; die vertikalen Abstände richten sich nach der Schriftgröße.

Die Breite der einzelnen Elemente wird auf `22,75%` festgelegt. Diesen Wert müssen Sie abhängig von der Anzahl Links anpassen, damit auch alle Platz haben.

Listing 5.26: Die Stile zur Formatierung der horizontalen Navigationsleiste

```
#hornavigation {
    width:68%;
    float:right;
    padding:0;
}
#hornavigation a {
    float:left;
    margin-right:0.5%;
    margin-top:2px;
```

```
        padding-right:0.5%;
        padding-left:0.5%;
        padding-top:0.5em;
        padding-bottom:0.5em;
        border:1px solid gray;
        height:1em;
        width:22.75%
    }
```

Abbildung 5.45: Das Zwischenergebnis, hier im Internet Explorer

Vertikale Navigationsleiste

Die vertikale Navigationsleiste müssen Sie auch nicht zwingend positionieren. Es reicht aus, sie im normalen Fluss anzeigen zu lassen und deren Inhalte, wie im Beispiel die Links, als Floating-Elemente links auszurichten. Alternativ können Sie die einzelnen Links auch in Blockelemente, beispielsweise `p`, einfassen. Dann stehen die Links automatisch untereinander. Sie könnten aber natürlich auch die vertikale Navigationsleiste rechtsbündig ausrichten, entweder über absolute bzw. relative Positionierung oder als Floating-Box. Dann müssten Sie die horizontale Navigationsleiste entsprechend links ausrichten.

```
<div id="vertnavigation">
    <a href="index.html">Home</a>
    <a href="produkte.html">Produkte</a>
    <a href="service.html">Service</a>
    <a href="impressum.html">Impressum</a>
</div>
```

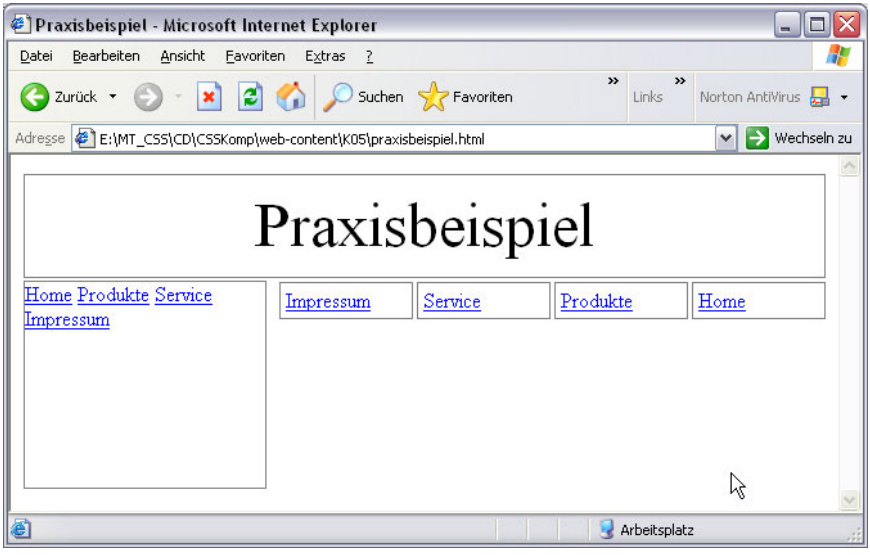
Listing 5.27: Der HTML-Code zur Definition der vertikalen Navigationsleiste

Um die Navigationsleiste korrekt auszurichten, sollten Sie den gleichen oberen Abstand wie für die horizontale Navigationsleiste wählen und eine Breite festlegen, die den noch verfügbaren Platz neben der horizontalen Navigationsleiste ausfüllt. Sinnvoll ist es, die gewünschte Breite mit `width` in Prozent und eine minimale Breite in Pixel zu definieren.

Die Höhe müssen Sie nur für die auf Gecko basierenden Browser definieren. Sie zeigen sonst das umgebende `div`-Element nicht in der für den Inhalt benötigten Höhe an.

```
#vertnavigation {
    border:1px solid gray;
    margin-top:2px;
    width:30%;
    min-width:150px;
    height:10em;
}
```

Abbildung 5.46: Das Zwischenergebnis



Nun müssen Sie noch dafür sorgen, dass die Links in einem angemessenen Abstand untereinander stehen. Dazu erstellen Sie einen zweiten Stil für die Links der Navigationsleiste: Die Angaben `float:left` und `clear:left` sorgen dafür, dass die Links untereinander und nicht nebeneinander stehen. Die weiteren Formatierungen definieren die gewünschte Breite und die Abstände.

Listing 5.28: Ausrichten der Links in der vertikalen Navigationsleiste

```
#vertnavigation a {
    float:left;
    clear:left;
    padding:0.25em;
    border:1px solid silver;
    margin:0.25em;
    width:85%;
}
```

Alternativ können Sie ein ähnliches Ergebnis auch erreichen, indem Sie die Links mit `display:block` als Blockelemente formatieren. Dann füllen sie immer die volle Breite aus, ohne dass Sie die explizit angeben müssen.

```
#vertnavigation a{
    display:block;
    padding:0.25em;
    border:1px solid silver;
    margin:0.25em;
    width:auto;
}
```

Listing 5.29: Die Alternative zum Formatieren der linken Navigationsleiste

5.5 Zweispaltiges Layout für den Inhalt

Zu guter Letzt fehlt noch der Platz für den eigentlichen Seiteninhalt. Dazu gibt es mindestens zwei Möglichkeiten. Die erste besteht darin, dass Sie das umgebende `div`-Element der vertikalen Navigationsleiste mit `float:left` als Floating-Element und dann das `div`-Element, in dem sich der Inhalt befindet, als rechtsbündiges Floating-Element formatieren. Diese Möglichkeit funktioniert sehr gut, wenn Sie wie im Beispiel geschehen, für die obere Navigationsleiste eine Breite vorgegeben haben. Dann können Sie für den Inhalt die gleiche Breite vorgeben, hier 68%.

```
#vertnavigation {
    border:1px solid gray;
    margin-top:2px;
    width:30%;
    min-width:150px;
    height:10em;
    float:left;
}
#inhalt {
    padding:0;
    width: 68%;
    height:auto;
    float:right;
    margin-top:2px;
    margin-left:0
}
```

Listing 5.30: Der notwendige Stil und erforderliche Ergänzungen für die Positionierung des Inhalts

Die Alternative wäre eine absolute bzw. relative Positionierung der Ebene. Das kommt aber bei dem hier vorgestellten Layout nicht in Frage. Durch die variablen Höhen der oberen Elemente, abhängig von der Schriftgröße, können Sie nicht exakt berechnen, wo die Ebene positioniert werden muss.

Nun müssen Sie noch die Spalten innerhalb des Elements mit der ID `inhalt` ausrichten. Dazu definieren Sie zwei Stile, `#spalte1` und `#spalte2`. Sie können beide Spalten entweder wieder als Floating-Boxen formatieren oder absolut innerhalb der Ebene `#inhalt` positionieren. Damit beide Spalten

gleich groß sind und dennoch eine Lücke dazwischen vorhanden ist, sollten Sie deren Breite so festlegen, dass sie zusammen etwas weniger als `100%` ergeben. Die Differenz geben Sie als rechten Innenabstand für die erste Spalte an und definieren bei Bedarf einen rechten Rahmen (`border-right:1px solid black`), wenn Sie eine Linie zwischen den Spalten wünschen. Die Einstellung `text-align-justify` sorgt für Blocksatz innerhalb der Spalten.

Listing 5.31: Positionieren der Spalten als Floating-Elemente

```
#spalte1 {
    float:left;
    top:0;
    left:0;
    width:48%;
    height:100%;
    padding-left:0;
    padding-right:2%;
    border-right:1px solid black;
    text-align:justify;
    margin-top:0;
}
#spalte2 {
    float:right;
    top:0;
    right:0;
    width:49%;
    height:100%;
    text-align:justify;
    margin-top:0;
}
```

Auch hier kommt absolute bzw. relative Positionierung deshalb nicht in Frage, weil nicht alle Teile der Seite eine exakte Größe und Position einnehmen und Sie so die Position der Spalten nicht genau bestimmen können.

Der HTML-Code für die Anzeige der Spalten fällt sehr einfach aus:

Listing 5.32: Der notwendige HTML-Code

```
<div id="inhalt">
    <h1>Seitentitel</h1>
    <div id="spalte1"><p>Das ist der Inhalt der Spalte, das ist der ...
        ist der Inhalt der Spalte,das ist der Inhalt der
        Spalte.</p></div>
    <div id="spalte2"><p>Das ist der Inhalt der Spalte, das ist der ...
        ist der Inhalt der Spalte,das ist der Inhalt der
        Spalte.</p></div>
</div>
```

Abbildung 5.47: Das fertige Layout in Firefox

Netscape-Kompatibilität

Während sich das Ergebnis in allen modernen Browsern sehen lassen kann, verursacht es im Netscape Navigator 4.x Chaos. Um das zu beseitigen, sollten Sie die Stile in eine externe CSS-Datei auslagern und über die `@import`-Anweisung einbinden. Die Stile werden dann vom Netscape Navigator nicht berücksichtigt.

Erstellen Sie zudem eine Kopie der CSS-Datei unter dem Namen *formate_NN.css* und verknüpfen Sie sie vor dem `style`-Element mithilfe des `link`-Elements. Damit haben Sie eine spezielle CSS-Datei für den Netscape Navigator erzeugt, in der Sie nun die Formatierungen löschen können, die für das Chaos bei der Anzeige sorgen. Das sind vor allem die Formatierungen für die Floating-Ebenen, insbesondere die Eigenschaft `float` kombiniert mit `margin` und `padding`.

Sie finden die CSS-Datei für den Netscape Navigator auf der Buch-CD in der Datei BSP/K05/formate_NN.css.

Abbildung 5.48: Ein nicht perfektes, aber akzeptables Ergebnis für den Netscape Navigator 4.x

Listing 5.33: Stylesheet für den Netscape Navigator definieren und allgemeines Stylesheet importieren

```
<link rel="stylesheet" type="text/css" media="screen"
   href="formate_NN.css">
<style type="text/css">
<!--
    @import url("formate.css");
-->
</style>
```

6 Listen und Aufzählungen

Dieses Kapitel behandelt die CSS-Eigenschaften, die Sie zur Formatierung von Listen und Aufzählungen benötigen. Dazu gehören vor allem die Elemente zum Definieren und Ausrichten von Listenzeichen. Aber auch die in CSS 2.0 neu eingeführten Marker werden beschrieben.

6.1 Einführung

In CSS erzeugen fast alle Elemente eine einzelne Box, die so genannte Hauptbox. Es gibt allerdings einige wenige Elemente, dazu gehören Listenelemente, die neben der Hauptbox eine zweite Box erzeugen, in der ein Aufzählungszeichen, eine Grafik etc. angezeigt werden kann.

Was eine Box ist und was es mit dem Boxmodel auf sich hat, können Sie in Kapitel 5, »Größen, Abstände und Positionierung«, nachlesen. Dort ist auch die für Marker relevante Eigenschaft `display` *im Detail erläutert.*

Um einen Marker zu definieren, müssen Sie für ein Element die `display`-Eigenschaft auf `marker` setzen und das Element zur Laufzeit erzeugen, indem Sie beispielsweise das Pseudo-Element `:before` verwenden. Danach können Sie dann über die `content`-Eigenschaft den Inhalt bestimmen, der im Marker angezeigt wird. Dazu stehen folgende Möglichkeiten zur Verfügung:

- Bilder
- Zeichenketten
- Zahlen
- Zähler

Mit Zähler sind dabei automatische Zähler gemeint, die Sie inkrementieren und zurücksetzen können. Das ermöglicht es Ihnen, hierarchische Listen mit eigenen Zahlenformaten zu erzeugen. Diese können Sie über die `content`-Eigenschaft definieren und ansprechen.

Marker erzeugen Sie, indem Sie mithilfe des Pseudo-Elements `:before` eine neue Box vor einem Element erzeugen und diese Box mit `display:marker` formatieren. Über die `marker-offset`-Eigenschaft können Sie dann die Positionierung des Markers bestimmen. Analog funktioniert das natürlich auch mit `:after`, wenn Sie den Marker nach dem Element erzeugen möchten.

6.2 Praxistaugliche Attribute

Nachfolgend finden Sie CSS-Attribute, die Sie problemlos oder mit nur geringen Einschränkungen in der Praxis einsetzen können. Entweder ist die Browserunterstützung sehr gut oder aber ein Nichtausführen der CSS-Anweisung wirkt sich nicht so negativ aus, dass die Seite nicht mehr bedienbar ist.

Listengrafik (list-style-image)

Mit der Eigenschaft `list-style-image` können Sie eine Grafik festlegen, die als Aufzählungszeichen verwendet wird.

CSS-Element: `list-style-image`	*Mögliche Werte: none*, URI, `inherit`
CSS-Version: CSS 1-3, TV, Mobile	*Zulässig für folgende (X)HTML-Elemente:* alle Elemente, die mit `display:list-item` formatiert sind
Medium: Visual	*Vererbt:* Ja

Palm	NN	Mozilla		Internet Explorer							Opera			Safari		iCab	Kq	FF
2.x	4.x	1.x	1.8	4.x	5.0x (Win)	5.0 (Mac)	5.1+ (Mac)	5.5 (Win)	6	7	6	7	8	1.x	2.0	2.9+	3.x	1.0
●		●	●	●	●	● [1]	● [1]	● [1]	● [1]	● [1]	●	●	●	●	●	● [2]	●	●

[1] Grafiken werden nicht optimal zum Text positioniert.
[2] ab Version 3.0

INFO

Zu den Elementen, auf die die Eigenschaft anwendbar ist, gehören natürlich nicht nur Elemente, die Sie selbst mit `display:list-item` *definiert haben, sondern auch das Element* `li`*, für das das interne Stylesheet des Browsers den entsprechenden Wert für* `display` *gesetzt hat.*

Geben Sie mit `list-style-image` *ein Bild an, überschreibt es den Wert, den Sie für* `list-style-type` *angegeben haben. Dieser wird nur dann verwendet, wenn der Browser die Grafik nicht laden kann. Das kann der Fall sein, wenn die Grafik nicht gefunden wurde oder die Datei in einem nicht unterstützten Grafikformat vorliegt. Geben Sie für* `list-style-image` *den Wert* `none` *an, wird ebenfalls der mit* `list-style-type` *festgelegte Wert bzw. der Standardwert für die jeweilige Listenart (*`ol`*=dezimal,* `ul`*=punkt) verwendet.*

Sie finden das Beispiel auf der Buch-CD in der Datei BSP/K06/list_style_image.html.

Beispiel

Listing 6.1: Der Code zum Testen der list-style-image-Eigenschaft

```
...
    <head>
        <title>list-style-image-Eigenschaft</title>
        <style type="text/css">
        <!--
            .none { list-style-image:none }
            .bild { list-style-image: url("plus.gif") }
        -->
        </style>
    </head>

    <body>
        <h1>list-style-image-Eigenschaft</h1>
        <ul class="bild">
            <li>Listeneintrag mit Bild</li>
            <li>Listeneintrag mit Bild</li>
            <li class="none">Listeneintrag ohne Bild</li>
            <li>Listeneintrag mit Bild</li>
        </ul>
    </body>
...
```

Alle Internet-Explorer-Versionen bis einschließlich Version 7 stellen die Grafik im Verhältnis zum Text zu hoch dar.

Die übrigen Browser positionieren die Grafik weitgehend korrekt, wenn die Höhe der Grafik in etwa der Höhe der Schrift entspricht. Legen Sie die Schrift auf 10 Pixel fest, sollten Sie auch die Grafik 10 Pixel hoch erstellen und das Aufzählungszeichen dort mittig positionieren.

Listenstil (list-style)

Die Eigenschaft `list-style` stellt eine Kurzform der Eigenschaften `list-style-type`, `list-style-image` und `list-style-position` dar. Die Syntax lautet:

```
list-style: list-style-type-Wert list-style-position-Wert list-style-image-Wert
```

Mit `list-style: square inside url("plus.gif")` können Sie also eine Listenformatierung festlegen, die äquivalent ist zu:

```
list-style-type:square;
list-style-position:inside;
list-style-image:url("plus.gif")
```

CSS-Element: `list-style`	*Mögliche Werte:* siehe Einzelwerte
CSS-Version: CSS 1-3, TV, Mobile	*Zulässig für folgende (X)HTML-Elemente:* alle mit `display:list-item` formatierten Elemente
Medium: Visual	*Vererbt:* Ja

Palm	NN	Mozilla		Internet Explorer							Opera			Safari		iCab	Kq	FF
2.x	4.x	1.x	1.8	4.x	5.0x (Win)	5.0 (Mac)	5.1+ (Mac)	5.5 (Win)	6	7	6	7	8	1.x	2.0	2.9+	3.x	1.0
●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●

!! STOP

Für die Browser gelten hinsichtlich der unterstützten Werte die Einschränkungen und Informationen zu den einzelnen Eigenschaften `list-style-type`, `list-style-image` ***und*** `list-style-position`.

CD

Sie finden das Beispiel auf der Buch-CD in der Datei BSP/K06/list_style.html.

Listentyp (list-style-type)

Mit der Eigenschaft `list-style-type` können Sie den Typ einer Liste definieren. Außerdem ist es möglich, zusammen mit den anderen Eigenschaften (`list-style-image`, `list-style-position` und `list-style`) das Aussehen der Markerbox von Listeneinträgen zu bestimmen. Anders als bei mit CSS 2.0 eingefügten allgemeinen Markern (*siehe Kapitel 5, Abschnitt »Anzeigeart (display)«*), können Sie jedoch keine Schriften, Farben und Formatierungen festlegen, die von denen der Hauptbox abweichen.

Geben Sie zusätzlich ein Bild als Listenzeichen an (`list-style-image`*), wird das* `list-style-type`*-Listenzeichen nur dann angezeigt, wenn der Wert der* `list-style-image`*-Eigenschaft* `none` *ist oder das Bild nicht verfügbar ist.*

CSS-Element: `list-style-type`	*Mögliche Werte: disc*, `circle`, `square`, `decimal`, `decimal-leading-zero`[1+2+3], `lower-roman`[3], `upper-roman`[3], `lower-greek`[1+2+3], `lower-alpha`, `lower-latin`[1+3], `upper-alpha`, `upper-latin`[2+3], `armenian`[1+2+3], `georgian`[1+2+3], `hebrew`[1+2+3], `cjk-ideographic`[1+2+3], `hiragana`[1+2+3], `katakana`[1+2+3], `hiragana-iroha`[1+2+3], `katakana-iroha`[1+2+3], `none`, `inherit`
CSS-Version: CSS 1-3, TV, Mobile	*Zulässig für folgende (X)HTML-Elemente:* alle mit `display:list-item` formatierten Elemente
Medium: Visual	*Vererbt:* Ja

Palm	NN	Mozilla		Internet Explorer							Opera			Safari		iCab	Kq	FF
2.x	4.x	1.x	1.8	4.x	5.0x (Win)	5.0 (Mac)	5.1+ (Mac)	5.5 (Win)	6	7	6	7	8	1.x	2.0	2.9+	3.x	1.0
● 5+10	● 4	● 5	● 5+ 6+7	● 5	● 5	● 5	● 5	● 5	● 5	● 5	● 5	● 5+6	● 5+6	● 5+8+ 9	● 5+6+8 +9	● 5+11	● 5+6+ 8+9	● 5+6+ 7

1 nicht in TV-Spezifikation
2 nicht in Mobile-Spezifikation
3 nicht in CSS 1.0
4 `decimal` und Formatierungen für Nummerierungen nur in Elementen innerhalb eines `ol`-Elements.
5 die Werte: `disc`, `circle`, `square`, `decimal`, `lower-alpha`, `upper-alpha`, `lower-roman`, `upper-roman`, `none`, `inherit`
6 zusätzlich die Werte: `decimal-leading-zero`, `lower-greek`, `lower-latin`, `upper-latin`, `armenian`
7 zusätzlich die Werte: `hebrew`, `georgian`, `lower-latin`, `upper-latin`, `armenian`, `katakana`, `katakana-iroha`
8 zusätzlich `hebrew`, `georgian`
9 ohne `decimal-leading-zero` und `armenian`
10 zusätzlich `decimal-leading-zero`.
11 ab der Version 3.0 alle Werte außer `hiragana-iroha` und `hiragana`

Zu den Elementen, auf die die Eigenschaft anwendbar ist, gehören natürlich nicht nur Elemente, die Sie selbst mit `display:list-item` *definiert haben, sondern auch das Element* `li`*, für welches das interne Stylesheet des Browsers den entsprechenden Wert für* `display` *gesetzt hat.*

Nicht alle Werte sind in allen CSS-Versionen verfügbar. Mit CSS 2.0 wurden einige neue Werte eingeführt, die eine schlechtere Browserunterstützung aufweisen als die Werte aus der Version 1.0.

Mögliche Werte gemäß CSS 1.0

- `none`: Es werden weder Listenzeichen noch Nummerierungen verwendet.
- `circle`: Kreise werden als Aufzählungszeichen verwendet.
- `disc`: Der Browser hat die Wahl zwischen gefüllten Kreisen und Quadraten als Aufzählungszeichen.
- `square`: Quadrate werden als Aufzählungszeichen verwendet.
- `decimal`: Einfache Dezimalzahlen werden als Nummerierung verwendet. Der erste Listeneintrag bekommt die Nummer 1.
- `lower-alpha`: Nummerierung mit kleinen Buchstaben beispielsweise, a, b, ... z
- `upper-alpha`: Nummerierung mit großen Buchstaben beispielsweise, A, B, ... Z

Sie finden das Beispiel auf der Buch-CD in der Datei BSP/K06/list_style_type.html.

Browserunterstützung

Die Browserunterstützung für die Werte aus CSS 1.0 ist durchgehend sehr gut. Diese Werte sind daher durchaus praxistauglich.

iCab 2.9.x zeigt Listenelemente mit dem Wert `decimal` unterschiedlich an, abhängig davon, ob sie sich in einem `ol` oder `ul`-Element befinden. Bei einem `ol`-Element werden sie mit etwas Abstand zum Text des Elements angezeigt, so dass es wirklich wie eine Nummerierung wirkt. Wenden Sie den gleichen Wert auf ein Listenelement innerhalb eines `ul`-Elements an, wird keine Lücke zwischen Nummer und Text angezeigt, so dass es wirkt, als wenn die Nummer zum Text des Listeneintrags gehört. In der Version 3.0 ist das Problem behoben.

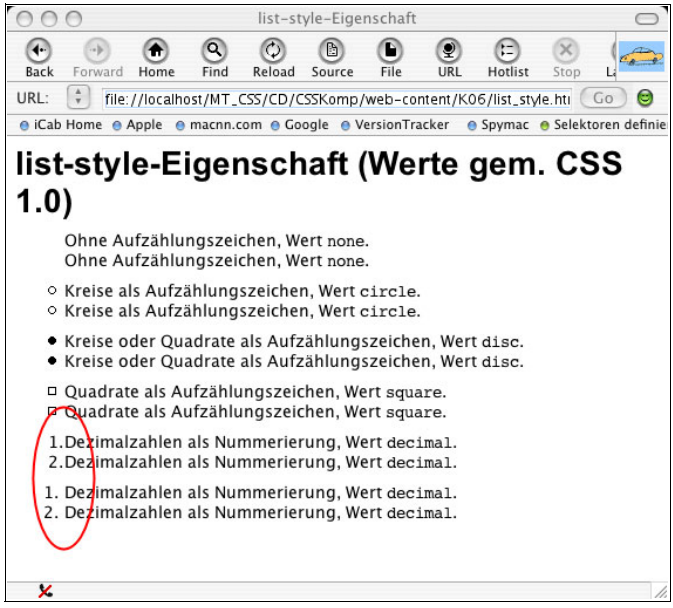


Abbildung 6.1: Fehlerhafte Darstellung des Wertes decimal

Mögliche Werte gemäß CSS 2.0

- `lower-latin`: Nummerierung mit kleinen lateinischen Buchstaben beispielsweise, a, b, ... z
- `upper-latin`: Nummerierung mit großen lateinischen Buchstaben beispielsweise, A, B, ... Z
- `decimal-leading-zero`: Es werden Dezimalzahlen als Aufzählung verwendet, denen einen 0 vorangestellt wird, falls sie einstellig sind. Die Liste beginnt also mit 01, 02, 03 ..., so dass alle Zahlen bis einschließlich 99 zweistellig angezeigt werden.
- `lower-roman`: kleine römische Zahlen beispielsweise i, ii, iii, iv, v ...
- `upper-roman`: große römische Zahlen wie I, II, III, IV, V ...
- `lower-greek`: kleine griechische Zeichen beispielsweise α, β, χ, ...
- `hebrew`: Nummerierung mit hebräischen Nummern
- `georgian`: Nummerierung mit georgischen Nummern
- `armenian`: armenische Nummern
- `cjk-ideoraphic`: ideographische Nummern
- `hiragana`: japanische hiragana-Zeichen
- `katakana`: japanische katakana-Zeichen
- `hiragana-iroha`: japanische hiragana-iroha-Zeichen
- `katakana-iroha`: japanische katakana-iroha-Zeichen

Sie finden das Beispiel auf der Buch-CD in der Datei BSP/K06/list_style_type2.html.

Generell gilt für alle alphabetischen Nummerierungen, dass der CSS-Standard nicht vorgibt, wie mit Nummern zu verfahren ist, die über das Alphabet hinausgehen. Auch die Berechnung römischer Zahlen und Nummern ist nicht definiert. Sie müssen daher mit unterschiedlichen Darstellungen in den einzelnen Browsern rechnen und sollten besser auf numerische Listen zurückgreifen, wenn Sie sehr lange Listen erstellen möchten.

Unterstützt ein Browser bestimmte Nummerierungen nicht, soll er den Wert `decimal` *verwenden.*

Browserunterstützung

Für die von CSS 2.0 definierten Werte fällt die Browserunterstützung sehr unterschiedlich aus. Zumindest verwenden alle Browser standardmäßig den Wert `decimal`, wenn die gewünschte Nummerierungsart nicht unterstützt wird. Römische Ziffern werden jedoch von allen Browsern unterstützt und können daher problemlos eingesetzt werden.

Listenzeichenposition (list-style-position)

Mit der Eigenschaft `list-style-position` können Sie angeben, wo das Listenzeichen angezeigt werden soll. Für das Listenzeichen wird eine eigene Inline-Box erzeugt, die als Markerbox bezeichnet wird. Diese Box kann innerhalb (`inside`) oder außerhalb (`outside`) der Hauptblockbox des Listenelements stehen. Das zeigt sich vor allem bei mehrzeiligem Text im Listenelement. In diesem Fall wird das mit `list-style-position:inside` formatierte Aufzählungszeichen vom Text umflossen. Zeichnen Sie mit `border` einen Rahmen um das Element, steht das Listenzeichen beim Wert `inside` innerhalb, bei `outside` außerhalb des Rahmens.

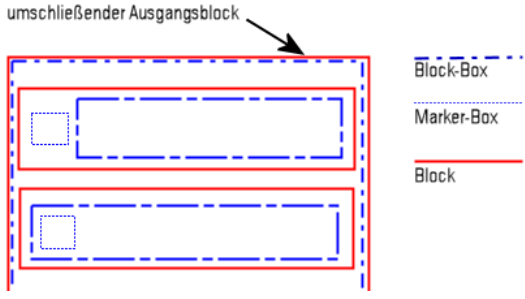


Abbildung 6.2: Darstellung der Werte outside und inside

CSS-Element: `list-style-position`	*Mögliche Werte:* `inside`, *outside*, `inherit`
CSS-Version: CSS 1-3, Mobile, TV	*Zulässig für folgende (X)HTML-Elemente:* alle Elemente, die mit `display:list-item` formatiert sind
Medium: Visual	*Vererbt:* Ja

Palm	NN	Mozilla		Internet Explorer							Opera			Safari		iCab	Kq	FF
2.x	4.x	1.x	1.8	4.x	5.0x (Win)	5.0 (Mac)	5.1+ (Mac)	5.5 (Win)	6	7	6	7	8	1.x	2.0	2.9+	3.x	1.0
○ 2	○ 1	○ 2	○ 2	○ 2	○ 2	○ 2	○ 2	○ 2	○ 2	○ 2	○ 2	○ 2	○ 2	○ 2	○ 2	○ 2	○ 2	○ 2

1 nur `outside`
2 Anwendung nur bei den Elementen `li`, `ul` und `ol` (siehe dazu das Beispiel zur `marker-offset`-Eigenschaft)

Zu den Elementen, auf die die Eigenschaft anwendbar ist, gehören natürlich nicht nur Elemente, die Sie selbst mit `display:list-item` *definiert haben, sondern auch die Elemente* `li` *sowie* `ul` *und* `ol`*, für welche das interne Stylesheet des Browsers den entsprechenden Wert für* `display` *gesetzt hat.*

INFO

Sie finden das Beispiel auf der Buch-CD in der Datei BSP/K06/list_style_position.html.

CD

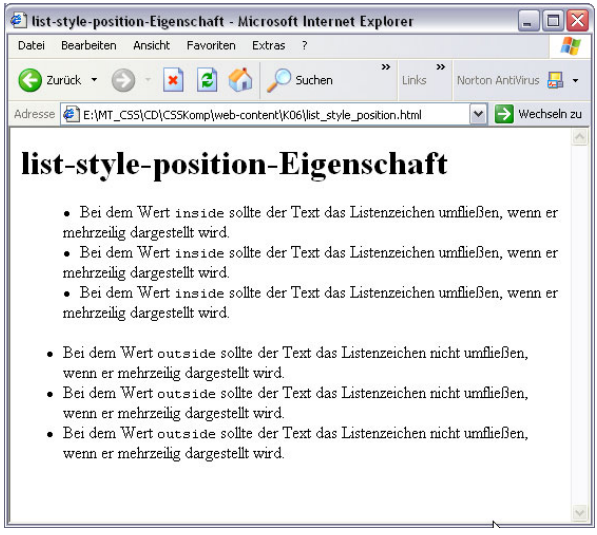


Abbildung 6.3: Korrekte Darstellung beider Werte im Internet Explorer 6

6.3 Nicht/schlecht unterstützte Attribute

In dieser Rubrik finden Sie solche Attribute, die von vielen oder zumindest von sehr wichtigen Browser nicht oder fehlerhaft unterstützt werden und die somit noch nicht praxistauglich sind.

Inhaltserzeugung (content)

Mit der `content`-Eigenschaft können Sie Inhalte generieren, das heißt, Sie können vor oder nach einem Element Texte ausgeben, Zähler erzeugen oder Anführungszeichen definieren.

CSS-Element: `content`	*Mögliche Werte:* Zeichenkette, URI[1], Zählername, Attributwert, `open-quote`, `close-quote`, `no-open-quote`, `no-close-quote`, `inherit`, ***normal***[6], `none`[6]
CSS-Version: CSS 2, CSS 2.1 CSS 3	*Zulässig für folgende (X)HTML-Elemente:* Elemente mit `display: marker` und die Pseudo-Elemente `:before` und `:after`
Medium: Visual	*Vererbt:* Nein

Palm	NN	Mozilla		Internet Explorer							Opera			Safari		iCab	Kq	FF
2.x	4.x	1.x	1.8	4.x	5.0x (Win)	5.0 (Mac)	5.1+ (Mac)	5.5 (Win)	6	7	6	7	8	1.x	2.0	2.9+	3.x	1.0
		○ 3+4	○ 3								⊙ 2	⊙ 2	⊙ 2	○ 4	○ 4	● 5	○ 2	○ 3+4

1 Der Wert steht in CSS 2.1 nicht zur Verfügung.
2 `content`-Eigenschaft wird für beliebige Selektoren ausgeführt und ersetzt in diesem Fall den Inhalt des Elements, falls ein einfacher Text bzw. Zähler angegeben wird.
3 ohne die Werte für die Erzeugung bzw. Inkrementierung von Anführungszeichen
4 ohne Zähler
5 ab Version 3.0
6 erst ab CSS 2.1

Sie finden das Beispiel auf der Buch-CD in der Datei BSP/K06/content.html.

Mögliche Werte

- Zeichenkette: Die angegebene Zeichenkette wird eingefügt. Beispiel: `.listenzeichen:before {content:"+ "}`.
- URI: Gibt eine externe Datei an, aus der der Inhalt erzeugt wird. Beispiel: `.uri:before{content:url("plus.gif")}`.
- Zählername: Definiert einen Zähler, der beispielsweise in einem Marker oder vor einem Element angezeigt werden kann. Mit dem folgenden Code wird vor jedem Absatz mit der Klasse `.zaehler` eine fortlaufende Nummer angezeigt: `.zaehler:before{`**`content:counter(Zaehler1) ». «`**`;counter-increment:Zaehler1}`.
  Alternativ können Sie auch einen Stil für den Zähler angeben. Dazu stehen alle Werte für die `list-style-type`-Eigenschaft zur Verfügung, die eine numerische oder alphanumerische Liste definieren. Den Stil geben Sie als zweiten Parameter hinter dem Zählernamen an. Beispiel: `content:counter(Zaehler2,upper-alpha)`.
- `none`: Es wird kein Pseudo-Element erzeugt.
- `normal`: Für `:before` und `:after` wird kein Inhalt erzeugt.

Weitere Informationen zu Zählern finden Sie im Abschnitt »Zähler erhöhen (counter-increment)«.

REF

- Mit den Werten `open-quote` und `close-quote` können Sie festlegen, dass öffnende oder schließende Anführungszeichen eingefügt werden sollen. Welche Zeichen dafür zu verwenden sind, können Sie mit der `quote`-Eigenschaft bestimmen.
  Mit folgendem Code können Sie bewirken, dass vor dem Element mit der CSS-Klasse `zitat` ein rotes öffnendes Anführungszeichen eingefügt wird und danach ein rotes schließendes:

```
.zitat        { quotes:"«" "»" }
.zitat:before { content:open-quote; color:red }
.zitat:after  { content:close-quote; color:red }
```

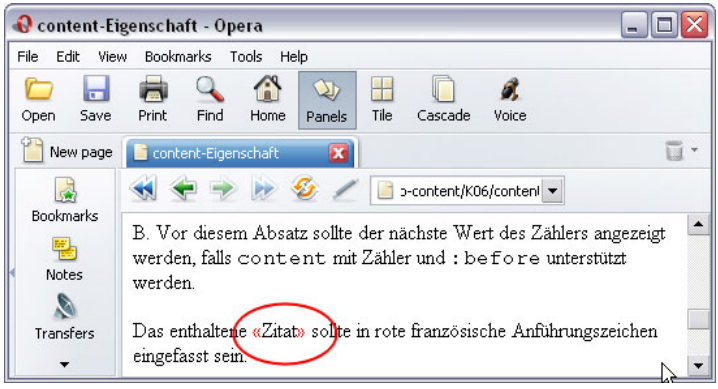


Abbildung 6.4: Das Ergebnis im Opera-Browser

Informationen zur `quotes`*-Eigenschaft finden Sie in Kapitel 4, »Textformatierungen«.*

- Der Wert `no-open-quote` erhöht die Verschachtelungstiefe für die Anführungszeichen, ohne jedoch ein Anführungszeichen einzufügen.
- Mit dem Wert `no-close-quote` wird kein Anführungszeichen eingefügt, aber dennoch die Verschachtelungstiefe der Anführungszeichen verringert.
- Der Wert `attr(x)` definiert, dass der Wert des Attributs `x` als Inhalt angezeigt wird. Mit `content:attr(lang) ": "` würden Sie den Wert des `lang`-Attributs gefolgt von einem Doppelpunkt und einem Leerzeichen als Inhalt einfügen. Falls das Attribut nicht vorhanden ist, wird eine leere Zeichenkette anstelle des Attributwerts verwendet.

Sehr gut lässt sich dieser Wert in Kombination mit Attribut-Selektoren einsetzen. Wenn Sie vermeiden möchten, dass, wie im obigen Beispiel, bei Fehlen des Attributs dennoch ein Doppelpunkt und ein Leerzeichen eingefügt werden, können Sie folgenden Selektor verwenden:

```
p[lang]:before {
    content:attr(lang) ": ";
    font-weight:bold;
    color:red;
    text-transform: uppercase
}
```

In diesem Fall wird der Inhalt nur dann erzeugt, wenn das Attribut vorhanden ist. Der Attributwert wird zudem fett und rot formatiert und in Großbuchstaben umgewandelt. Ist folgender HTML-Code vorhanden, ergibt sich dann die Anzeige wie in Abbildung 6.5:

```
<p lang="de">
    Dieser Absatz enthält deutschen Inhalt
</p>
<p lang="en">
    This paragraph contains english content
</p>
```

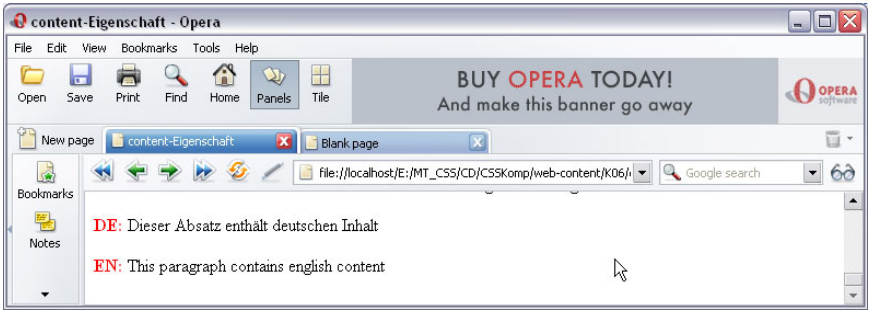


Abbildung 6.5: Der angewendete Attribut-Selektor mit Ausgabe des lang-Attributs

Die Opera-Browser 7 und 8 für Windows zeigen Darstellungsfehler an, wenn die Werte `open-quote` und `close-quote` für die `content`-Eigenschaft verwendet werden und diese Formatierungen Elemente betreffen, die ineinander verschachtelt werden. Zum einen wird vor dem inneren `span`-Element das falsche Anführungszeichen angezeigt, nämlich ein schließendes statt ein öffnendes. Zum anderen wird nicht das Anführungszeichen aus der `quotes`-Eigenschaft als schließendes Anführungszeichen nach dem eingeschobenen Element verwendet, sondern einfach gerade Anführungszeichen.

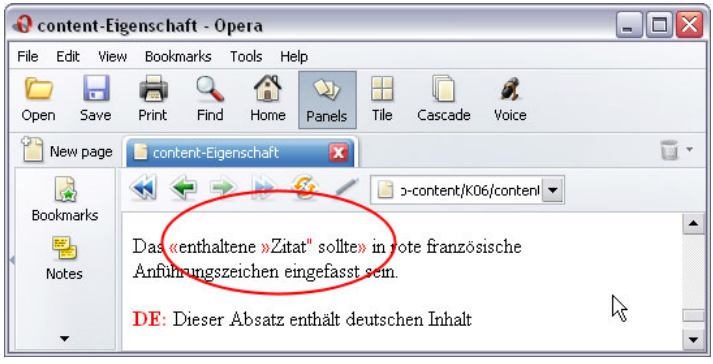


Abbildung 6.6: Darstellungsfehler

Browserkompatibilität

Die Browserunterstützung für ausgegebene Texte und Bilder ist recht gut. Allerdings beherrscht der Internet Explorer, der immer noch Marktführer ist, die `content`-Eigenschaft nicht, weshalb es noch nicht sinnvoll ist, die Eigenschaft einzusetzen.

Marker-Position (marker-offset)

Mit der Eigenschaft `marker-offset` können Sie festlegen, wie groß der Abstand der Markerbox zur Hauptbox ist. Als Werte kommen numerische Werte (auch negative) mit Längeneinheit in Frage.

CSS-Element: `marker-offset`	*Mögliche Werte: auto*, `inherit`, numerischer Wert mit Einheit
CSS-Version: CSS 2, CSS 3	*Zulässig für folgende (X)HTML-Elemente:* alle Elemente, die mit `display:marker` formatiert sind
Medium: Visual	*Vererbt:* Nein

Palm	NN	Mozilla		Internet Explorer							Opera			Safari		iCab	Kq	FF
2.x	4.x	1.x	1.8	4.x	5.0x (Win)	5.0 (Mac)	5.1+ (Mac)	5.5 (Win)	6	7	6	7	8	1.x	2.0	2.9+	3.x	1.0

[1] Da kein Browser `display:marker` unterstützt, lässt sich nicht feststellen, ob `marker-offset` ausgeführt wird.

Sie finden das Beispiel auf der Buch-CD in der Datei BSP/K06/marker_offset.html.

Geben Sie als Wert einen numerischen Wert an, legt dieser immer den Abstand zwischen den beiden Kanten von Markerbox und Hauptblockbox fest, die sich am nächsten liegen. Bei einem Marker, der vor einem Element angezeigt wird, bezieht sich der Wert also auf den Abstand zwischen rechter Markerkante und linker Hauptblockboxkante.

Mit

```
.liste:before {
    content:"+"; display:marker;
    border:1px solid silver;
    padding:2px;
    height:12px;
    width:12px;
    color:silver;
    font-size:8px;
    text-align:center;
    marker-offset:20px;
}
```

würde ein Marker mit Plus-Symbol erzeugt und mit einem grauen Rahmen versehen. Er sollte einen Abstand von `20px` zum zugehörigen Listeneintrag haben.

Abbildung 6.7: So sollte die marker-offset-Eigenschaft angezeigt werden: mit 20 Pixel Abstand zwischen Marker und Hauptblockbox.

Kein Browser unterstützt bisher die `marker-offset`*-Eigenschaft. Der Abstand in der Abbildung wurde hier mit* `margin-right` *erzeugt. Das ist zumindest eine Möglichkeit, den Abstand zur Hauptblockbox zu kontrollieren.*

!! STOP

Dieses Beispiel zeigt außerdem, dass kein Browser bei Angabe von `display:list-item` die Eigenschaft `list-style-position:outline` korrekt anwendet. Dann dürfte der Text den Marker nämlich nicht umfließen.

Zähler erhöhen (counter-increment)

Die Eigenschaft `counter-increment` erhöht einen definierten Zähler um 1 oder um einen angegebenen Wert. Das geschieht immer dann, wenn der Stil angewendet wird, der die Eigenschaft enthält.

CSS-Element: `counter-increment`	*Mögliche Werte: none*, `inherit`, Zählerinkrement
CSS-Version: CSS 2, CSS 3	*Zulässig für folgende (X)HTML-Elemente:* alle
Medium: All	*Vererbt:* Nein

Palm	NN	Mozilla		Internet Explorer							Opera			Safari		iCab	Kq	FF
2.x	4.x	1.x	1.8	4.x	5.0x (Win)	5.0 (Mac)	5.1+ (Mac)	5.5 (Win)	6	7	6	7	8	1.x	2.0	2.9+	3.x	1.0
			○ [2]								●	●	●			● [1]		

[1] ab Version 3.0
[2] Unterstützung nur für bestimmte Elemente, wie `table`

Sie finden das Beispiel auf der Buch-CD in der Datei BSP/K06/counter.html.

Der Wert Zählerinkrement besteht aus einer Kombination von Zählername und Wert. Mit der Angabe `counter-increment: zaehler 3` würde der Zähler `zaehler` um den Wert 3 erhöht werden. Sie können den Wert auch weglassen, in diesem Fall wird der Zähler um 1 erhöht. Geben Sie einen negativen Wert an, wird der Zähler um den entsprechenden Wert reduziert.

Folgendes Beispiel definiert zwei Zähler: `Kap` und `UKap`. Der Zähler `Kap` wird vor `h2`-Elementen ausgegeben und dann erhöht. Der Zähler `UKap` wird mit vorangestelltem Zähler `Kap` vor den `h3`-Elementen ausgegeben und dann ebenfalls um 1 erhöht.

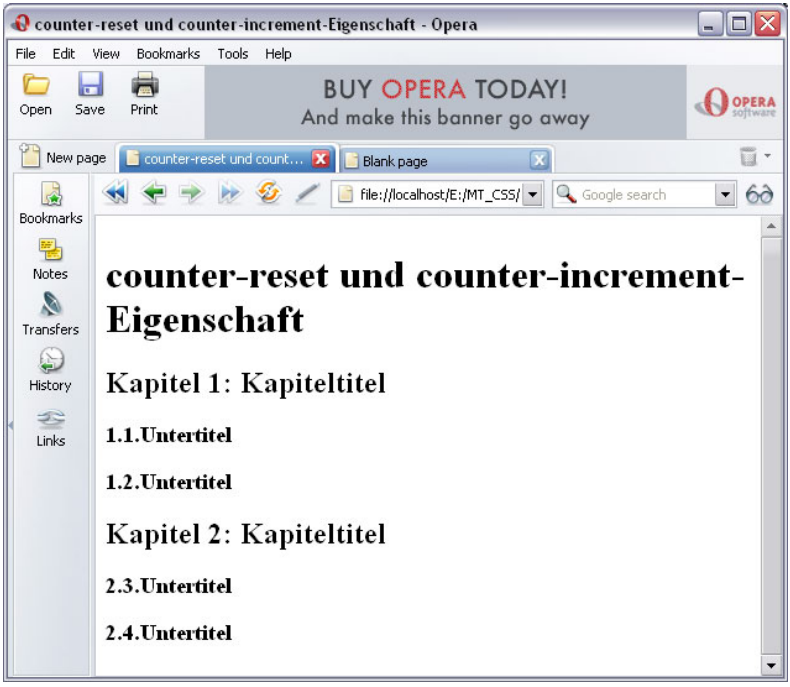


Abbildung 6.8: Über die Zähler und die Eigenschaft counter-increment erzeugte Nummerierung

```
<html>
    <head>
        <title>counter-reset und counter-increment-Eigenschaft</title>
        <style type="text/css">
        <!--
            h2:before { content:"Kapitel " counter(Kap) ": ";
                        counter-increment:Kap }
            h3:before { content:counter(Kap) "." counter(UKap) ".";
                        counter-increment:UKap 1 }
        -->
        </style>
    </head>
    <body>
        <h1>counter-reset und counter-increment-Eigenschaft</h1>
        <h2>Kapiteltitel</h2>
        <h3>Untertitel</h3>
        <h3>Untertitel</h3>
        <h2>Kapiteltitel</h2>
        <h3>Untertitel</h3>
        <h3>Untertitel</h3>
    </body>
</html>
```

Listing 6.2: Beispiel zur Eigenschaft counter-increment

Wie Sie an der Abbildung sehen können, wird der Zähler `UKap` *nicht automatisch zu Beginn des zweiten Kapitels auf 1 zurückgesetzt. Um das zu erreichen, benötigen Sie die* `counter-reset`*-Eigenschaft.*

Mehr zur `counter-reset`*-Eigenschaft finden Sie im Abschnitt »Zähler zurücksetzen (counter-reset)«.*

Zähler zurücksetzen (counter-reset)

Die Eigenschaft `counter-reset` setzt einen definierten Zähler zurück. Das geschieht immer dann, wenn der Stil angewendet wird, der die Eigenschaft enthält.

CSS-Element: `counter-reset`	*Mögliche Werte: none*, `inherit`, Zählerinkrement
CSS-Version: CSS 2, CSS 3	*Zulässig für folgende (X)HTML-Elemente:* alle
Medium: All	*Vererbt:* Nein

Palm	NN	Mozilla		Internet Explorer							Opera			Safari		iCab	Kq	FF
2.x	4.x	1.x	1.8	4.x	5.0x (Win)	5.0 (Mac)	5.1+ (Mac)	5.5 (Win)	6	7	6	7	8	1.x	2.0	2.9+	3.x	1.0
			○ [2]								●	●	●			● [1]		

[1] ab Version 3.0
[2] nur für bestimmte Elemente, wie `table`

Sie finden das Beispiel auf der Buch-CD in der Datei BSP/K06/counter.html.

Der Wert Zählerinkrement besteht aus einer Kombination von Zählername und Wert. Mit der Angabe `counter-reset: zaehler 3` würde der Zähler `zaehler` auf den Wert 3 zurückgesetzt werden. Den Wert können Sie auch weglassen, in diesem Fall wird der Zähler auf 0 gesetzt.

Um den Zähler `UKap` aus dem Beispiel zur `counter-increment`-Eigenschaft (siehe *Abschnitt »Zähler erhöhen (counter-increment)«*) so zurückzusetzen, dass er nach einem `h2`-Element wieder bei 1 beginnt, setzen Sie ihn auf 0 zurück, wenn der Stil `h2` angewendet wird.

```
h2:before { content:"Kapitel " counter(Kap) ": ";
            counter-increment:Kap 1;
            counter-reset: UKap 0 }
```

7 Bilder und Hintergründe

Bilder, eingesetzt als Grafiken oder Hintergrundbilder, sind oft der ausschlaggebende Faktor bei einer ansprechenden Gestaltung von Webseiten. Welche Möglichkeiten die einzelnen CSS-Standards bei dieser Gestaltung bieten, zeigt dieses Kapitel.

7.1 Einführung

Der Hintergrund eines Elements ist die Fläche, die es bedeckt. Dazu gehört der Inhaltsbereich der Box sowie der Innenabstand. Der Rahmen wird hingegen über die `border-color`-Eigenschaft formatiert und der Außenabstand ist immer transparent.

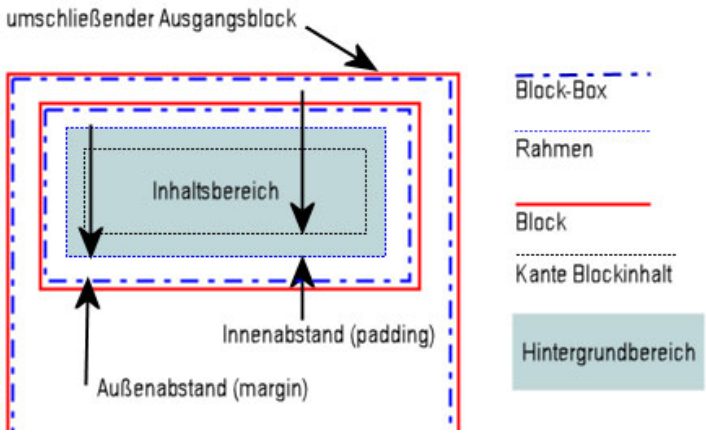


Abbildung 7.1: Definition des Hintergrundbereichs gemäß Boxmodell

Transparent bedeutet, dass der Hintergrund durchscheint. Der Hintergrund kann aus Inhalten bestehen, deren Boxen in der Stapelreihenfolge unter dem Element liegen, oder einfach aus dem Hintergrund des umgebenden Elements. Standardmäßig hat die Eigenschaft `background-color` den Wert transparent und `background-image` den Wert `none`. Dadurch ist ein Element, dessen Hintergrundfarbe und Hintergrundbild Sie nicht festgelegt haben, in jedem Fall durchsichtig.

Aus diesem Grund sollten Sie zumindest für das `body`*-Element die Hintergrundfarbe explizit festlegen, da der CSS-Standard nicht vorschreibt, wie der Viewport dargestellt werden soll, wenn keine Hintergrundfarbe definiert wurde.*

Weitere Informationen zum Boxmodell finden Sie in Kapitel 5, »Größen, Abstände und Positionierung«.

Der Netscape Navigator 4.x und früher stellt Hintergründe fehlerhaft dar. Hinterlegen Sie beispielsweise einen Absatz mit einer Hintergrundfarbe, wird nicht die gesamte Blockbox gefüllt, sondern nur der Teil, in dem sich der Text befindet.

7.2 Praxistaugliche Attribute

Nachfolgend finden Sie CSS-Attribute, die Sie problemlos oder mit nur geringen Einschränkungen in der Praxis einsetzen können. Entweder ist die Browserunterstützung sehr gut oder aber ein Nichtausführen der CSS-Anweisung wirkt sich nicht so negativ aus, dass die Seite nicht mehr bedienbar ist.

Hintergrund (background)

Die `background`-Eigenschaft ermöglicht eine verkürzte Angabe der Eigenschaften `background-color`, `background-image`, `background-repeat`, `background-attachment` und `background-position`. Die Syntax lautet:

```
background: background-color background-image background-repeat
background-attachment background-position
```

Wobei für die einzelnen Eigenschaften die gleichen Einschränkungen und Werte gelten, als wenn Sie die lange Schreibweise verwenden.

CSS-Element: `background`	*Mögliche Werte:* `inherit`, siehe Einzeleigenschaften
CSS-Version: CSS 1-3, TV, Mobile	*Zulässig für folgende (X)HTML-Elemente:* alle
Medium: Visual	*Vererbt:* Nein

Palm	NN	Mozilla		Internet Explorer							Opera			Safari		iCab	Kq	FF
2.x	4.x	1.x	1.8	4.x	5.0x (Win)	5.0 (Mac)	5.1+ (Mac)	5.5 (Win)	6	7	6	7	8	1.x	2.0	2.9+	3.x	1.0
● [1]	● [1]	●	●	● [1]	●	●	●	●	●	●	●	●	●	●	●	● [1]	●	●

[1] Jeweils mit den Einschränkungen/Fehlern, die auch für die einzelnen Werte gelten.

Sie finden das Beispiel auf der Buch-CD in der Datei BSP/K07/background.html.

Mit der Angabe

```
body { background:white url(blaetter.jpg) repeat-y
       scroll center center; color:black; }
```

erzeugen Sie ein Hintergrundbild für die Seite, das horizontal mittig angezeigt wird und vertikal gekachelt wird. Die Kachelung erfolgt ausgehend von der Mitte der Seite.

Hintergrundbild (background-image)

Die Eigenschaft `background-image` definiert ein Hintergrundbild für ein Element. Ist dieses kleiner als der Inhaltsbereich des Elements, wird es gekachelt, bis die Fläche gefüllt ist.

Falls Sie für `background-image` ein Bild angeben und die Hintergrundfarbe über `background-color` spezifizieren, wird das Hintergrundbild vor der Hintergrundfarbe angezeigt. Enthält es transparente Bereiche, scheint dort die Hintergrundfarbe durch. Die Hintergrundfarbe wird zudem auch dann angezeigt, wenn das Hintergrundbild nicht geladen werden kann. Sie sollten daher zusammen mit einem Hintergrundbild auch immer eine Hintergrundfarbe definieren.

CSS-Element: `background-image`	*Mögliche Werte:* URI, *none*, `inherit`
CSS-Version: CSS 1-3, TV, Mobile	*Zulässig für folgende (X)HTML-Elemente:* alle
Medium: Visual	*Vererbt:* Nein

Palm	NN	Mozilla		Internet Explorer							Opera			Safari		iCab	Kq	FF
2.x	4.x	1.x	1.8	4.x	5.0x (Win)	5.0 (Mac)	5.1+ (Mac)	5.5 (Win)	6	7	6	7	8	1.x	2.0	2.9+	3.x	1.0
○ [2]	○ [1]	●	●	●	●	●	●	●	●	●	●	●	●	●	●	○ [3+4]	●	●

1 Pfade werden relativ zum HTML-Dokument interpretiert, transparente GIF-Bilder werden nicht angezeigt.
2 Keine vollständige Füllung der Box
3 Der Außenabstand wird bis einschl. Version 2.9.x ebenfalls gefüllt.
4 Ab Version 3.0 werden Transparenzen in Hintergrundbildern nicht unterstützt.

CD

Sie finden das Beispiel auf der Buch-CD in der Datei BSP/K07/background_image.html.

Die Adresse des zu ladenden Bildes geben Sie als URI mit der `url`-Anweisung an. Die Syntax lautet: `url("Dateiname")`, wobei Sie dem Dateinamen selbstverständlich auch einen Pfad voranstellen können.

INFO

Relative Pfade für die Grafik sollen gemäß CSS-Standard relativ zum Stylesheet interpretiert werden. Befinden sich die Stile also in einer externen CSS-Datei, sollte der Browser eine Grafik, die Sie ohne Pfad angeben, immer im Verzeichnis der CSS-Datei suchen, nicht im Verzeichnis der HTML-Datei.

Der Netscape Navigator 4.x und früher hält sich allerdings nicht daran. Er interpretiert den Pfad relativ zum HTML-Dokument, in dem die CSS-Datei verwendet wird. Als Lösung bietet sich an, dass Sie entweder nur absolute Pfade wie `http://www..../` *verwenden oder darauf achten, dass sich CSS-Datei, Grafik und HTML-Dateien im gleichen Verzeichnis befinden.*

Der Wert `none` bewirkt, dass kein Bild angezeigt wird. In diesem Fall wird die Hintergrundfarbe verwendet oder falls die nicht definiert ist, ein transparenter Hintergrund angezeigt.

Beispiel Mit den beiden Stilen

```
body { background-color:white; color:black }
p    { background-color:red;
       background-image:url("hgrd.gif");
       color:white }
```

erreichen Sie, dass in allen Absätzen eine rote Hintergrundfarbe angezeigt wird und darüber das Hintergrundbild gekachelt wird. Da dieses im Beispiel transparente Streifen enthält, scheint dort die rote Hintergrundfarbe durch.

Abbildung 7.2: Die korrekte Darstellung von Hintergrundbild und Hintergrundfarbe

Der Netscape Navigator 4.x stellt keine Hintergrundbilder dar, bei denen es sich um (teil-)transparente GIF-Bilder handelt. Mit JPG-Grafiken gibt es keine Probleme. Zudem werden relative Pfade relativ zum HTML-Dokument statt relativ zum Stylesheet interpretiert. Auch beim Einsatz der `background-image`-Eigenschaft sollten Sie einen Rahmen definieren, damit die komplette Box gefüllt wird.

Mehr Informationen zu diesem Problem finden im Abschnitt »Hintergrundfarbe (background-color)«.

Der Palm-Browser füllt Blockelemente nicht vollständig. Rechts und unten bleiben je 1 Pixel übrig, die nicht mit dem Hintergrundbild gefüllt werden und bei denen dann die Hintergrundfarbe sichtbar ist. Daher sollten Sie unbedingt darauf achten, eine passende Hintergrundfarbe zu definieren.

iCab 2.9.x kachelt die Grafiken über den Rahmen hinaus in den Außenabstand hinein. Der Außenabstand (`margin`) wird also ebenfalls mit Hintergrundbild unterlegt, zumindest rechts und unten. Um das zu vermeiden, können Sie für `margin` den Wert 0 definieren, falls dies ins Layout passt.

In iCab 3.0 Beta werden transparente Hintergrundbilder nicht transparent angezeigt. Zumindest ist eine ebenfalls definierte Hintergrundfarbe nicht durch die transparenten Pixel sichtbar.

Hintergrund fixieren (background-attachment)

Mit der Eigenschaft `background-attachment` legen Sie fest, ob das Hintergrundbild fixiert werden soll oder mit dem Inhalt des Elements gescrollt wird.

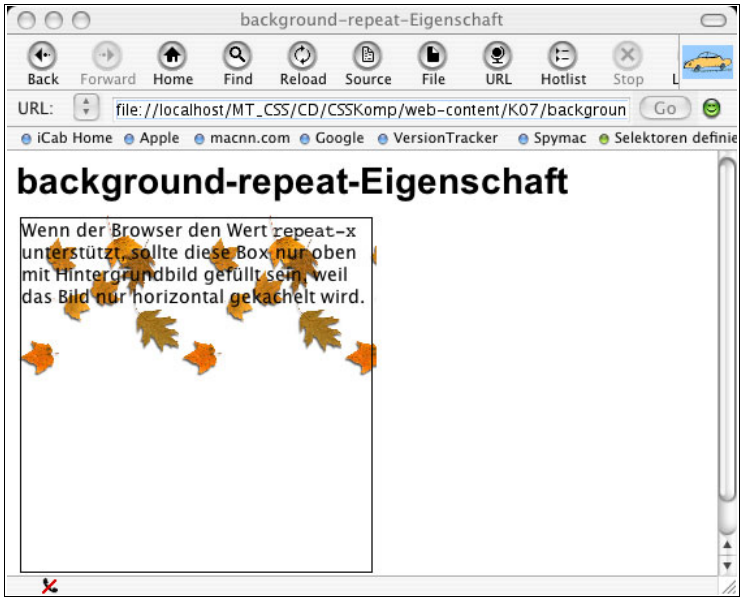


Abbildung 7.3: Fehlerhafte Darstellung des Hintergrundbildes in iCab

CSS-Element: `background-attachment`	*Mögliche Werte: scroll*, `fixed`, `inherit`
CSS-Version: CSS 1-3	*Zulässig für folgende (X)HTML-Elemente:* alle
Medium: Visual	*Vererbt:* Nein

Palm	NN	Mozilla		Internet Explorer							Opera			Safari		iCab	Kq	FF
2.x	4.x	1.x	1.8	4.x	5.0x (Win)	5.0 (Mac)	5.1+ (Mac)	5.5 (Win)	6	7	6	7	8	1.x	2.0	2.9+	3.x	1.0
●		●	●	⊙	●	●	●	●	●	●	●	●	●	●	●	●	●	●

Mögliche Werte

- `scroll`: gibt an, dass das Hintergrundbild mit dem Inhalt des Elements gescrollt wird.
- `fixed`: bestimmt, dass das Bild bezogen auf den Viewport beim Scrollen stehen bleibt. Der Inhalt des Elements gleitet dann darüber hinweg.

CD

Sie finden das Beispiel auf der Buch-CD in der Datei BSP/K07/background_attachment.html.

Hintergrundfarbe (background-color)

Die Eigenschaft `background-color` legt fest, mit welcher Farbe der Hintergrund eines Elements gefüllt werden soll.

Welche Möglichkeiten es gibt, Farben anzugeben, welche Syntax und möglichen Werte also Farbangaben haben, ist in Kapitel 2, »Stile definieren« beschrieben.

CSS-Element: `background-color`	*Mögliche Werte:* Farbwert, *transparent*, `inherit`
CSS-Version: CSS 1 - 3, TV, Mobile	*Zulässig für folgende (X)HTML-Elemente:* alle
Medium: Visual	*Vererbt:* Nein

Palm	NN	Mozilla		Internet Explorer							Opera			Safari		iCab	Kq	FF
2.x	4.x	1.x	1.8	4.x	5.0x (Win)	5.0 (Mac)	5.1+ (Mac)	5.5 (Win)	6	7	6	7	8	1.x	2.0	2.9+	3.x	1.0
●	○ [1]	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●

[1] Nicht der gesamte Inhaltsbereich und Innenabstand der Box werden gefüllt (mit Ausnahme des Hintergrundes des Viewports).

Die Angabe von Hintergrundfarben, ohne eine Vordergrundfarbe festzulegen, kann dazu führen, dass Inhalte nicht mehr lesbar sind. Wenn Sie eine Hintergrundfarbe angeben, sollten Sie daher immer auch eine Vordergrundfarbe definieren.

Sie finden das Beispiel auf der Buch-CD in der Datei BSP/K07/background_color.html.

Netscape füllt nicht die komplette Box eines Blockelements mit der Hintergrundfarbe.

Sie können das Problem mit Netscape umgehen, indem Sie für das Element einen Rahmen erzeugen. Soll der nicht sichtbar sein, geben Sie als Rahmenfarbe einfach transparent an.


```
<p style="border:1px solid transparent">Dieser Text sollte auf
blauem Hintergrund mit weißem Text erscheinen. Auch NN 4+ sollte
den Hintergrund korrekt darstellen, da der Absatz mit einem
```

Abbildung 7.4: Fehlerhafte Darstellung der Hintergrundfarbe im Netscape Navigator 4.x

```
transparenten Rahmen versehen wurde.</p>
```

Mit dem Farbwert `transparent` in der Rahmendefinition hat aber wiederum iCab 2.9.x Probleme und stellt den Rahmen weiß dar. Soll Ihre Seite auch mit iCab korrekt angezeigt werden, sollten Sie als Rahmenfarbe daher die gleiche Farbe verwenden wie beim Hintergrund. Dann stellt der Netscape Navigator die Hintergrundfarbe und iCab 2.9.x den Rahmen korrekt dar.

Beispiel

Folgende Stile sorgen dafür, dass die Seite einen schwarzen Hintergrund erhält und Absätze mit einem blauem Hintergrund versehen werden.

```
body { background-color:#000; color:white }
p    { background-color:rgb(0%,0%,100%) }
```

Hintergrundposition (background-position)

Die Eigenschaft `background-position` legt die Position des Hintergrundbildes fest. Ausgehend von dieser Position erfolgt die mit `background-repeat` definierte Kachelung. Erfolgt eine Kachelung des Bildes, ist die angegebene Position der Ausgangspunkt, von dem aus vertikal bzw. horizontal oder in alle Richtungen gekachelt wird.

CSS-Element: `background-position`	*Mögliche Werte:* `inherit`, Positionswerte in Prozent, Positionswerte als numerische Werte mit Einheit, eines der Schlüsselwörter `top`, `center`, `bottom`, kombiniert mit einem der Schlüsselwörter `left`, `center`, `right`
CSS-Version: CSS 1-3, TV, Mobile	*Zulässig für folgende (X)HTML-Elemente:* alle Blockelemente[1]
Medium: Visual	*Vererbt:* Nein

Palm	NN	Mozilla		Internet Explorer							Opera			Safari		iCab	Kq	FF
2.x	4.x	1.x	1.8	4.x	5.0x (Win)	5.0 (Mac)	5.1+ (Mac)	5.5 (Win)	6	7	6	7	8	1.x	2.0	2.9+	3.x	1.0
●		●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●

[1] Ab CSS 3 ist die Anwendung nicht mehr auf Blockelemente beschränkt.

Prozentuale Werte beziehen sich auf die Größe der Blockbox. Ab CSS 2.1 ist eine kombinierte Angabe von prozentualen Werten und Schlüsselwörtern möglich. Die Angabe `background-position: 50% center` *wäre damit also gültig.*

INFO

Mögliche Werte

Generell gilt, dass Positionswerte paarweise definiert werden sollten. Der erste Wert bestimmt die horizontale Position, der zweite die vertikale. Geben Sie also `background-position: right top` an, wird das Bild rechts oben positioniert.

Optimal ist es, wenn Sie beide Werte nach einer einheitlichen Methode angeben. Also entweder beide in Form von Schlüsselwörtern, beide als numerische Werte oder beide als prozentuale Werte.

Ab CSS 2.1 ist es erlaubt, dass Sie Schlüsselwörter und prozentuale Werte mischen. Explizit verboten ist jedoch die Kombination von numerischen Werten und Schlüsselwörtern.

!! STOP

- prozentuale Werte: Gemäß CSS-Standard geben prozentuale Werte den Abstand des Bildes von der Rahmenkante des Elements an. Ein Wertepaar von 0% 0% positioniert die linke obere Ecke des Bildes an der linken oberen Ecke der Rahmenkante, das Wertepaar 100% 100% positioniert die rechte untere Ecke des Bildes an der rechten unteren Ecke der Rahmenkante. Der Wert 50% 50% bewirkt, dass das Bild sowohl vertikal, wie horizontal zentriert dargestellt wird.
- numerische Werte mit Einheit: Bei numerischen Werten mit Einheit geben die Werte den Abstand der oberen linken Ecke des Bildes von der oberen linken Ecke der Rahmenkante an. Die Werte dürfen auch negativ sein. Dann wird nur der Teil des Bildes angezeigt, der sich innerhalb der Box befindet.

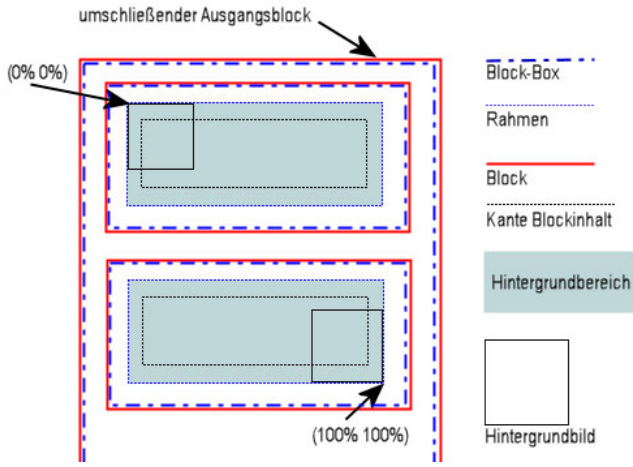


Abbildung 7.5: Positionierung mittels Prozentwerten

- Schlüsselwörter: Die Schlüsselwörter entsprechen den Prozentwerten. Das heißt sie haben folgende Bedeutung:
  - `left` = 0% (horizontal)
  - `right` = 100% (horizontal)
  - `center` = 50% (vertikal oder horizontal)
  - `top` = 0% (vertikal)
  - `bottom` = 100% (vertikal)

Beispiel

Das Beispiel zeigt, wie Sie mit allen verfügbaren Angaben eine zentrierte Ausrichtung des Hintergrundbildes erreichen können. Bei den absoluten Werten müssen Sie dazu die Position der oberen linken Ecke für das Bild selbst ausrechnen und zwar nach folgender Formel:

$$\text{Horizontaler Wert} = \frac{\text{Breite des Elements} + \text{Innenabstand links} + \text{Innenabstand rechts}}{2} - \frac{\text{Breite des Bildes}}{2}$$

$$\text{Vertikaler Wert} = \frac{\text{Höhe des Elements} + \text{Innenabstand oben} + \text{Innenabstand unten}}{2} - \frac{\text{Höhe des Bildes}}{2}$$

Beispiel-Code

Bei einer Breite und Höhe des Elements von 250px und einem Innenabstand von 0 ergibt sich bei einem Bild von 125 x 125 Pixel Größe eine Position von 62,5px vertikal wie horizontal.

```
<p style="background-position:50% 50%">Wenn der Browser Prozentwerte
   unterst&uuml;tzt, sollte das Hintergrundbild mittig angezeigt
   werden.</p>
<p style="background-position:center center">Wenn der Browser
Schl&uuml;sselw&ouml;rter (center center) unterst&uuml;tzt, sollte das
Hintergrundbild mittig angezeigt werden.</p>
<p style="background-position:50% center">Wenn der Browser eine gemischte
Notation (50% center) unterst&uuml;tzt, sollte das Bild mittig platziert
werden.</p>
<p style="background-position:50% 62.5px">Wenn der Browser eine gemischte
Angabe (50% 62.5px) unterst&uuml;tzt, sollte das Bild  zentriert
ausgerichtet werden.</p>
<p style="background-position:62.5px 62.5px">Wenn der Browser
   ausschlie&szlig;lich numerische Werte (62.5px 62.5px) mit Einheit
   akzeptiert, m&uuml;sste das Bild in der Mitte der Box stehen.</p>
```

Listing 7.1: Alle Methoden zum Zentrieren des Hintergrundbildes in einer Box

Für alle Absätze der Seite gilt außerdem der folgende Elementstil, der dafür sorgt, dass das Bild nicht gekachelt wird, und der die Größe der Boxen bestimmt.

```
p {
    width:250px; height:250px;
    background-image:url(blaetter.jpg);
    border:1px solid black;
    margin:3px;
    background-repeat:no-repeat;
    float:left
}
```

Sie finden das Beispiel auf der Buch-CD in der Datei BSP/K07/background_position.html.

Der Palm-Browser reduziert die Höhe der Boxen, um den Platz im Viewport besser ausnutzen zu können. Es scheint aber so, als ob die Positionierung korrekt wäre.

Hintergrundwiederholung (background-repeat)

Die Eigenschaft `background-repeat` bestimmt, ob und gegebenenfalls wie ein Hintergrundbild gekachelt wird.

CSS-Element: `background-repeat`	*Mögliche Werte: repeat*, `repeat-x`, `repeat-y`, `no-repeat`, `inherit`
CSS-Version: CSS 1-3, TV, Mobile	*Zulässig für folgende (X)HTML-Elemente:* alle
Medium: Visual	*Vererbt:* Nein

Palm	NN	Mozilla		Internet Explorer							Opera			Safari		iCab	Kq	FF
2.x	4.x	1.x	1.8	4.x	5.0x (Win)	5.0 (Mac)	5.1+ (Mac)	5.5 (Win)	6	7	6	7	8	1.x	2.0	2.9+	3.x	1.0
●	⊙ [1]	●	●	⊙ [2]	●	●	●	●	●	●	●	●	●	●	●	● [2]	●	●

[1] Nur der Wert `repeat` wird fehlerfrei ausgeführt, andere Werte nur in Abhängigkeit von anderen Angaben.
[2] Fehler in Kombination mit `background-position` bis einschließlich Version 2.9.x

Mögliche Werte

- `repeat`: Das Bild wird sowohl horizontal als auch vertikal wiederholt. Dies ist die Standardeinstellung.
- `repeat-x`: Das Bild wird nur horizontal gekachelt und vertikal an der mit `background-position` definierten Position bzw. oben angezeigt.
- `repeat-y`: Das Bild wird nur vertikal gekachelt.
- `no-repeat`: Das Bild wird nur einmal angezeigt und zwar links oben innerhalb des Elements, wenn es nicht explizit mit `background-position` gekachelt wurde.

Sie finden das Beispiel auf der Buch-CD in der Datei BSP/K07/background_repeat.html.

Der Internet Explorer 4 und iCab 2.9.x kacheln Bilder, die mit `background-repeat: repeat-y` gekachelt sind, nur von der Ausgangsposition nach unten und entsprechend bei `repeat-x` von der Ausgangsposition nach rechts. Eigentlich müsste das Hintergrundbild bei `repeat-x` nach links und rechts und bei `repeat-y` nach unten und oben gekachelt werden.

7.3 Nicht/schlecht unterstützte Attribute

In dieser Rubrik finden Sie solche Attribute, die von vielen oder zumindest von sehr wichtigen Browser nicht oder fehlerhaft unterstützt werden und die somit noch nicht praxistauglich sind.

Deckkraft (opacity)

Die Eigenschaft `opacity` legt die Deckkraft für ein Element fest. Sehr gut lässt sich damit die Deckkraft von Bildern beeinflussen oder ein halbtransparenter Schatten erzeugen.

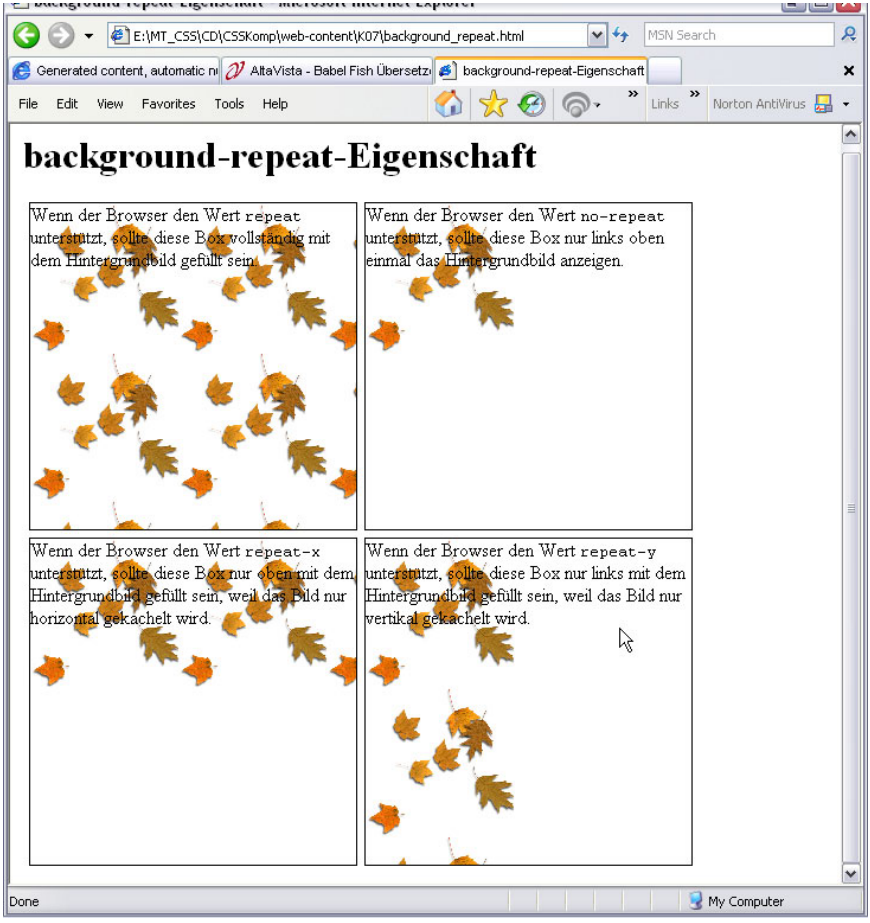


Abbildung 7.6: Korrekte Darstellung der Eigenschaft background-repeat im Internet Explorer

CSS-Element: `opacity`	*Mögliche Werte:* `inherit`, Alpha-Wert
CSS-Version: CSS 3	*Zulässig für folgende (X)HTML-Elemente:* alle
Medium: Visual	*Vererbt:* Nein

Palm	NN	Mozilla		Internet Explorer							Opera			Safari		iCab	Kq	FF
2.x	4.x	1.x	1.8	4.x	5.0x (Win)	5.0 (Mac)	5.1+ (Mac)	5.5 (Win)	6	7	6	7	8	1.x	2.0	2.9+	3.x	1.0
		●[1]	●												●		●	●

[1] ab Version 1.4+

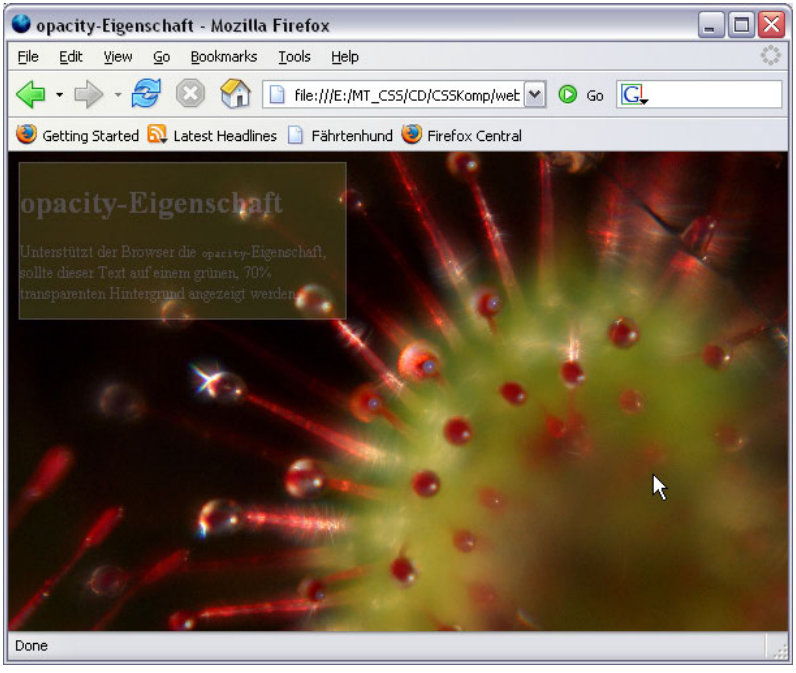
INFO

Ein Alpha-Wert ist ein Wert für den Alphakanal einer Farbe. Dabei handelt es sich um einen Wert von 0 bis 1. Je höher der Wert, desto größer die Deckkraft.

CD

Sie finden das Beispiel auf der Buch-CD in der Datei BSP/K07/opacity.html.

Abbildung 7.7: Teiltransparente Ebene auf einem Hintergrundbild



Beispiel

Das Beispiel zeigt den Einsatz der `opacity`-Eigenschaft. Zunächst wird ein Hintergrundbild und eine schwarze Hintergrundfarbe für die Seite (`body`-Element) festgelegt und darüber eine Ebene mit dem Seiteninhalt gelegt. Diese wird mit der CSS-Klasse `.halbtrans` formatiert, die mit `opacity: 0.3` eine Deckkraft von 30% festlegt. Die Ebene ist damit zu 70% transparent. Die Ebene (Hintergrundfarbe und Inhalt) deckt das Hintergrundbild also nur zu 30% ab.

Listing 7.2: Anwendung der opacity-Eigenschaft

```
<head>
    <title>opacity-Eigenschaft</title>
    <style type="text/css">
    <!--
        body {
            background-image:url(sonnentau74.JPG);
            background-repeat:no-repeat;
            background-color:black;
            color:white
        }
```

```
        .halbtrans {
            background-color: #9a9436;
            opacity:0.3;
            border:1px solid white;
            width:250px;
            display:block;
            color:white
        }
    -->
    </style>
</head>

<body>
    <div class="halbtrans">
    <h1>opacity-Eigenschaft</h1>
    <p>Unterst&uuml;tzt der Browser die <code>opacity</code>-Eigenschaft,
    sollte dieser Text auf einem grünen, 70% transparenten
    Hintergrund angezeigt werden.</p>
    </div>
</body>
```

Grundsätzlich wird das gesamte Element, das Sie mit `opacity` *formatieren, halbtransparent dargestellt. Das gilt folglich auch für den Rahmen.*

:-) TIPP

Wenn Sie um ein halbtransparentes Element einen Rahmen ziehen möchten, der die volle Deckkraft haben soll, fassen Sie das Element einfach in ein `div`*-Element ein und definieren Sie den Rahmen dann für dieses* `div`*-Element. Der HTML-Code dazu könnte wie folgt aussehen. Hier wird das Bild transparent dargestellt, der Rahmen darum jedoch mit voller Deckkraft.*

```
<div class="bild"><img src="SaracFlava20.JPG" alt="Saracenia
Flava"></div>
```

Bewirkt wird dies durch die beiden Stile:

```
img   { opacity:0.5; margin:0 }
.bild { position:absolute; top:150px; left:50%;
        border:1px solid white; padding:0 }
```

7.4 Praxisbeispiel

Basierend auf dem Layout aus Kapitel 5 erfahren Sie hier, wie Sie mithilfe von Hintergrundfarben eine Seite ansprechend gestalten können.

Banner gestalten

Zunächst sollten Sie dazu ein Hintergrundbild für das Seitenbanner definieren und den vorhanden Layoutrahmen entfernen. Dazu setzen Sie die `border`-Eigenschaft auf `none` und fügen die `background-color`-Eigenschaft und die `background-image`-Eigenschaft hinzu.

Listing 7.3: Formatieren des Banners

```
#banner {
    border: none;
    height:1em;
    font-size:3em;
    padding-top:0.25em;
    padding-bottom:0.25em;
    text-align:center;
    margin-bottom:1px;
    background-color:#75130a;
    background-image:url(bannerhgrd.jpg)
}
```

Seitenhintergrund

Auch den Hintergrund der Seite sollten Sie setzen. Hier bietet es sich an, die Farbe zu verwenden, die die Grundfarbe des Banners darstellt. Ergänzen Sie dazu einfach einen Elementstil für das `body`-Element. Über die `color`-Eigenschaft können Sie dann auch gleich eine ansprechende Vordergrundfarbe definieren.

```
body { background-color:#75130a; color:#f5c5a9 }
```

Die Schaltflächen formatieren

Für die Schaltflächen der Navigationsleisten sollten Sie eine Hintergrundfarbe und eine andere Rahmenfarbe definieren. Wenn Sie einen Hover-Effekt erzielen möchten, definieren Sie für die jeweiligen Links der Navigationsleiste einen Selektor mit dem Pseudo-Element `:hover` und dann für diesen Stil die umgekehrte Kombination aus Rahmen- und Hintergrundfarbe. Schließlich sollten Sie noch für den Stil `#vertnavigation` die `border`-Eigenschaft auf `none` setzen, um den Rahmen zu entfernen.

Listing 7.4: Notwendige Ergänzungen zum Formatieren der Navigationsleisten

```
#hornavigation a {
…
    margin-left:0;
    background-color:#e75c39;
    border:1px solid #f5c5a9
}
#hornavigation a:hover {
    background-color:#f5c5a9;
    border:1px solid #e75c39
}
#vertnavigation {
    border: none
    margin-top:2px;
    width:30%;
    min-width:150px;
    height:10em;
    float:left;
}
```

Abbildung 7.8: Geringe Änderungen bewirken schon eine ganze Menge ...

```
#vertnavigation a {
…
    background-color:#e75c39;
    border:1px solid #f5c5a9
}
#vertnavigation a:hover {
    background-color:#f5c5a9;
    border:1px solid #e75c39
}
```

8 Linien und Rahmen

Mit Rahmen können Sie nicht nur Teile der Webseite einrahmen und damit von anderen Elementen abgrenzen, sondern auch Links unterstreichen und die Standardformatierung der Links durch stärkere Linien, andere Farben oder Linienstile ersetzen.

8.1 Einführung

Der Rahmen eines Elements umgibt den Inhaltsbereich und den Innenabstand und liegt auf dem Hintergrundbereich des Elements. Seine Breite hat außerdem Einfluss auf die Größe des Elements, da die Rahmenstärke in die Gesamtbreite des Elements mit eingerechnet wird.

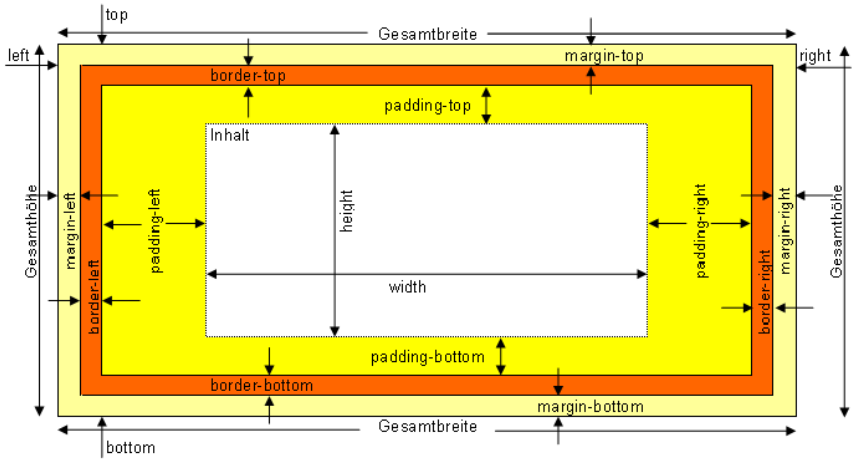


Abbildung 8.1: Das Boxmodell zeigt die Lage des Rahmens innerhalb der Box.

Weitere Informationen zum Boxmodell finden Sie in Kapitel 5, »Größen, Abstände und Positionierung«.

Rahmen eignen sich z.B. hervorragend, um die Standardunterstreichung von Hyperlinks durch eigene Formatierungen zu ersetzen oder aus einfachen Links richtige 3D-Schaltflächen zu erzeugen.

Ein Beispiel dazu finden Sie im Abschnitt »Praxisbeispiel«.

8.2 Praxistaugliche Attribute

Nachfolgend finden Sie CSS-Attribute, die Sie problemlos oder mit nur geringen Einschränkungen in der Praxis einsetzen können. Entweder ist die Browserunterstützung sehr gut oder aber ein Nichtausführen der CSS-Anweisung wirkt sich nicht so negativ aus, dass die Seite nicht mehr bedienbar ist.

Rahmen (border, border-top, border-right, border-left, border-bottom)

Die Eigenschaft `border` stellt eine zusammenfassende Eigenschaft dar, die die Werte für die Eigenschaften `border-color`, `border-width` und `border-style` für alle vier Kanten gleich definiert. Sie können mit den Eigenschaften `border-top`, `border-right`, `border-left` und `border-bottom` analog auch die Rahmen einzelner Kanten definieren.

CSS-Element: `border`, `border-top`, `border-right`, `border-bottom` , `border-left`	*Mögliche Werte:* siehe Einzeleigenschaften `border-width`, `border-style` und `border-color`
CSS-Version: CSS 1-3, TV, Mobile	*Zulässig für folgende (X)HTML-Elemente:* alle
Medium: Visual	*Vererbt:* Nein

Palm	NN	Mozilla		Internet Explorer							Opera			Safari		iCab	Kq	FF
2.x	4.x	1.x	1.8	4.x	5.0x (Win)	5.0 (Mac)	5.1+ (Mac)	5.5 (Win)	6	7	6	7	8	1.x	2.0	2.9+	3.x	1.0
●	●[1]	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●

[1] nur die `border`-Eigenschaft, nicht für Tabellen

Sie finden das Beispiel auf der Buch-CD in der Datei BSP/K08/border.html.

Einschränkungen und Inkompatibilitäten hinsichtlich einzelner Rahmenformatierungen finden Sie in den Beschreibungen der Einzeleigenschaften `border-width`, `border-style` *und* `border-color`.

Als Werte geben Sie nacheinander die Rahmenstärke, den Rahmenstil und die Rahmenfarbe an. Die Syntax lautet also:

```
border: border-width border-style border-color
```

Folgendes Listing zeigt die Anwendung der Eigenschaften.

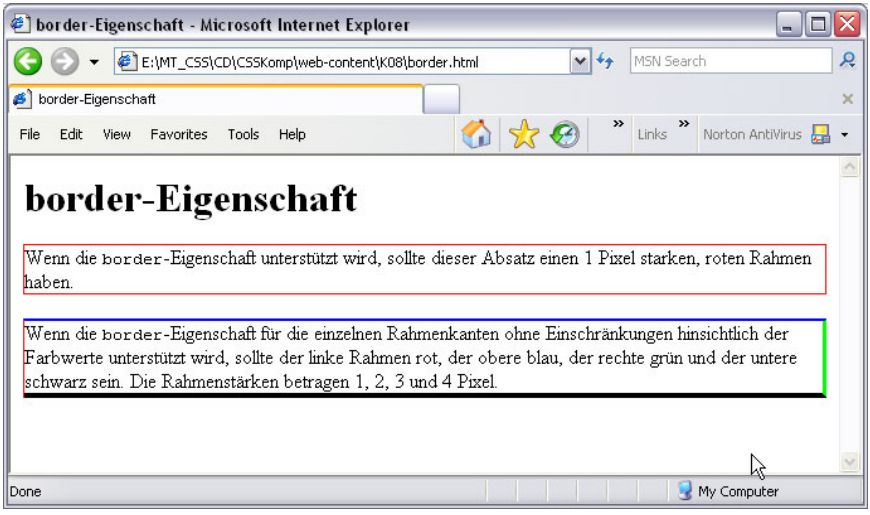


Abbildung 8.2: Das Ergebnis des Listings

```
<p style="border:1px solid red">Wenn die <code>border</code>-
   Eigenschaft unterst&uuml;tzt wird, sollte dieser Absatz einen 1
   Pixel starken, roten Rahmen haben.</p>
<p style="border-left:1px solid red; border-top:2px solid blue;border-
   right:3px solid lime;border-bottom:4px solid black">Wenn die
   <code>border</code>-Eigenschaft f&uuml;r die einzelnen Rahmenkanten
   ohne Einschr&auml;nkungen hinsichtlich der Farbwerte
   unterst&uuml;tzt wird, sollte der linke Rahmen rot, der obere blau,
   der rechte gr&uuml;n und der untere schwarz sein. Die
   Rahmenst&auml;rken betragen 1, 2, 3 und 4 Pixel.</p>
```

Listing 8.1: Definieren der Rahmen mit den zusammenfassenden Eigenschaften

Bei Verwendung eines Rahmens für Tabellen, für die mit dem `caption`-Element eine Tabellenüberschrift definiert wurde, wird der Rahmen nicht nur um die eigentliche Tabelle gezeichnet, sondern auch um die Überschrift. Dafür wird ein Rahmen, der für das `caption`-Element definiert wurde, gar nicht angezeigt. Das scheint keine fehlerhafte Implementierung der Rahmen zu sein, sondern daran zu liegen, dass iCab 2.9.x und früher für die Tabellenüberschrift keine eigene Box erzeugt. In iCab 3.0 tritt dieser Fehler nicht mehr auf.

Rahmenfarbe (border-color, border-top-color, border-right-color, border-left-color, border-bottom-color)

Die Eigenschaft `border-color` definiert die Rahmenfarbe eines Elements. Die Farbe der Rahmen an einzelnen Kanten können Sie analog dazu mit den Eigenschaften `border-top-color` (oben), `border-left-color` (links), `border-right-color` (rechts) und `border-bottom-color` (unten) festlegen.

Der Standardwert ist der Wert der `color`-Eigenschaft. Geben Sie keine Rahmenfarbe an, wird der Rahmen immer in der Vordergrundfarbe angezeigt.

CSS-Element: `border-color`, `border-top-color`, `border-right-color`, `border-bottom-color`, `border-left-color`	*Mögliche Werte:* Farbwert, `inherit`, `transparent`[1]
CSS-Version: CSS 1-3, TV, Mobile	*Zulässig für folgende (X)HTML-Elemente:* alle
Medium: Visual	*Vererbt:* Nein

Palm	NN	Mozilla		Internet Explorer							Opera			Safari		iCab	Kq	FF
2.x	4.x	1.x	1.8	4.x	5.0x (Win)	5.0 (Mac)	5.1+ (Mac)	5.5 (Win)	6	7	6	7	8	1.x	2.0	2.9+	3.x	1.0
● [2]	● [2]	●	●	●	●	●	●	●	●	●	●	●	●	●	●	● [2]	●	●

[1] In CSS 1 und 2 nur zulässig für die Eigenschaft `border-color`.
[2] siehe `border-width`-Eigenschaft

Nähere Informationen zu Farbwerten finden Sie in Kapitel 2, »Stile definieren«.

Für die Eigenschaft `border-color` *können Sie bis zu vier Werte festlegen, da dies die Kurzform für die Eigenschaften* `border-top-color`, `border-right-color`, `border-bottom-color` *und* `border-left-color` *darstellt. Geben Sie nur einen Wert an, gilt der für alle vier Rahmenkanten. Geben Sie vier Werte an, müssen Sie folgende Reihenfolge einhalten:*

```
border-color: border-top-color border-right-color border-bottom-
color border-left-color
```

Die einzelnen Werte werden durch Leerzeichen getrennt, und für jeden einzelnen Wert können Sie einen der nachfolgend erläuterten Werte angeben.

Sie finden das Beispiel auf der Buch-CD in der Datei BSP/K08/border_color.html.

Beispiel

Der folgende Codeausschnitt aus der Beispielseite zeigt die Anwendung der Eigenschaften. Zunächst wird die Rahmenfarbe für alle Kanten mit der `border-color`-Eigenschaft definiert. Anschließend werden abweichende Angaben für einzelne Kanten festgelegt.

```
<p style="border-color:#000; border-style:solid; border-left-color:red;
border-top-color:rgb(0,255,0); border-right-color:#0000FF;">Wenn die
border-color-Eigenschaft sowie die Eigenschaften f&uuml;r die einzelnen
Rahmenkanten ohne Einschr&auml;nkungen hinsichtlich der Farbwerte
unterst&uuml;tzt werden, sollte der linke Rahmen rot, der obere
gr&uuml;n, der rechte blau und der untere schwarz sein. </p>
```

Listing 8.2: Festlegen der Rahmenfarbe

Damit überhaupt ein Rahmen angezeigt wird, müssen Sie auch den Rahmenstil (`border-style`*) festlegen.*

Rahmenstärke (border-width, border-top-width, border-right-width, border-left-width, border-bottom-width)

Die Eigenschaft `border-width` definiert die Rahmenstärke eines Elements. Die Breite der Rahmen an einzelnen Kanten können Sie analog dazu mit den Eigenschaften `border-top-width` (oben), `border-left-width` (links), `border-right-width` (rechts) und `border-bottom-width` (unten) festlegen.

CSS-Element: `border-width`, `border-top-width`, `border-right-width`, `border-bottom-width`, `border-left-width`	*Mögliche Werte:* `thin`, *medium*, `thick`, numerischer Wert mit Einheit
CSS-Version: CSS 1-3, TV, Mobile	*Zulässig für folgende (X)HTML-Elemente:* alle
Medium: Visual	*Vererbt:* Nein

Palm	NN	Mozilla		Internet Explorer							Opera			Safari		iCab	Kq	FF
2.x	4.x	1.x	1.8	4.x	5.0x (Win)	5.0 (Mac)	5.1+ (Mac)	5.5 (Win)	6	7	6	7	8	1.x	2.0	2.9+	3.x	1.0
⊙ 1	○ 1	● 2	● 2	● 1	● 1	● 2+3	● 2+3	● 1	● 1	● 1	● 1	● 1	● 1	● 1	● 1	● 1+4	● 1	● 2

1 Rahmen mit einer Stärke von weniger als 1px werden nicht dargestellt.
2 Rahmen mit einer Stärke von weniger als 1px werden mit einer 1px-Stärke dargestellt.
3 Rahmen mit einer Stärke von weniger als 0.5px werden nicht dargestellt.
4 fehlerfrei ab Version 3.0

INFO

Für die Eigenschaft `border-width` *können Sie bis zu vier Werte festlegen, da dies die Kurzform für die Eigenschaften* `border-top-width`, `border-right-width`, `border-bottom-width`, `border-left-width` *darstellt. Geben Sie nur einen Wert an, gilt der für alle vier Rahmenkanten. Geben Sie vier Werte an, müssen Sie folgende Reihenfolge einhalten:*

```
border-width: border-top-width border-right-width border-bottom-
width border-left-width
```

Die einzelnen Werte werden durch Leerzeichen getrennt, und für jeden einzelnen Wert können Sie einen der nachfolgend erläuterten Werte angeben.

:-) TIPP

Eigentlich ist die Reihenfolge, in der die Formatierungen für die vier Kanten angegeben werden, ganz einfach: ausgehend von der oberen Kante im Uhrzeigersinn.

Mögliche Werte

- `thin`: erzeugt einen dünnen Rahmen.
- `thick`: erzeugt einen dicken Rahmen.
- `medium`: erzeugt einen mittleren Rahmen, der dicker als `thin` und dünner als `thick` dargestellt werden muss.

!! STOP

Wie dick die Rahmen dargestellt werden, hängt vom Browser ab. Der CSS-Standard macht dazu keine Vorschriften. Da jedoch die Rahmenstärke auch Einfluss auf die Gesamtgröße eines Elements hat, ist dies nicht immer optimal, wenn es auf exakte Positionen und Größen ankommt.

- Numerischer Wert mit Einheit: numerische Werte müssen positiv sein und legen die exakte Rahmenstärke fest.

CD

Sie finden das Beispiel auf der Buch-CD in der Datei BSP/K08/border_width.html.

Beispiel

Der folgende Codeausschnitt aus der Beispielseite zeigt die Anwendung der Eigenschaften. Zunächst wird der Rahmen für alle Kanten mit der `border-width`-Eigenschaft definiert. Anschließend werden abweichende Angaben für einzelne Kanten festgelegt.

Listing 8.3: Festlegen der Rahmenstärke mithilfe von Schlüsselwörtern

```
<p style="border-color:black; border-style:solid; border-
width:thin;border-left-width:thin; border-top-width:medium; border-
right-width:thick;">Falls Schl&uuml;sselw&ouml;rter f&uuml;r die
Einzeleigenschaften und <code>border-width</code> unterst&uuml;tzt
werden, sollte der linke Rahmen d&uuml;nn, der obere mittel, der rechte
dick und der untere wieder d&uuml;nn sein.</p>
```

Damit überhaupt ein Rahmen angezeigt wird, müssen Sie auch die Rahmenfarbe (`border-color`) und den Rahmenstil (`border-style`) festlegen.

!! STOP

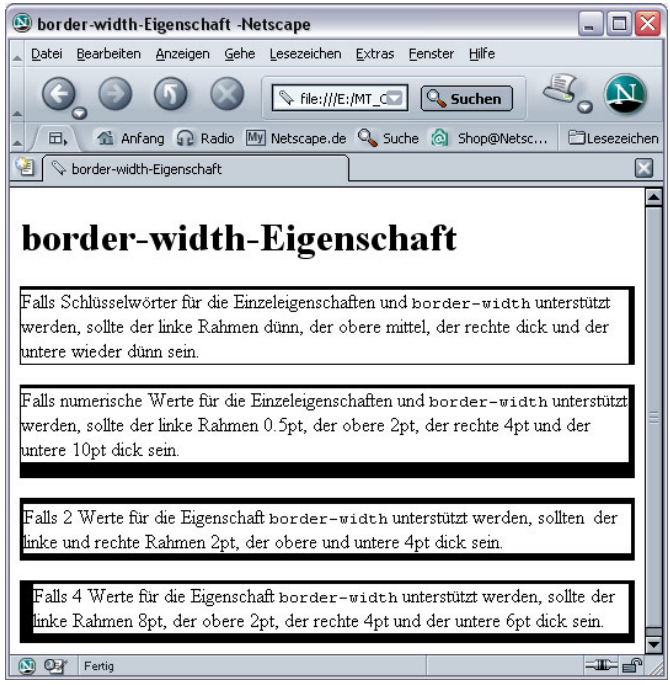


Abbildung 8.3: Die Beispielseite in Netscape 7.1

Numerische Werte kleiner als 1 sind zwar nicht explizit verboten, der CSS-Standard sagt aber auch nichts dazu aus, wie die Browser mit solchen Werten verfahren sollen. Im Test hat sich gezeigt, dass die Browser recht unterschiedlich damit umgehen. Ergeben sich nach Umrechnung der Einheit in Pixel Werte, die kleiner als ein Pixel sind, stellen alle auf Mozilla-Gecko basierenden Browser die Rahmen mit einer Stärke von einem Pixel dar. Die anderen Browser verfahren da recht unterschiedlich und zeigen meist Rahmenstärken unter einem Pixel nicht an. Einzig der Internet Explorer 5.x für Mac macht noch weitere Unterschiede. Er stellt alle Werte größer und gleich 0.5px mit einem Pixel dar und zeigt Rahmen in kleineren Stärken gar nicht an.

!! STOP

Vorsicht ist hier vor allem bei Angaben in der Einheit Punkt geboten. Die Umrechnung von `1pt` in Pixel führt nämlich zu unterschiedlichen Werten (abhängig von Ausgabemedium und Auflösung des Ausgabemediums). So kann es passieren, dass eine Rahmenstärke von 0.75 pt auf dem PC noch dargestellt wird, weil die Umrechnung einen Wert von 1px oder größer ergibt. Auf einem anderen Ausgabegerät könnte sich jedoch ein Wert unter 1px ergeben, der dann nicht mehr dargestellt wird.

i INFO

Der Netscape Navigator unterstützt den Standardwert `medium` nicht. Ohne Angabe der Rahmenstärke wird daher kein Rahmen angezeigt, wenn Rahmenfarbe und Rahmenstil festgelegt wurden. Legen Sie für einzelne Kanten beispielsweise mit `border-left-color` eine Rahmenfarbe fest, müssten Sie auch explizit für diese Kante mit `border-left-width` die Stärke und mit `border-left-style` den Stil festlegen. Bei der Angabe `border-left-color:red; border-left-style:solid;` **`border-width`**`:medium` würde ebenfalls kein Rahmen angezeigt werden. Darüber hinaus stellt der Netscape Navigator einen weißen Abstand zwischen Hintergrund und Rahmen dar, auch dann, wenn `padding` auf 0 gesetzt ist. Man kann also nicht mal sagen, dass der Browser fälschlicherweise den Innenabstand nicht füllt.

iCab 2.9.x verhält sich ganz ähnlich wie der Netscape Navigator. Allerdings reicht es für iCab aus, wenn Sie die Rahmenstärke festlegen. Darüber hinaus gibt es Darstellungsfehler, wenn Sie einen Rahmen definieren und gleichzeitig einen Außenabstand mit `margin` festlegen. Näheres dazu finden Sie im Abschnitt *»Rahmenstil (border-style, border-top-style, border-right-style, border-left-style, border-bottom-style)«*.

Der Palm-Browser zeigt rechts und unten einen Pixel zu wenig Rahmen an, wenn Sie gleichzeitig eine Hintergrundfarbe definieren. Dann verdeckt der Hintergrund Teile des Rahmens. Zudem wird der Rahmen nicht auf dem Hintergrund gezeichnet.

Rahmenstil (border-style, border-top-style, border-right-style, border-left-style, border-bottom-style)

Die Eigenschaft `border-style` definiert den Rahmenstil eines Elements. Den Stil der Rahmen an einzelnen Kanten können Sie analog dazu mit den Eigenschaften `border-top-style` (oben), `border-left-style` (links), `border-right-style` (rechts) und `border-bottom-style` (unten) festlegen.

CSS-Element: `border-style`, `border-top-style`, `border-right-style`, `border-bottom-style` , `border-left-style`	*Mögliche Werte: none*, `hidden`, `dotted`, `dashed`, `solid`, `double`, `groove`, `ridge`, `inset`, `outset`, `inherit`
CSS-Version: CSS 1-3, TV, Mobile	*Zulässig für folgende (X)HTML-Elemente:* alle
Medium: Visual	*Vererbt:* Nein

Palm	NN	Mozilla		Internet Explorer							Opera			Safari		iCab	Kq	FF
2.x	4.x	1.x	1.8	4.x	5.0x (Win)	5.0 (Mac)	5.1+ (Mac)	5.5 (Win)	6	7	6	7	8	1.x	2.0	2.9+	3.x	1.0
○ 4+5	○ 1+2+6	●	●	● 2	● 3	⊙ 6	⊙ 6	⊙ 5	⊙ 5	⊙ 5	●	●	●	●	●	● 1+4+7	●	●

1 siehe `border-width`-Eigenschaft
2 ohne `dotted`, `dashed`, `hidden`
3 ohne `dotted` und `dashed`
4 ohne `groove`, `ridge`, `inset` und `outset`
5 `dotted` wird teilweise fehlerhaft dargestellt
6 `hidden` wird fehlerhaft angezeigt
7 ab Version 3.0 fehlerfrei und komplett

Mögliche Werte

- `none`: Es wird kein Rahmen angezeigt. Gleichzeitig müssen Sie `border-width` auf 0 setzen oder nicht angeben.
- `hidden`: Außerhalb von Tabellen wird der Wert wie `none` angezeigt.
- `dotted`: Der Rahmen wird gepunktet dargestellt.
- `dashed`: Der Rahmen wird gestrichelt angezeigt.
- `solid`: Der Rahmen besteht aus einer durchgezogenen Linie.
- `double`: Der Rahmen ist eine doppelte Linie. Welche Stärke die einzelnen Linien haben, ist nicht definiert, nur müssen sie zusammen mit dem Abstand zwischen ihnen die definierte Rahmenbreite ergeben. Daher macht dieser Stil eine Mindestbreite von 3px erforderlich.
- `groove`: Die Rahmenlinie erscheint als vertiefte 3D-Linie.
- `ridge`: Die Rahmenlinie ist eine erhöhte 3D-Linie.
- `inset`: Die Box, die von der Rahmenlinie umgeben wird, erscheint vertieft.
- `outset`: Die Box, die von der Rahmenlinie umgeben wird, erscheint erhöht.

!! STOP

Damit die Rahmenstile `groove`, `ridge`, `inset`, `outset` *und* `double` *deutlich sichtbar sind, sollten Sie mindestens eine Rahmenstärke von* `3px` *definieren. Die Farbe, die Sie für den Rahmen festlegen, wird bei den Stilen* `groove`, `ridge`, `inset` *und* `outset` *vom Browser abgedunkelt, aufgehellt oder anderweitig verändert, um den gewünschten Effekt zu erzielen.*

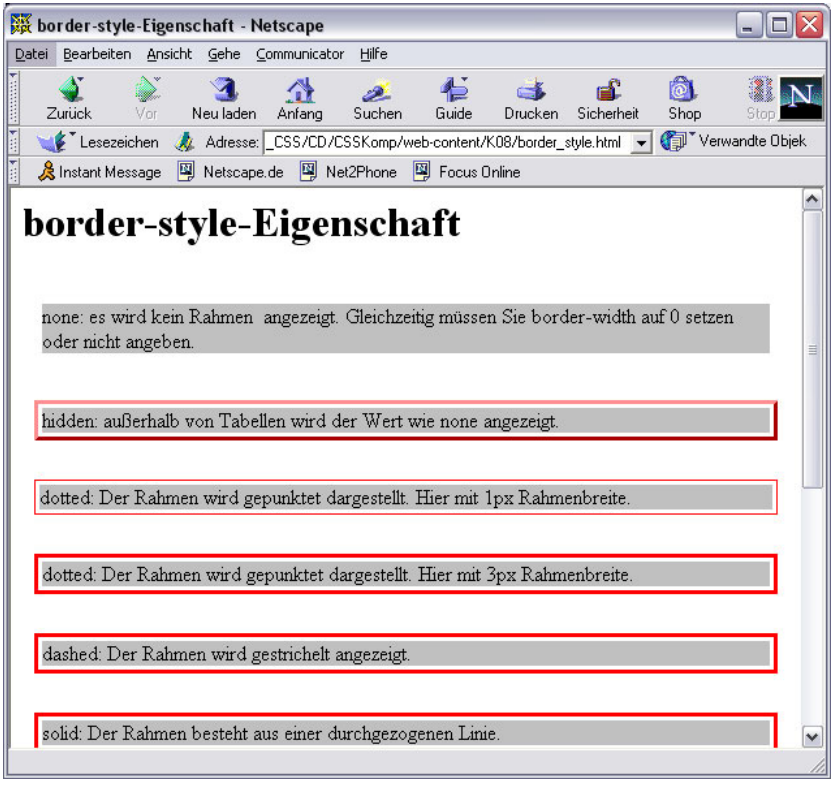
Gemäß CSS-Standard dürfen die Stile `dotted`, `dashed`, `double`, `groove`, `ridge`, `inset` *und* `outset` *auch als* `solid` *angezeigt werden. In diesem Fall verhält sich der Browser immer noch standardkonform.*

Rahmen werden immer auf den Hintergrund gezeichnet. Bei gepunkteten und gestrichelten Rahmenlinien muss die Hintergrundfarbe der Box also in den Rahmenlücken sichtbar sein.

Sie finden das Beispiel auf der Buch-CD in der Datei BSP/K08/border_style.html.

Der Netscape Navigator stellt `dotted` und `dashed` als durchgezogene Linien dar, `hidden` wird wie `outset` fehlerhaft angezeigt. Zudem wird ein weißer Abstand zwischen Rahmenlinie und Hintergrund eingeblendet.

Abbildung 8.4: Fehlerhafte Darstellung im Netscape Navigator



iCab 2.9.x stellt die Werte `groove`, `ridge`, `inset` und `outset` einheitlich dar, aber nicht als durchgezogene Linie. Es lässt sich nicht eindeutig sagen, welchem Wert die Anzeige am nächsten kommt. Das ist allerdings das geringere Problem. Bei allen Werten außer `none` ragt die Schrift der oberen Inhaltszeile über die Rahmenkante hinaus, wenn gleichzeitig mit einem Rahmen auch ein Außenabstand (`margin`) festgelegt wird. Ab Version 3.0 wird die Eigenschaft fehlerfrei unterstützt.

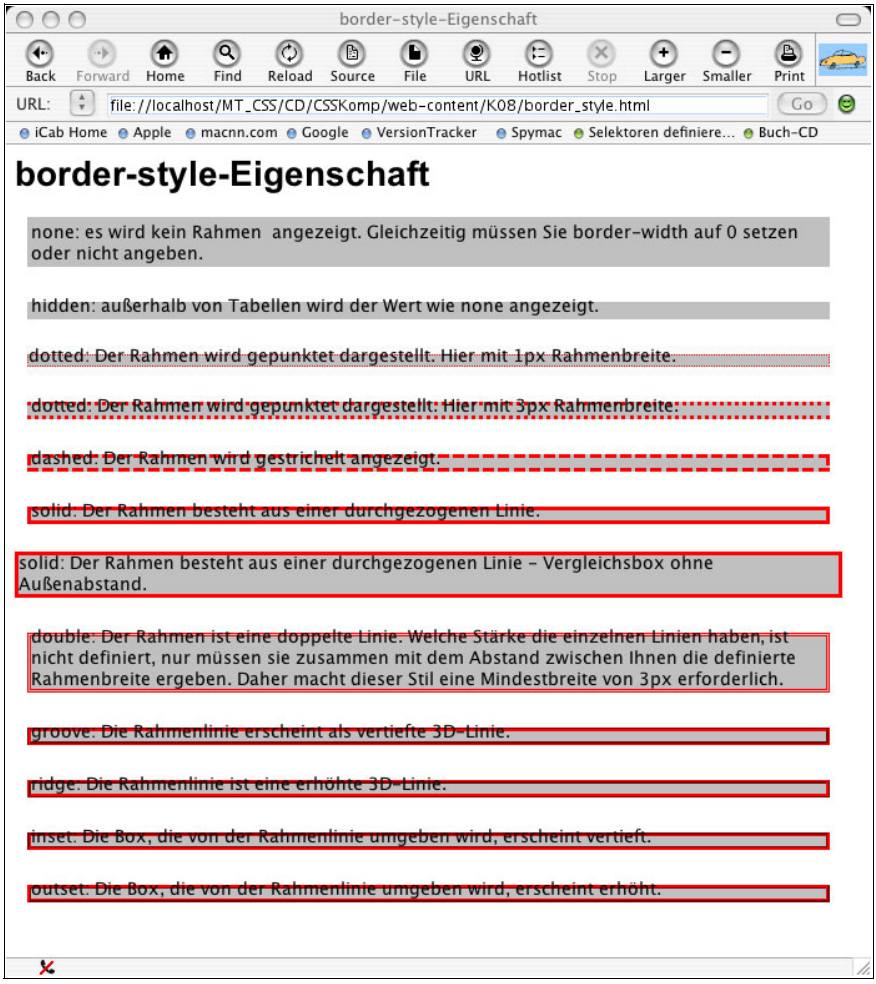


Abbildung 8.5:
Darstellung in iCab

Es gibt allerdings eine Lösung für das Problem – das heißt, im Prinzip sogar zwei. Wenn Sie den gleichen Wert, den Sie für den Außenabstand verwendet haben, auch für den oberen Innenabstand (`padding-top`*) festlegen, zeigt iCab 2.9.x das Element korrekt an. Allerdings zeigen alle anderen Browser dann einen zu großen Innenabstand an, nämlich ebenfalls den definierten Wert. Besser ist daher die zweite Lösung. Statt den Außenabstand für das Element mit dem Rahmen zu definieren, setzen Sie dessen Außenabstand auf 0 und legen dafür ein* `div`*-Element um das Element herum, für das Sie dann den Außenabstand festlegen.*

:-)
TIPP

```
<div style="margin:10px"><p style="border-style:solid;border-
width:3px;border-color:red;background-color:silver;margin:0">solid:
Der Rahmen besteht aus einer durchgezogenen Linie - Vergleichsbox
ohne Außenabstand mit umgebenden div-Element für den
Außenabstand.</p></div>
```

Der Internet Explorer 5.5 und höher für Windows stellt den Wert `dotted` bei einer Rahmenstärke von einem Pixel wie `dashed`, d.h. gestrichelt dar. Bei höheren Rahmenstärken wird er korrekt angezeigt und sogar mit abgerundeten Punkten, während die allermeisten anderen Browser die einzelnen Punkte als Quadrate anzeigen, genau wie die Mac-Version des Internet Explorers.

Der Internet Explorer 5.2 für Mac stellt den Wert `hidden` wie `solid` dar.

Für die Eigenschaft `border-style` *können Sie bis zu vier Werte festlegen, da dies die Kurzform für die Eigenschaften* `border-top-style`, `border-right-style`, `border-bottom-style`, `border-left-style` *darstellt. Geben Sie nur einen Wert an, gilt der für alle vier Rahmenkanten. Geben Sie vier Werte an, müssen Sie folgende Reihenfolge einhalten:*

```
border-style: border-top-style border-right-style border-bottom-
style border-left-style
```

Die einzelnen Werte werden durch Leerzeichen getrennt, und für jeden einzelnen Wert können Sie einen der nachfolgend erläuterten Werte angeben.

Der Wert `dotted` wird bei einer Rahmenstärke von `1px` als durchgezogene Linie dargestellt. Bei größeren Rahmenstärken haben die Punkte senkrecht und waagerecht unterschiedliche Abstände. Die Rahmenlinien werden außerdem nicht auf einem Hintergrund angezeigt, sondern umschließen den Hintergrund der Box, was nicht korrekt ist. Bei gleichzeitiger Definition eines Hintergrundes wird dieser um einen Pixel nach rechts unten verschoben. Dadurch entsteht links und oben ein 1 Pixel starker Streifen in der Farbe des Hintergrundes der umschließenden Box. Dafür verdeckt der Hintergrund in einer Stärke von einem Pixel an der rechten und unteren Kante den Rahmen. Das bewirkt, dass bei einem Rahmen, der nur einen Pixel breit ist, und einem definierten Hintergrund rechts und unten kein Rahmen sichtbar ist.

Damit überhaupt ein Rahmen angezeigt wird, müssen Sie auch die die Rahmenstärke (`border-width`*) und Rahmenfarbe (*`border-color`*) festlegen.*

Für den Netscape Navigator 4.x und iCab 2.9.x sollten Sie zusätzlich auch die Rahmenfarbe definieren. Mehr dazu finden Sie im Abschnitt »Rahmenstärke (border-width, border-top-width, border-right-width, border-left-width, border-bottom-width)«.

Auch wenn die Übersicht weiter oben recht schlecht aussieht, was die Unterstützung der Browser angeht, lässt sich die `border-style`-Eigenschaft durchaus in der Praxis einsetzen. Auf den Wert `hidden` sollten Sie allerdings verzichten und für eine korrekte Darstellung in iCab 2.9.x darauf achten, keinen Außenrand für Elemente mit Rahmen zu definieren. Alle anderen Fehler und Einschränkungen der Browser verursachen keine Fehler, die das Layout einer Seite zerstören würden.

Browserunterstützung

Umrandung (outline)

Mit der Eigenschaft `outline` können Sie eine zusätzliche Umrandung (zusätzlich zum Rahmen) definieren. Die Umrandung liegt über dem Element und verdeckt damit unter Umständen auch angrenzende Elemente, wenn diese in der Stapelreihenfolge unter dem umrandeten Element liegen. Umrandungen unterscheiden sich ganz erheblich von Rahmen und zwar durch folgende Punkte:

- sie müssen nicht zwingend rechteckig sein,
- sie benötigen keinen Platz (siehe Boxmodell).

Sie können das testen, indem Sie die Beispielseite aufrufen. Dort werden in den letzten beiden Absätzen die Unterschiede zwischen Umrandung und Rahmen demonstriert. Fahren Sie dazu einfach mit der Maus über die Links.

:-) TIPP

CSS-Element: `outline`	*Mögliche Werte:* `inherit`, `outline`-Wert
CSS-Version: CSS 2, CSS 3, TV	*Zulässig für folgende (X)HTML-Elemente:* alle
Medium: Visual, Interaktiv	*Vererbt:* Nein

Palm	NN	Mozilla		Internet Explorer							Opera			Safari		iCab	Kq	FF
2.x	4.x	1.x	1.8	4.x	5.0x (Win)	5.0 (Mac)	5.1+ (Mac)	5.5 (Win)	6	7	6	7	8	1.x	2.0	2.9+	3.x	1.0
			●				○ [1]					●	●		○ [2]		○ [2]	

1 Nicht für alle Elemente; Eingabefelder werden beispielsweise unterstützt, Hyperlinks nicht.

2 Nicht für alle Elemente; Eingabefelder werden beispielsweise nicht unterstützt, Hyperlinks schon.

INFO

Als Outline-Wert müssen Sie eine Kombination aus den Einzelwerten `outline-width`, `outline-style` *und* `outline-color` *angeben. Leider sieht hier die Reihenfolge etwas anders aus als bei der* `border`*-Eigenschaft. Die Syntax lautet:*

```
outline: outline-color outline-style outline-width
```

Die Werte entsprechen denen der Einzeleigenschaften. Möchten Sie also eine rote, gepunktete Umrandung mit 1 Pixel Stärke festlegen, geben sie dazu an:

```
outline:red dotted 1px
```

CD

Sie finden das Beispiel auf der Buch-CD in der Datei BSP/K08/outline.html.

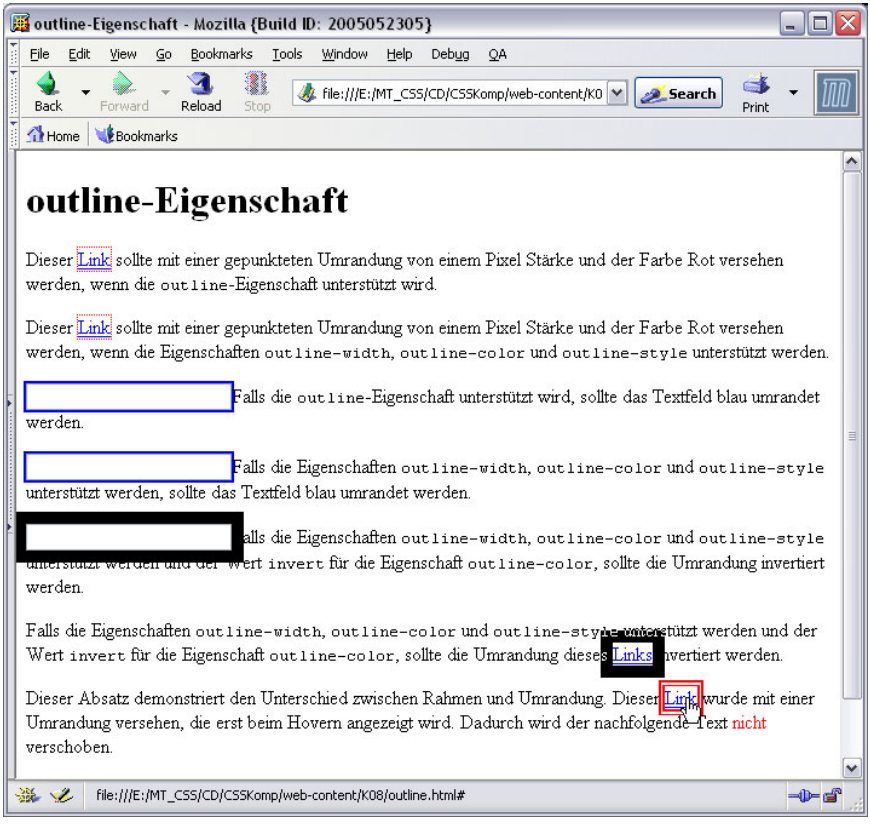


Abbildung 8.6: Korrekte Darstellung der Umrandungen für Links und Eingabefelder in Mozilla 1.8

Optimal lässt sich die `outline`*-Eigenschaft einsetzen, um Elemente hervorzuheben, die gehovert werden. Dazu verwenden Sie die Eigenschaften einfach in einem Stil für das Pseudo-Element* `:hover`*. Mit dem folgenden Stil werden beispielsweise Hyperlinks mit einem doppelten Rahmen versehen, wenn sie gehovert werden:*

TIPP

```
a:hover { outline: red double 5px }
```

Trotz der nicht gerade guten Browserunterstützung ist die Umrandung in der Praxis einsetzbar, da Seiten, die die Eigenschaft verwenden, in der Regel auch dann brauchbar und benutzbar bleiben, wenn der Browser die Eigenschaft nicht unterstützt.

INFO

Auf dem Macintosh wird bei keinem Browser eine Umrandung für Formularfelder angezeigt. Das kann daran liegen, dass das Betriebssystem Elemente mit Fokus bereits mit einer Umrandung versieht.

Mac

Umrandungsbreite (outline-width)

Die Eigenschaft `outline-width` legt die Breite der Umrandung fest. Die Umrandung liegt über einem Element und benötigt im Gegensatz zum Rahmen keinen Platz

CSS-Element: `outline-width`	*Mögliche Werte:* `inherit`, numerischer Wert mit Einheit, `thin`, *medium*, `thick`
CSS-Version: CSS 2, CSS 3, TV	*Zulässig für folgende (X)HTML-Elemente:* alle
Medium: Visual, Interaktiv	*Vererbt:* Nein

Palm	NN	Mozilla		Internet Explorer							Opera			Safari		iCab	Kq	FF
2.x	4.x	1.x	1.8	4.x	5.0x (Win)	5.0 (Mac)	5.1+ (Mac)	5.5 (Win)	6	7	6	7	8	1.x	2.0	2.9+	3.x	1.0
			●				○ [1]					●	●		○ [2]		○ [2]	

1 Nicht für alle Elemente; Eingabefelder werden beispielsweise unterstützt, Hyperlinks nicht.

2 Nicht für alle Elemente; Eingabefelder werden beispielsweise nicht unterstützt, Hyperlinks schon.

Sie finden das Beispiel auf der Buch-CD in der Datei BSP/K08/outline.html.

Die Werte `thin`, `medium` *und* `thick` *entsprechen denen der Eigenschaften* `border-width`, *die Sie im Abschnitt »Rahmenstärke (border-width, border-top-width, border-right-width, border-left-width, border-bottom-width)« erläutert finden.*

Umrandungsfarbe (outline-color)

Die Eigenschaft `outline-color` legt die Farbe der Umrandung fest. Die Umrandung liegt über einem Element und benötigt im Gegensatz zum Rahmen keinen Platz.

CSS-Element: `outline-color`	*Mögliche Werte:* `inherit`, *invert*, Farbwert
CSS-Version: CSS 2, CSS 3, TV	*Zulässig für folgende (X)HTML-Elemente:* alle
Medium: Visual, Interaktiv	*Vererbt:* Nein

Palm	NN	Mozilla		Internet Explorer							Opera			Safari		iCab	Kq	FF
2.x	4.x	1.x	1.8	4.x	5.0x (Win)	5.0 (Mac)	5.1+ (Mac)	5.5 (Win)	6	7	6	7	8	1.x	2.0	2.9	3.x	1.0
			● 3				○ 1					● 3	● 3		○ 2+3		○ 2+3	

1 Nicht für alle Elemente; Eingabefelder werden beispielsweise unterstützt, Hyperlinks nicht.

2 Nicht für alle Elemente; Eingabefelder werden beispielsweise nicht unterstützt, Hyperlinks schon.

3 Beim Wert `invert` wird die Umrandung über nachfolgende Elemente gelegt, auch wenn diese in der Stapelreihenfolge über dem umrandeten Element liegen.

Sie finden das Beispiel auf der Buch-CD in der Datei BSP/K08/outline.html.

Der Wert `invert` legt fest, dass die Rahmenfarbe invertiert werden soll, so dass der Rahmen unabhängig von der Hintergrundfarbe sichtbar ist. Allerdings definiert der CSS-Standard nicht, welche Farbkombination als Ausgangswert verwendet wird.

Mozilla und der Opera-Browser verwenden dazu die Vordergrund- und Hintergrundfarben des umgebenden Elements bzw. des `body`-Elements.

Safari verwendet als Basis die Farben des umrandeten Elements.

Umrandungsstil (outline-style)

Die Eigenschaft `outline-style` legt den Stil der Umrandung fest.

CSS-Element: `outline-style`	*Mögliche Werte: none*, `hidden`, `dotted`, `dashed`, `solid`, `double`, `groove`, `ridge`, `inset`, `outset`, `inherit`
CSS-Version: CSS 2, CSS 3, TV	*Zulässig für folgende (X)HTML-Elemente:* alle
Medium: Visual, Interaktiv	*Vererbt:* Nein

Palm	NN	Mozilla		Internet Explorer							Opera			Safari		iCab	Kq	FF
2.x	4.x	1.x	1.8	4.x	5.0x (Win)	5.0 (Mac)	5.1+ (Mac)	5.5 (Win)	6	7	6	7	8	1.x	2.0	2.9+	3.x	1.0
			●				○ 1					●	●		○ 2		○ 2	

1 Nicht für alle Elemente; Eingabefelder werden beispielsweise unterstützt, Hyperlinks nicht.

2 Nicht für alle Elemente; Eingabefelder werden beispielsweise nicht unterstützt, Hyperlinks schon.

Sie finden das Beispiel auf der Buch-CD in der Datei BSP/K08/outline.html.

Die Werte haben die gleiche Bedeutung wie bei der `border-style`*-Eigenschaft, die im Abschnitt »Rahmenstil (border-style, border-top-style, border-right-style, border-left-style, border-bottom-style)« beschrieben wird.*

8.3 Nicht/schlecht unterstützte Attribute

In dieser Rubrik finden Sie solche Attribute, die von vielen oder zumindest von sehr wichtigen Browser nicht oder fehlerhaft unterstützt werden und die somit noch nicht praxistauglich sind.

Box-Schatten (box-shadow)

Mit der Eigenschaft `box-shadow` können Sie einen Schatten für eine Box definieren. Er wird allerdings nur dann angezeigt, wenn für die Box auch ein Rahmen definiert wurde – und zwar nur an den Kanten, für die es einen sichtbaren Rahmen gibt.

CSS-Element: `box-shadow`	*Mögliche Werte: none*, Schattenwerte
CSS-Version: CSS 3	*Zulässig für folgende (X)HTML-Elemente:* alle Elemente
Medium: Visual	*Vererbt:* Nein

Palm	NN	Mozilla		Internet Explorer							Opera			Safari		iCab	Kq	FF
2.x	4.x	1.x	1.8	4.x	5.0x (Win)	5.0 (Mac)	5.1+ (Mac)	5.5 (Win)	6	7	6	7	8	1.x	2.0	2.9+	3.x	1.0

Der Wert `none` *bewirkt, dass es keinen Schatten gibt. Als Schattenwert müssen Sie drei numerische Werte und einen Farbwert angeben. Die numerischen Werte müssen eine Einheit haben und legen der Reihe nach den horizontalen Versatz, den vertikalen Versatz und die Breite des Schattens fest. Der Farbwert definiert die Farbe des Schattens.*

Folgende CSS-Klasse definiert einen grauen Schatten und einen schwarzen Rahmen um das Element, auf das es angewendet wird:

```
.schatten {
    border:1px solid black;
    box-shadow:3px 5px 5px gray;
    background-color:white
}
```

Sie finden das Beispiel auf der Buch-CD in der Datei BSP/K08/box_shadow.html.

Abbildung 8.7: So sollten Browser den Schatten darstellen.

Rahmenbruch (border-break)

Mit der Eigenschaft `border-break` können Sie bestimmen, welcher Rahmen an der Bruchlinie einer Box angezeigt werden soll, wenn die Box durch einen Seitenumbruch beispielsweise beim Ausdruck geteilt wird.

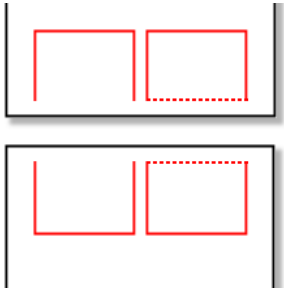
CSS-Element: `border-break`	*Mögliche Werte: none*, Rahmen-Wert
CSS-Version: CSS 3	*Zulässig für folgende (X)HTML-Elemente:* alle Elemente mit einem Rahmen
Medium: Visual	*Vererbt:* Ja

Palm	NN	Mozilla		Internet Explorer							Opera			Safari		iCab	Kq	FF
2.x	4.x	1.x	1.8	4.x	5.0x (Win)	5.0 (Mac)	5.1+ (Mac)	5.5 (Win)	6	7	6	7	8	1.x	2.0	2.9+	3.x	1.0

Der Standardwert ist `none`*, er bewirkt, dass bei einem Seitenumbruch kein Rahmen angezeigt wird. Wünschen Sie einen Rahmen, geben Sie dazu wie bei der* `border`*-Eigenschaft die Rahmenbreite, den Rahmenstil und die Rahmenfarbe an. Mit* `border-break:1px dotted red` *wird eine gepunktete Rahmenlinie an der Bruchstelle gezeichnet.*

Sie finden das Beispiel auf der Buch-CD in der Datei BSP/K08/border_break.html. *Dort wird je ein Rahmen definiert. Der linke enthält die Einstellung* `border-break:none`*, der rechte wurde mit* `border-break:1px dotted red` *definiert.*

Abbildung 8.8: So sollten die Browser die Beispielseite darstellen.



Rahmenradius (border-radius, border-top-left-radius, border-top-right-radius, border-bottom-right-radius, border-bottom-left-radius)

Mit der Eigenschaft `border-radius` können Sie den Radius eines Rahmens in den vier Ecken der Box festlegen und damit abgerundete Ecken erzielen. Die Eigenschaften `border-top-left-radius`, `border-top-right-radius`, `border-bottom-right-radius`, `border-bottom-left-radius` legen den Radius für einzelne Ecken fest, nämlich oben-links, oben-rechts, unten-rechts und unten-links.

CSS-Element: `border-radius`, `border-top-left-radius`, `border-top-right-radius`, `border-bottom-right-radius`, `border-bottom-left-radius`	*Mögliche Werte: 0*, zwei numerische Werte mit Einheit
CSS-Version: CSS 3	*Zulässig für folgende (X)HTML-Elemente:* alle Elemente mit Rahmen, ausgenommen Tabellenzellen mit `border-collapse:collapse`
Medium: Visual	*Vererbt:* Nein

Palm	NN	Mozilla		Internet Explorer							Opera			Safari		iCab	Kq	FF
2.x	4.x	1.x	1.8	4.x	5.0x (Win)	5.0 (Mac)	5.1+ (Mac)	5.5 (Win)	6	7	6	7	8	1.x	2.0	2.9+	3.x	1.0

Für die Eigenschaften müssen Sie zwei numerische Werte mit Einheit angeben. Der erste gibt den horizontalen Radius (X-Wert) und der zweite den vertikalen Radius (Y-Wert) an. Falls Sie den zweiten Wert weglassen, gibt der erste Wert beide Radien an. Standardwert für die Eigenschaft ist 0 und entspricht damit keiner Rundung.

Abbildung 8.9: Angaben für die Eckrundungen

Sie finden das Beispiel auf der Buch-CD in der Datei BSP/K08/border_radius.html.

Folgendes Beispiel zeigt die Verwendung der Eigenschaft. Für den ersten Absatz wird jede Ecke mit `25px` gerundet, im zweiten Absatz wird nur die linke Ecke gerundet.

```
<p style="border:1px solid red;border-radius:25px 25px">Wenn die
   <code>border-radius</code>-Eigenschaft unterst&uuml;tzt wird, sollte
   dieser Absatz einen 1 Pixel starken, roten Rahmen haben, dessen
   Ecken mit 25px,25px abgerundet sind.</p>
<p style="border:2px solid blue;border-top-left-radius:50px 25px">Wenn
   die <code>border-top-left-radius</code>-Eigenschaft unterst&uuml;tzt
   wird, sollte die obere linke Ecke abgerundet sein. </p>
```

Listing 8.4: Anwendung der Eigenschaften

Derzeit unterstützt kein Browser diese Eigenschaft. Wenn das jedoch der Fall wäre, sollte der zweite Absatz in etwa wie in Abbildung 8.10 aussehen.

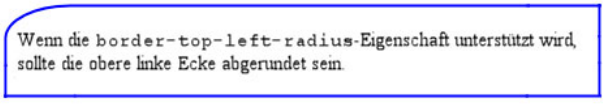


Abbildung 8.10: So sollte der zweite Absatz formatiert werden.

Zurzeit unterstützt noch kein Browser diese Eigenschaft gemäß CSS-Standard. Die Mozilla-Browser bieten jedoch eine Implementierung für abgerundete Ecken, die Sie über die Eigenschaften `-moz-border-radius`, `-moz-border-radius-topleft`, `-moz-border-radius-topright`, `-moz-border-radius-bottomright` und `-moz-border-radius-bottomleft` definieren können. Die Eigenschaften sind zwar nicht CSS 3-konform, bieten aber schon mal einen Vorgeschmack auf die neuen Möglichkeiten. Beachten Sie jedoch, dass Sie bei den `-moz-border-radius`-Eigenschaften immer nur einen Wert angeben können, der dann als X- und Y-Wert verwendet wird. Mit der folgenden Angabe können Sie für Mozilla die Ausgabe aus Abbildung 8.10 erzeugen.

```
border:2px solid blue;-moz-border-radius-topleft:50px
```

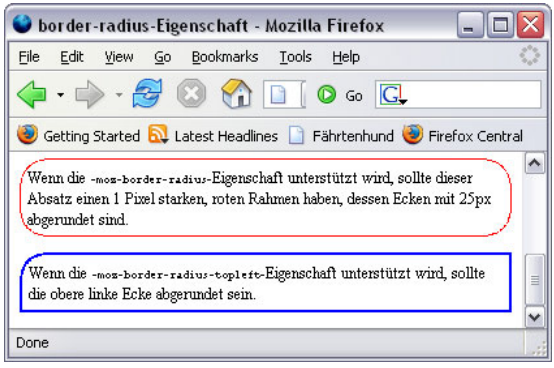


Abbildung 8.11: Darstellung der -moz-border-radius-Eigenschaft in Firefox

8.4 Praxisbeispiel

Rahmeneigenschaften eignen sich zusammen mit Außen- und Innenabständen und Hintergrundeigenschaften hervorragend dazu, Navigationsleisten zu erstellen oder einfache Hyperlinks mit Effekten zu versehen, die nicht an die langweilige Unterstreichung der Standard-Linkformatierung erinnern. Wie das mit wenig Aufwand geht, zeigt dieses Praxisbeispiel.

CD

Sie finden das Beispiel auf der Buch-CD in der Datei BSP/K08/praxisbeispiel.html.

Aufbau der Navigationsleisten

Das Beispiel definiert zunächst zwei `div`-Elemente, in denen die Navigationslinks als ganz normale Links angeordnet werden.

Listing 8.5: Aufbau der Seite

```
<h1>Kapitel 8: Praxisbeispiel</h1>
<div class="navigation1">
    <a href="#">Kontakt</a> <a href="#">Impressum</a>
    <a href="#">Produkte</a> <a href="#">Service</a> <a href="#">Home</a>
</div>
<div class="navigation2">
    <a href="#">Kontakt</a> <a href="#">Impressum</a>
    <a href="#">Produkte</a> <a href="#">Service</a> <a href="#">Home</a>
</div>
```

Navigationsleiste formatieren

Beide Navigationsleisten sollen eine graue Hintergrundfarbe und einen Rahmen bekommen. Dazu erstellen Sie einfach für beide CSS-Klassen einen Klassenstil:

```
.navigation1, .navigation2 {
   padding:10px;
   margin:10px;
   color: #666;
   background-color: #ccc;
   border:1px solid gray
}
```

Link-Unterstreichung löschen

Um die Link-Unterstreichung zu entfernen, setzen Sie im Stil für das `a`-Element die Eigenschaft `text-decoration` auf `none`. Wenn Sie möchten, können Sie mit `font-weight:bold` die Links auch fett formatieren.

```
a { text-decoration: none; font-weight:bold }
```

Links gepunktet unterstreichen

Statt der Standardunterstreichung können Sie nun die Links der ersten Navigationsleiste mit gepunkteten Linien unterstreichen und die Links weiß einfärben. Für den Rahmen legen Sie die Eigenschaft `border-bottom` auf `1px dotted white` fest.

```
.navigation1 a { border-bottom:1px dotted white;
                 color:white; padding-right:10px}
```

Hovereffekt erzeugen

Wenn Sie nun einen Hover-Effekt erzeugen möchten, legen Sie die abweichenden Formatierungen in einem Stil für das Pseudo-Element `a.hover` fest:

```
.navigation1 a:hover {
     border-bottom:1px solid orange;
     color:orange
}
```

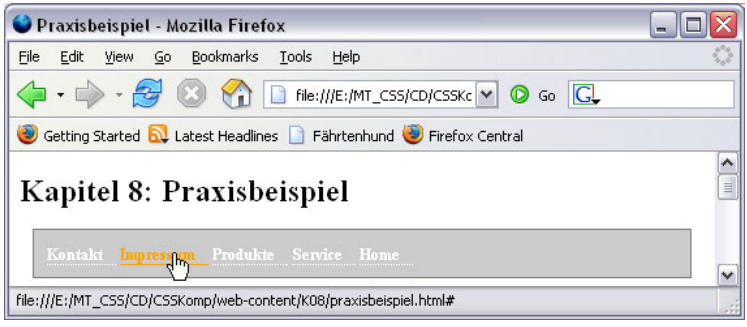


Abbildung 8.12: Das Ergebnis: die Formatierungen für die erste Navigationsleiste

Die zweite Navigationsleiste

Die zweite Navigationsleiste soll die Links senkrecht untereinander anzeigen. Daher sollten Sie zunächst die Breite der Navigationsleiste mit der `width`-Eigenschaft definieren:

```
.navigation2 { width:130px }
```

3D-Effekte erzeugen

Für die zweite Navigationsleiste sollen die Links so formatiert werden, dass Buttons mit 3D-Effekt entstehen. Dazu müssen Sie die Links zunächst mit `display:block` als Blockelemente anzeigen lassen.

Damit die Links auch wie Buttons aussehen, sollten Sie einen Innenabstand von ca. 5 Pixel definieren und die Breite so festlegen, dass die Buttons die Navigationsleiste ausfüllen. Darüber hinaus müssen Sie einen Rahmen definieren. Der ist für den 3D-Effekt erforderlich. Damit die Buttons erhaben aussehen, müssen Sie den linken und oberen Rahmen heller als den Hinter-

grund und die rechte und untere Rahmenlinie dunkler als den Hintergrund formatieren.

```
.navigation2 a {
   display:block;
   padding:5px; width:120px;
   border-top:1px solid white;
   border-left:1px solid white;
   border-right:1px solid gray;
   border-bottom:1px solid gray;
   margin-top:7px;margin-bottom:7px
}
```

Wenn der Button beim Hovern eingedrückt aussehen soll, müssen Sie für den Stil `a:hover` die Farben für die Rahmenlinien genau umgekehrt definieren:

```
.navigation2 a:hover {
   border-bottom:1px solid white;
   border-right:1px solid white;
   border-left:1px solid gray;
   border-top:1px solid gray;
   margin-top:7px;margin-bottom:7px }
```

Abbildung 8.13: Der erzeugte 3D- und Hover-Effekt in der zweiten Navigationsleiste

9 Tabellen und Spalten

Tabellen sollten eigentlich zur Darstellung tabellarischer Daten verwendet werden, auch wenn das durch den Gebrauch von Layout-Tabellen häufig in den Hintergrund rückt. Gerade für die Sprachausgabe und die Darstellung von Tabellen in Textbrowsern wie Lynx ist nicht nur ein korrekter Aufbau der Tabelle wichtig, sondern auch deren Formatierung. Mit Hilfe von CSS sind Sie in der Lage, eine Tabelle so zu formatieren, dass sie auch für blinde und sehbehinderte Surfer verständlich ist, denen die Inhalte von einem Screenreader vorgelesen oder mit einer Braillezeile dargestellt werden.

Mit der Eigenschaft `speak-header` *können Sie festlegen, ob und wie Tabellenbeschriftungen ausgesprochen werden. Mehr zu dieser Eigenschaft finden Sie in Kapitel 10, »Spezielle Medien: Druck- und Sprachausgabe«.*

9.1 Einführung

Hinsichtlich Tabellen ist CSS sowohl für die visuelle Darstellung der Tabelle wichtig als auch für die akustische Ausgabe. Gerade bei der Sprachausgabe stellt sich das Problem, dass Zellen, die nur als Überschriften formatiert aber nicht als solche definiert (`th`-Element) wurden, nicht vor den Werten vorgelesen werden. Stattdessen werden dem Zuhörer zuerst die Überschriften und danach die Daten vorgelesen. Spätestens nach der dritten Zeile weiß der Zuhörer dann nicht mehr, welche Daten zu welcher Spalte gehören. Daher ist es ganz wichtig, die Tabelle mit Hilfe der vorhandenen (X)HTML-Elemente zu strukturieren und mit CSS zu formatieren.

Tabellen können über eine Tabellenunterschrift bzw. -überschrift verfügen, die Sie mit dem `caption`-Element definieren. Sie können aber auch jedes andere Element als Tabellenüberschrift formatieren, indem Sie die `display`-Eigenschaft verwenden.

Details zur `display`*-Eigenschaft finden Sie in Kapitel 5, »Größen, Abstände und Positionierung«.*

In diesem Kapitel geht es ausschließlich um die visuelle Ausgabe von Tabellen. CSS-Eigenschaften, die die Sprachausgabe von Tabellen betreffen, finden Sie in Kapitel 10, »Spezielle Medien: Druck- und Sprachausgabe«.

9.2 Praxistaugliche Attribute

Nachfolgend finden Sie CSS-Attribute, die Sie problemlos oder mit nur geringen Einschränkungen in der Praxis einsetzen können. Entweder ist die Browserunterstützung sehr gut oder aber ein Nichtausführen der CSS-Anweisung wirkt sich nicht so negativ aus, dass die Seite nicht mehr bedienbar ist.

Rahmenabstand (border-spacing)

Wenn Sie für Tabellen `border-collapse:separate` verwenden, können Sie mit der Eigenschaft `border-spacing` den Rahmenabstand der Zellen bestimmen. Der Standardwert ist 0, somit sollten Browser, die die `border-spacing`-Eigenschaft unterstützen, die Rahmen ohne Abstand anzeigen, wenn keine `border-spacing`-Eigenschaft angegeben wurde.

CSS-Element: `border-spacing`	*Mögliche Werte:* `inherit`, Abstandswert, *0*
CSS-Version: CSS 2, CSS 2.1, CSS 3	*Zulässig für folgende (X)HTML-Elemente:* alle Tabellenelemente, auch die mit `display:table` und `display:inline-table` formatierten
Medium: Visual	*Vererbt:* Ja

Palm	NN	Mozilla		Internet Explorer							Opera			Safari		iCab	Kq	FF
2.x	4.x	1.x	1.8	4.x	5.0x (Win)	5.0 (Mac)	5.1+ (Mac)	5.5 (Win)	6	7	6	7	8	1.x	2.0	2.9+	3.x	1.0
		○ 1+3	○ 1									○ 1	○ 1		○ 1	○ 2	○ 1	○ 1

1 Ohne Angabe der `border-spacing`-Eigenschaft wird der Standardwert nicht gesetzt.

2 In Version 2.9.x werden keine Zellrahmen angezeigt, daher funktioniert auch `border-spacing` nicht. Ab Version 3 wird ohne Angabe der `border-spacing`-Eigenschaft deren Standardwert nicht gesetzt.

3 ab Version 1.4+

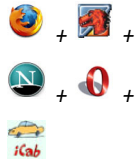
Als Abstandswert geben Sie einen oder zwei numerische Werte mit Einheit an. Der erste von zwei Werten bestimmt dabei den horizontalen Abstand, der zweite den vertikalen. Geben Sie nur einen Wert an, gilt der für beide Abstände.

INFO

Sie finden das Beispiel auf der Buch-CD in der Datei BSP/K09/border_spacing.html.

CD

Die auf Mozilla basierenden Browser, sowie der Opera-Browser und iCab 3.0 wenden den Standardwert nicht an. Geben Sie also keine `border-spacing`-Eigenschaft an, wird der Standardwert 0 nicht gesetzt.



Sie sollten daher den Wert 0 immer explizit setzen, wenn Sie keinen Abstand wünschen.

:-) TIPP

Rahmenverschmelzung (border-collapse)

Die `border-collapse`-Eigenschaft bestimmt, wie Zellrahmen dargestellt werden. Generell gibt es zwei Möglichkeiten. Jede Zelle hat einen eigenen Rahmen, benachbarte Zellen haben damit einen doppelt so dicken Rahmen. Das ist der Standard.

Da das nicht immer gewünscht ist, besteht mit der `border-collapse`-Eigenschaft die Möglichkeit festzulegen, dass Rahmen benachbarter Zellen verschmelzen und daher nur die einfache Stärke haben.

CSS-Element: `border-collapse`	*Mögliche Werte: collapse*[1], `separate`, `inherit`
CSS-Version: CSS 2, 2.1, CSS 3	*Zulässig für folgende (X)HTML-Elemente:* alle Tabellenelemente, auch die mit `display:table` und `display:inline-table` formatierten
Medium: Visual	*Vererbt:* Ja

Palm	NN	Mozilla		Internet Explorer							Opera			Safari		iCab	Kq	FF
2.x	4.x	1.x	1.8	4.x	5.0x (Win)	5.0 (Mac)	5.1+ (Mac)	5.5 (Win)	6	7	6	7	8	1.x	2.0	2.9+	3.x	1.0
● 3		● 2	●						● 3	● 3		●	●		●	● 4	●	●

1 in CSS 2.1 ist `separate` der Standardwert
2 ab Version 1.4+.
3 siehe unten
4 Bis zur Version 2.9.x werden keine Zellrahmen dargestellt.

Mögliche Werte

- Der Wert `collapse` bewirkt, dass Rahmen benachbarter Zellen zusammengelegt werden.
- Der Wert `separate` zeichnet für alle Zellen einen separaten Rahmen und entspricht damit dem, was Browser anzeigen, die die `border-collapse`-Eigenschaft nicht unterstützten.

INFO

Der CSS-Standard sieht beim Wert `collapse` *vor, dass die Rahmen der Zellen, die die Rahmenkanten der Tabelle berühren, zugunsten des Tabellenrahmens entfallen, wenn der Tabellenrahmen sich in mehr als nur der Farbe von den Zellrahmen unterscheidet. Sind die Rahmen bis auf die Rahmenfarbe gleich, gehen die Rahmen der Zellen denen der Tabelle vor. Wurde also ein Tabellenrahmen definiert, muss der auf jeden Fall angezeigt werden, auch wenn* `border-collapse:collapse` *definiert wurde.*

CD

Sie finden das Beispiel auf der Buch-CD in der Datei BSP/K09/border_collapse.html.

Im Beispiel wurde für alle Tabellen ein roter Rahmen (`3px`) definiert und für die Zellen ein blauer (`2px`). Ein korrekt arbeitender Browser sollte also den Tabellenrahmen zeichnen und für die Zellen nur die Rahmen, die nicht die Tabellenrahmenkanten berühren.

```
table  { border:3px solid red; border-spacing:0 }
td, th { border:2px solid blue; margin:0 }
```

Die Testtabelle sieht wie folgt aus und enthält daher unten rechts eine verbundene Zelle.

Listing 9.1: Aufbau der Vergleichstabelle

```
<table>
    <caption>Dies ist die Vergleichstabelle ohne <code>border-
    collapse</code>.</caption>
    <tr><th>Überschrift 1</th><th>Überschrift 2</th><th>Überschrift
    3</th></tr>
    <tr><td>Zelle 1</td><td>Zelle 2</td><td rowspan="2">Zelle 3</td></tr>
    <tr><td>Zelle 4</td><td>Zelle 5</td></tr>
</table>
```

Listing 9.2: Die Tabellen mit der collapse-Eigenschaft

```
    <table style="border-collapse:collapse">
        <caption>Diese Tabelle wurde mit <code>border-
        collapse:collapse</code> formatiert.</caption>
        <tr><th>Überschrift 1</th><th>Überschrift
            2</th><th>Überschrift 3</th></tr>
            <tr><td>Zelle 1</td><td>Zelle 2</td><td
            rowspan="2">Zelle 3</td>
        </tr>
        <tr><td>Zelle 4</td><td>Zelle 5</td></tr>
    </table>
    <p></p>
    <table style="border-collapse:separate">
        <caption>Diese Tabelle wurde mit <code>border-
        collapse:collapse</code> formatiert.</caption>
        <tr><th>Überschrift 1</th><th>Überschrift
            2</th><th>Überschrift 3</th></tr>
            <tr><td>Zelle 1</td><td>Zelle 2</td><td
            rowspan="2">Zelle 3</td>
        </tr>
        <tr><td>Zelle 4</td><td>Zelle 5</td></tr>
    </table>
```

Der Internet Explorer 6/7 und der Palm-Browser stellen auch bei der Verwendung von `separate` den Zellabstand dar, auch wenn Sie keinen mit dem `cellspacing`-Attribut angegeben haben. Das liegt daran, dass die Browser den Standardwert 0 für die `border-spacing`-Eigenschaft nicht unterstützen.

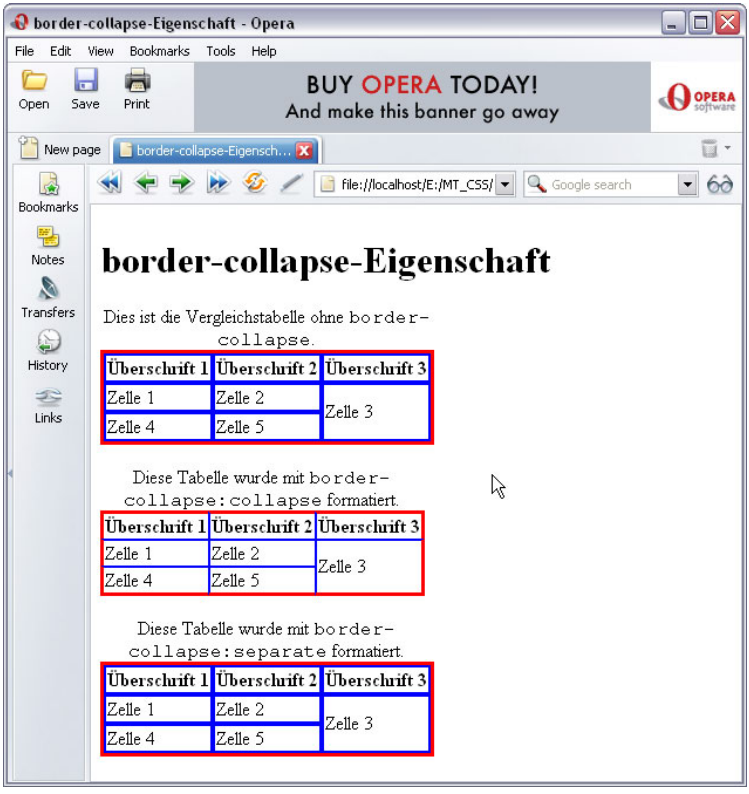


Abbildung 9.1: Korrekte Darstellung der Seite im Opera-Browser; die erste und dritte Tabelle werden gleich dargestellt.

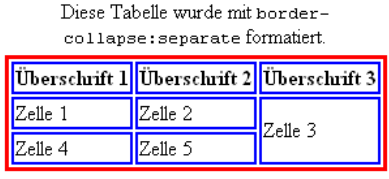


Abbildung 9.2: Fehlerhafte Darstellung des Internet Explorers

:-) TIPP

Sie können das Problem lösen, indem Sie die Tabelle mit `<table cellspacing="0">` *definieren und damit den Zellabstand per HTML auf 0 setzen.*

Tabellenlayout (table-layout)

Die Eigenschaft `table-layout` definiert, wie die Größe einer Tabelle berechnet wird.

CSS-Element: `table-layout`	*Mögliche Werte: auto*, `fixed`, `inherit`
CSS-Version: CSS 2, 2.1, 3	*Zulässig für folgende (X)HTML-Elemente:* Tabellen, Inline-Tabellen und alle Elemente, die mit der `display`-Eigenschaft als (Inline)-Tabellen ausgegeben werden
Medium: Visual	*Vererbt:* Nein

Palm	NN	Mozilla		Internet Explorer							Opera			Safari		iCab	Kq	FF
2.x	4.x	1.x	1.8	4.x	5.0x (Win)	5.0 (Mac)	5.1+ (Mac)	5.5 (Win)	6	7	6	7	8	1.x	2.0	2.9+	3.x	1.0
		● 1	● 1			● 1	● 1	● 2	● 2	● 2		● 1	● 1		● 1	● 1	● 1	● 1

1 Der Browser stellt `table-layout:fixed` mit Spaltenbreiten, die erst in der zweiten Zeile oder später spezifiziert wurden, so dar, als ob die Spaltenbreiten in der ersten Zeile gültig angegeben wären.

2 Der Browser stellt `table-layout:fixed` mit Spaltenbreiten, die erst in der zweiten Zeile oder später spezifiziert wurden, so dar, als ob keine Spalten- und keine Tabellenbreite angegeben wären. Die Tabellenbreite wird auf 100% gesetzt.

Der Wert `auto` *bewirkt, dass ein automatisches Tabellenlayout verwendet wird, beim Wert* `fixed` *wird ein festes Tabellenlayout verwendet. Ein festes Tabellenlayout bewirkt, dass die Breite einer Tabelle durch die angegebenen Werte für die Tabellenbreite und die Spaltenbreiten (inklusive den Werten für Innen- und Außenabstände) ermittelt wird. Der Inhalt wird bei der Berechnung der Spalten- und der Tabellenbreite nur dann berücksichtigt, wenn für eine oder mehrere Spalten oder die Tabelle selbst keine Breite oder* `width:auto` *angegeben wurde.*

Gemäß CSS-Standard werden für die Berechnung der Tabellenbreite beim festen Tabellenlayout nur Spaltenbreiten berücksichtig, die für ein `th`*-Element oder ein* `td`*-Element in der ersten Tabellenzeile angegeben wurden. Dadurch kann eine Tabelle schon formatiert werden, wenn die erste Datenzeile eingelesen wurde.*

Automatisches Tabellen-Layout

Beim automatischen Tabellenlayout erfolgt die Berechnung der Tabellen- und Spaltenbreiten nach Wahl des Browsers. Der CSS-Standard schreibt nicht vor, wie die Berechnung erfolgen soll. Die meisten Browser verfahren dabei aber wie folgt:

Zunächst werden die Spaltenbreiten berechnet. Für jede Spalte wird dabei die Mindestbreite berechnet, die erforderlich ist, um die Inhalte darzustellen. Wurde eine Breite für die Spalte angegeben, die größer ist als die Mindestbreite für den Inhalt, wird die angegebene Breite (`width`-Eigenschaft) als Mindestbreite verwendet. Die ermittelten Spaltenbreiten werden nun zusammen mit Außenabständen, Innenabständen und Rahmenbreiten addiert und ergeben die Mindesttabellenbreite.

Wurde für die Tabelle eine Breite definiert und ist diese kleiner als die Mindesttabellenbreite, wird die Mindesttabellenbreite verwendet. Falls der Wert größer ist, wird der verbleibende Platz zwischen den Spalten aufgeteilt.

Prozentuale Werte für Spaltenbreiten werden relativ zur Tabellenbreite definiert. Sie sollten vom Browser eingehalten werden, sofern dies möglich ist. Das heißt, der berechnete tatsächliche Wert der Spaltenbreite muss größer als die für den Inhalt ermittelte Mindestbreite und kleiner als die Tabellenbreite sein.

Aufgrund dieser vielen Berechnungen und eventuellen Korrekturen ist die Verwendung des automatischen Layouts vor allem bei großen und komplexen Tabellen sehr langsam. Sie sollten daher Tabellen möglichst mit festem Tabellenlayout definieren.

Sie finden das Beispiel auf der Buch-CD in der Datei BSP/K09/table_layout.html.

Tabellentitelposition (caption-side)

Die Eigenschaft `caption-side` bestimmt, wo die Tabellenüberschrift angezeigt werden soll.

CSS-Element: `caption-side`	*Mögliche Werte: top*, `bottom`, `left`[1] `right`[1], `inherit`
CSS-Version: CSS 2, 2.1, 3, TV	*Zulässig für folgende (X)HTML-Elemente:* `caption`-Elemente und alle Elemente, die mit `display:table-caption` formatiert sind
Medium: Visual	*Vererbt:* Ja

Palm	NN	Mozilla		Internet Explorer							Opera			Safari		iCab	Kq	FF
2.x	4.x	1.x	1.8	4.x	5.0x (Win)	5.0 (Mac)	5.1+ (Mac)	5.5 (Win)	6	7	6	7	8	1.x	2.0	2.9+	3.x	1.0
○[2]		○[2]	●[3]			○[2]	○[2]				○[2]	○[2]	○[2]		○[2]	○[2]	○[2]	●

1 nicht in CSS 2.1
2 nur die Werte `top` und `bottom`
3 ab Version 1.4+

Mögliche Werte

- `top`, positioniert die Überschrift oberhalb der Tabelle
- `bottom`, zeigt die Überschrift unterhalb der Tabelle an
- `left`, positioniert den Tabellentitel links von der Tabelle
- `right`, zeigt die Überschrift rechts neben der Tabelle an

Generell gilt, dass die Überschrift in einer eigenen Box angezeigt wird. Die horizontale Ausrichtung des Textes innerhalb der Überschriftenbox legen Sie mit der `text-align`-Eigenschaft fest. Für die vertikale Ausrichtung innerhalb der Überschriftenbox verwenden Sie die `vertical-align`-Eigenschaft. Dabei werden alle Werte außer `middle` und `bottom` wie `top` behandelt.

Sie finden ein Beispiel auf der Buch-CD in der Datei BSP/K09/caption_side.html.

Mehr zu den Eigenschaften `text-align` *und* `vertical-align` *finden Sie in Kapitel 4, »Textformatierungen«.*

9.3 Nicht/schlecht unterstützte Attribute

In dieser Rubrik finden Sie solche Attribute, die von vielen oder zumindest von sehr wichtigen Browser nicht oder fehlerhaft unterstützt werden und die somit noch nicht praxistauglich sind.

Leere Zellen (empty-cells)

Wenn Sie mit `border-collapse:separate` definiert haben, dass einzelne Rahmen um alle Zellen gezeichnet werden, bestimmt die `empty-cells`-Eigenschaft, ob auch Zellen mit einem Rahmen umgeben werden, die keinen sichtbaren Inhalt haben. Das sind Zellen, deren Inhalt mit `display:none` oder `visibility:hidden` ausgeblendet wurde, bzw., die keinen Inhalt außer Leerzeichen, Wagenrückläufe und Tabulatoren enthalten.

Das feste Leerzeichen ` ` *gilt als sichtbarer Inhalt.*

Mehr Informationen zur Eigenschaft `border-collapse` *finden Sie im Abschnitt »Rahmenverschmelzung (border-collapse)«.*

CSS-Element: `empty-cells`	*Mögliche Werte: show*, `hide`, `inherit`
CSS-Version: CSS 2 - 3	*Zulässig für folgende (X)HTML-Elemente:* alle Elemente, die Tabellenzellen darstellen bzw. Tabellenzellen gruppieren
Medium: Visual	*Vererbt:* Ja

Palm	NN	Mozilla		Internet Explorer							Opera			Safari		iCab	Kq	FF
2.x	4.x	1.x	1.8	4.x	5.0x (Win)	5.0 (Mac)	5.1+ (Mac)	5.5 (Win)	6	7	6	7	8	1.x	2.0	2.9	3.x	1.0
5	7	● 2	● 2				⊙ 1+4		6	6	● 1	● 1	● 1	● 2	● 2	● 3	● 2	● 2

1 Als Standardwert wird `hide` verwendet.
2 Beim Wert `hide` wird auch die Hintergrundfarbe nicht angezeigt.
3 Es werden bis zur Version 2.9.x keine Zellrahmen angezeigt, ab Version 3.0 fehlerfrei.
4 fehlerhaft in Verbindung mit `cellspacing` > 0
5 Es wird immer ein Rahmen um leere Zellen angezeigt.
6 Es wird nie ein Rahmen um leere Zellen angezeigt.
7 Leere Zellen werden gar nicht angezeigt.

Zwar spricht der CSS-Standard davon, dass die Eigenschaft bestimmt, ob um leere Zellen ein Rahmen gezeichnet werden soll, faktisch wirkt sich jedoch die Eigenschaft bei vielen Browsern, die die Eigenschaft unterstützen, dahingehend aus, dass eine leere Zelle beim Wert `hide` *gar nicht angezeigt wird. Das heißt, auch eine Hintergrundfarbe oder ein Hintergrundbild ist nicht sichtbar.*

Mögliche Werte

- `hide` sorgt dafür, dass keine Rahmen um Zellen gezogen werden, die keine sichtbaren Inhalte haben.
- `show` bestimmt, dass auch Zellen ohne sichtbaren Inhalt angezeigt werden und einen Rahmen erhalten (falls einer definiert ist).

Sie finden das Beispiel auf der Buch-CD in der Datei BSP/K09/emtpy_cells.html.

Beispiel

Wenn Sie eine Tabelle wie folgt definieren, sollte auch um die letzte Zelle, die keinen sichtbaren Inhalt enthält, ein Rahmen gezeichnet werden.

Listing 9.3:
Aufbau der Tabelle

```
<table style="border-collapse:separate;
    empty-cells:show">
    <caption>Dies ist die Tabelle mit <code>empty-cells:show</code>.
    Alle Zellen sollten mit Rahmen und Hintergrund versehen
    werden.</caption>
    <tr><th>volle Zellen</th><th>Zellen mit festem Leerzeichen
    </th><th>Leere Zelle</th></tr>
    <tr><td>Zelle 1</td><td> </td><td rowspan="2"></td></tr>
    <tr><td>Zelle 4</td><td> </td></tr>
</table>
```

Allerdings setzt das voraus, dass Sie auch Rahmen für die Tabellenzellen definiert haben, beispielsweise:

```
table  { border:1px solid black; }
td, th { border:1px solid blue;
         background-color:rgb(255,200,200) }
```

Der Internet Explorer 5.2 für Macintosh stellt den Rahmen verbundener, leerer Zellen falsch dar, wenn Sie `emtpy-cells:show` zusammen mit `border-collapse:separate` und einem Rahmenabstand (`border-spacing`) oder Zellabstand (`cellspacing`-Attribut) größer als 0 verwenden. In diesem Fall wird ein zu dicker Rahmen um die leere Zelle gezeichnet, der aber nicht direkt um die Zelle sondern um den angegebenen Zellabstand gezeichnet wird.

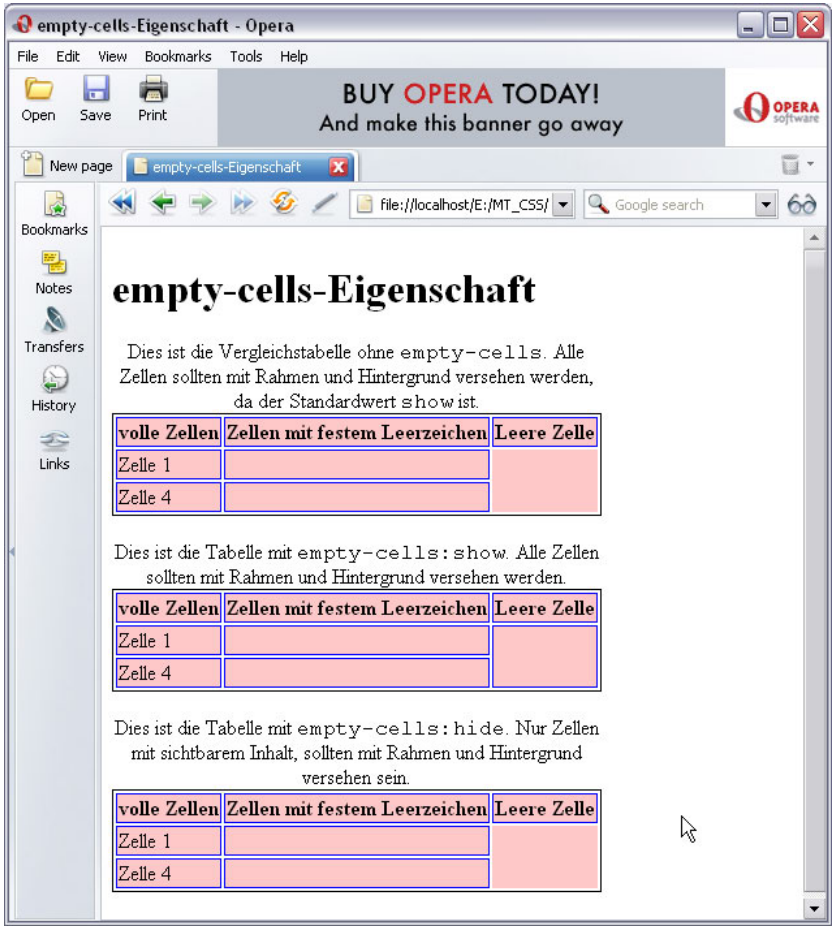


Abbildung 9.3: Korrekte Darstellung der Beispielseite im Opera-Browser

Win + +

Der Internet Explorer 6/7, Netscape Navigator und der Palm-Browser unterstützen die Eigenschaft alle nicht. Sie verhalten sich jedoch unterschiedlich, was die Anzeige leerer Zellen angeht. Der Palm-Browser zeigt auch leere Zellen immer mit dem definierten Rahmen und mit Hintergrund an. Der Internet Explorer zeigt zwar das Hintergrundbild an, nicht jedoch den Rahmen. Der Netscape Navigator 4.x zeigt leere Zellen gar nicht an, weder mit Rahmen noch mit Hintergrund.

Browserunterstützung

Aufgrund der schlechten Browserunterstützung und der unterschiedlichen Ergebnisse bei den Browsern, die die Eigenschaft nicht unterstützen, sollten Sie die Eigenschaft derzeit noch nicht einsetzen.

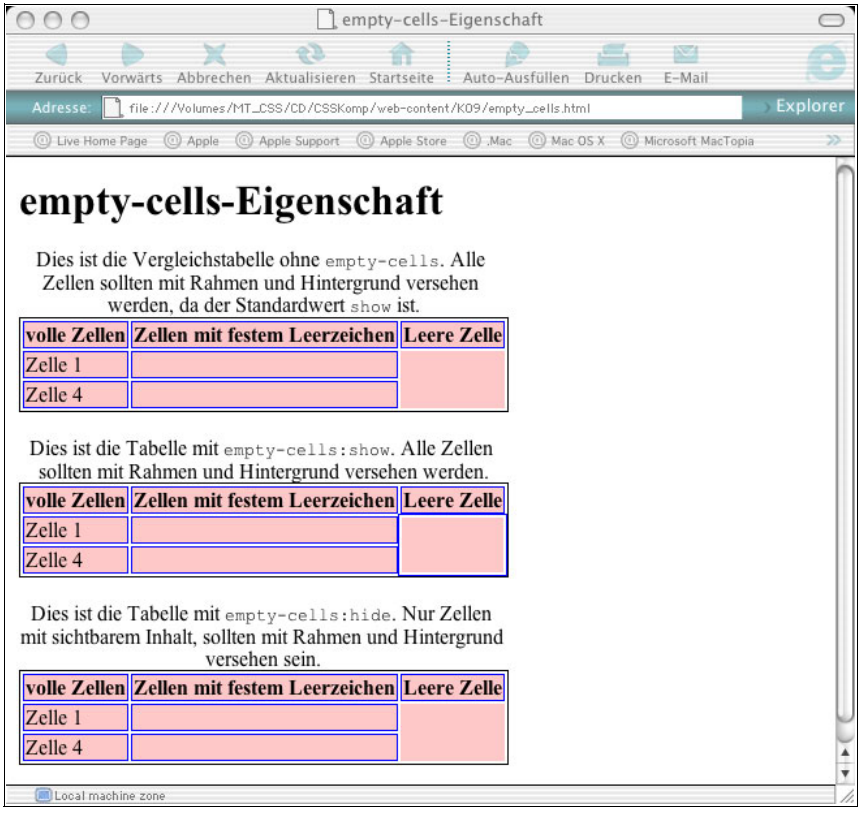


Abbildung 9.4: Fehlerhafte Darstellung des Internet Explorers für Macintosh

Sorgen Sie am besten dafür, dass es keine wirklich leeren Zellen gibt, indem Sie ein festes Leerzeichen ` ` *oder eine transparentes Bild einfügen.*

:-) TIPP

Textausrichtung (text-align)

Um Inhalte von Zellen horizontal auszurichten, verwenden Sie die `text-align`-Eigenschaft. Sie können die Eigenschaft nicht nur auf einzelne Zellen anwenden, sondern auch auf mehrere Zellen, beispielsweise auf eine ganze Zeile oder Spalte. In diesem Fall werden die Inhalte an einer vertikalen Linie ausgerichtet, die durch alle diese Zellen geht.

Falls der Zellinhalt über mehr als eine Zeile umbrochen wird, ist das Ergebnis nicht definiert. Daher sollten Sie für solche Zellen einen Zeilenumbruch verhindern, indem Sie die Inhalte mit Hilfe einer Zeichenkette ausrichten.

Die nachfolgenden Kompatibilitätsangaben beziehen sich ausschließlich auf die Verwendung innerhalb von Tabellen und Tabellenzellen.

CSS-Element: `text-align`	*Mögliche Werte:* `left` (linksbündig), `right` (rechtsbündig), `center` (zentriert), `justify` (Blocksatz), Zeichenkette[1], `inherit`
CSS-Version: CSS 1, CSS 2, CSS 2.1, CSS 3, TV, Mobile	*Zulässig für folgende (X)HTML-Elemente:* alle Blockelemente
Medium: Visual	*Vererbt:* Ja

Palm	NN	Mozilla		Internet Explorer							Opera			Safari		iCab	Kq	FF
2.x	4.x	1.x	1.8	4.x	5.0x (Win)	5.0 (Mac)	5.1+ (Mac)	5.5 (Win)	6	7	6	7	8	1.x	2.0	2.9+	3.x	1.0
	O 2+3	O 3	O 4	O 3+4	O 3	O 3+2	O 3+2	O 3+4	O 3+4	O 5	O 3	O 3	O 3	O 3	O 3	O 3+2	O 3	O 3

1 nicht in CSS 2.1, TV und Mobile-Spezifikation
2 kein Blocksatz
3 Keine Unterstützung, wenn die Eigenschaft auf ein `col`-Element angewendet wird; bei Anwendung auf `td`, keine Verwendung der Zeichenketten.
4 nur `center`, bei Anwendung auf `col`-Elemente
5 Auf `col`-Elemente werden `left`, `center` und `right` angewendet, keine Anwendung von Zeichenketten, keine Unterstützung von `text-align:justify`.

Weitere Informationen zu den Schlüsselwörtern finden Sie in Kapitel 4, »Textformatierungen«, wo diese Eigenschaft in Bezug auf normalen Fließtext schon behandelt wurde.

Beispiel

Mithilfe einer Zeichenkette als Wert können Sie beispielsweise alle Werte einer Tabellenspalte am Dezimaltrennzeichen oder an einer Währungsbezeichnung ausrichten. Sie geben dazu z.B. das Dezimaltrennzeichen als Zeichenkette an:

```
.dezimal { text-align:"," }
```

Dann sollte der Browser alle Zellen (die natürlich als Spalte angeordnet sein sollten) so ausrichten, dass das Komma untereinander steht. Das Komma bildet dann eine virtuelle vertikale Achse, an der die Inhalte ausgerichtet werden.

Werte, die die angegebene Zeichenkette nicht enthalten, werden links von dieser Achse angeordnet.

Sie finden das Beispiel auf der Buch-CD in der Datei BSP/K09/text_align.html.

Browserunterstützung

Sie können die Eigenschaft mithilfe einer CSS-Klasse oder einem `style`-Attribut sowohl auf Tabellenzellen (`td`-Elemente) wie auch auf ganze Spalten (`col`-Elemente) anwenden. Bis auf den Internet Explorer 6, der den Wert `center` auf `col`-Elemente anwendet, und den Internet Explorer 7, der auch die Werte `left` und `right` auf `col`-Elemente anwendet, wendet kein Browser irgendeinen Wert auf Tabellenspalten an, immer nur auf `td`-Elemente.

10 Spezielle Medien: Druck- und Sprachausgabe

Heute wird CSS hauptsächlich für die Ausgabe am Bildschirm eingesetzt. Gerade einmal die Druckausgabe wird von einigen Webautoren genutzt. Sprachausgabe und Stylesheets für Handhelds und andere Geräte werden eher selten eingesetzt. Die Elemente für die Sprachausgabe von Webseiten werden genauso wenig verwendet, was jedoch daran liegt, dass derzeit kein Browser die CSS-Befehle für die Sprachausgabe unterstützt.

iCab unterstützt die Sprachausgabe von Webseiten bereits in der Version 2.9.x. Die Qualität der Ausgabe ist aber gerade bei deutschen Seiten sehr dürftig, da der Sprecher trotz Angabe des `lang`-Attributs im `body`-Element versucht, die Wörter englisch auszusprechen. Zudem werden die CSS-Befehle zur Sprachausgabe nicht berücksichtigt. Dennoch hat iCab damit einen Anfang gemacht. Gerade wegen dieser Fähigkeiten muss mit dem zunehmenden Einsatz von iCab gerechnet werden, weshalb sie ihn unbedingt auch für die Bildschirmausgabe ihrer Seite berücksichtigen sollten.

In diesem Kapitel geht es um die CSS-Eigenschaften, die Sie für seitenorientierte Medien (Paged Media) zur Verfügung haben, sowie um die Befehle zur Steuerung der Sprachausgabe.

Wenn Sie Stylesheets für bestimmte Medien, wie die Druck- oder Sprachausgabe, erzeugen möchten, müssen Sie dazu das Ausgabemedium angeben. Wie das geht wird in Kapitel 3, »Browseroptimierung«, beschrieben.

10.1 Einführung in seitenorientierte Medien

Einige der neueren Browser bieten schon eine recht gute Unterstützung für die Befehle zur Ausgabe auf seitenorientierten Medien. Diese Befehle ermöglichen es Ihnen beispielsweise, an geeigneten Stellen in der Seite einen Seitenumbruch zu erzeugen oder die Größe der Seite zu bestimmen.

Kontrollieren lässt sich das jedoch nur, indem Sie die Beispiele wirklich ausdrucken oder sich an der Druckvorschau des Browsers orientieren. Hat der Browser keine Druckvorschau, bietet sich die Möglichkeit an, die Seite in eine PDF-Datei auszugeben.

Auf dem Macintosh geht das ohne weitere Software, da der Standard-Drucker-Dialog die Möglichkeit bietet, eine PDF-Datei zu erzeugen.

Unter Windows benötigen Sie dazu entweder Adobe® Acrobat, was recht teuer ist, oder ein Freeware- oder Shareware-Tool, wie beispielsweise FreePDF. Wichtig ist, dass Sie ein Tool verwenden, dass Sie wie einen Druckertreiber verwenden können, damit der Browser auch das Stylesheet für die Druckausgabe verwendet.

Für seitenorientierte Medien erweitert CSS das Boxmodell um die Seitenbox und das Seitenmodell. Diese ermöglichen es Ihnen, Seitengröße und -ränder anzugeben und Seitenumbrüche zu bestimmen.

Das Seitenmodell

Das Seitenmodell von CSS bestimmt, wie ein Dokument innerhalb der Seitenbox formatiert wird.

Die Seitenbox

Die Seitenbox bestimmt, welche Größe eine Seite hat; sie muss aber nicht der Seitengröße entsprechen, in der das Dokument später ausgegeben wird. Die Seitenbox ist damit analog zur Seite in einem Textverarbeitungsprogramm zu verstehen. Auch dort definieren die Seiteneinstellungen, wie groß die Seite ist, welche Kantenlängen und Abstände es gibt. Aber ob der Drucker für die spätere Ausgabe dies einhalten kann oder ob dann eine A4-Seite nicht auf A3 gedruckt wird, lässt sich im Vorhinein nicht bestimmen. Daher hat der Browser die Aufgabe, die Seitenbox möglichst optimal auf das Ausgabemedium zu übertragen. Wie das funktioniert, legt der CSS-Standard nicht fest.

Die Seitenbox besteht aus dem Seitenbereich und dem Randbereich. Dieser umgibt den Seitenbereich. Mit dem Randbereich können Sie den Innenabstand der Seite bestimmen. Dies ist die einzige Möglichkeit, da die `border`- und `padding`-Eigenschaften nicht auf Seiten angewendet werden.

@page-Regel

Die Seitenbox definieren Sie in CSS mit der `@page`-Regel. Ihre Syntax lautet:

```
@page Seiten-Selektor:Pseudo-Klasse {...}
```

Als Seiten-Selektor geben Sie einen Selektor an, der bestimmt, welche Seiten formatiert sind. Im Zweifelsfall ist dies der Universal-Selektor *, den Sie auch weglassen können. Auch die Pseudo-Klasse ist optional, mit ihr können Sie bestimmen, ob die Formatierung nur für die erste (`:first`), alle linken (`:left`) oder alle rechten (`:right`) Seiten gelten soll. Wenn Sie die Formatierungen für alle Seiten festlegen möchten, lassen Sie die Pseudo-Klasse weg. Die kürzeste Version der `@page`-Regel lautet also:

```
@page {}
```

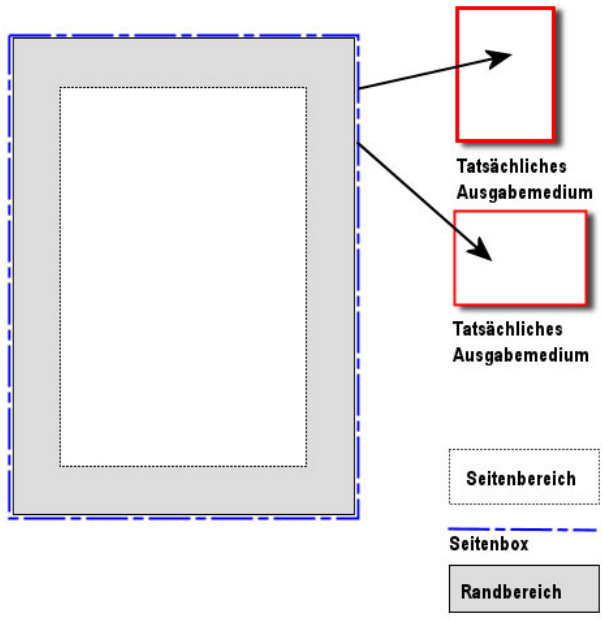


Abbildung 10.1: Aufbau des Seitenmodells von CSS 2.0

Möchten Sie nur die Formatierung für die erste Seite festlegen, geben Sie

```
@page :first{}
```

an.

Seitengröße

Die Seitengröße (siehe *Abschnitt »Seitenrand (margin)«*) können Sie nicht direkt festlegen, dafür aber die Größe der Seitenbox, nämlich mit der `size`-Eigenschaft. Die Größe des Seitenbereichs ergibt sich dann aus der Größe der Seitenbox abzüglich der Größe des Randbereichs.

Der Randbereich

Den Randbereich (siehe *Abschnitt »Seitenrand (margin)«*) definieren Sie über die Eigenschaften `margin`, `margin-top`, `margin-bottom` und `margin-left`. Dabei werden prozentuale Werte relativ zur Größe der Seitenbox berechnet und beziehen sich bei den horizontalen Rändern auf die Breite der Seitenbox und bei den vertikalen Rändern (`margin-top` und `margin-bottom`) auf die Höhe der Seitenbox.

!! STOP

Die Einheiten `em` *und* `ex` *sind für die Festlegung der Seitenränder nicht erlaubt. Erlaubt sind hingegen negative Randwerte und auch Positionierungen außerhalb des Seitenbereichs. Wie mit den Inhalten außerhalb des Seitenbereichs verfahren wird, hängt jedoch vom Browser, Drucker und weiteren an der Ausgabe beteiligten Geräten und Programmen ab.*

Die derzeit verfügbare iCab-Version 3.0 BETA erzeugt in vielen Fällen beim Einsatz der CSS-Befehle für die Druckausgabe komplett unleserliche Druckausgaben und Anzeigen in der Druckvorschau. Daher lässt sich bei vielen Befehlen nicht sagen, ob sie unterstützt werden. Da aber selbst einfache Befehle wie die für den Seitenumbruch nicht unterstützt werden, kann davon ausgegangen werden, dass auch der Rest noch nicht verfügbar ist.

10.2 Praxistaugliche Attribute

Nachfolgend finden Sie CSS-Attribute, die Sie problemlos oder mit nur geringen Einschränkungen in der Praxis einsetzen können. Entweder ist die Browserunterstützung sehr gut oder aber ein Nichtausführen der CSS-Anweisung wirkt sich nicht so negativ aus, dass die Seite nicht mehr bedienbar ist.

Seitenumbruch (page-break-before, page-break-after, page-break-inside)

Mit den Eigenschaften `page-break-before`, `page-break-after` und `page-break-inside` können Sie festlegen ob vor, nach oder innerhalb eines Elements ein Seitenumbruch erfolgen soll oder nicht.

Generell gilt, dass durch ein Seitenumbruch die aktuelle Seitenbox beendet und für den Rest des Dokuments eine neue Seitenbox erzeugt wird.

CSS-Element: `page-break-before`	*Mögliche Werte: auto*, `always`, `avoid`, `left`, `right`, `inherit`
CSS-Version: CSS 2, CSS 3	*Zulässig für folgende (X)HTML-Elemente:* alle Blockelemente
Medium: seitenorientierte Medien	*Vererbt:* Nein

Palm	NN	Mozilla		Internet Explorer							Opera			Safari		iCab	Kq	FF
2.x	4.x	1.x	1.8	4.x	5.0x (Win)	5.0 (Mac)	5.1+ (Mac)	5.5 (Win)	6	7	6	7	8	1.x	2.0	2.9+	3.x	1.0
[1]			●				●		●	●	●	●	●		●		●	●

[1] keine Druckausgabe möglich

CSS-Element: `page-break-after`	*Mögliche Werte: auto*, `always`, `avoid`, `left`, `right`, `inherit`
CSS-Version: CSS 2, CSS 3	*Zulässig für folgende (X)HTML-Elemente:* alle Blockelemente
Medium: seitenorientierte Medien	*Vererbt:* Nein

Palm	NN	Mozilla		Internet Explorer							Opera			Safari		iCab	Kq	FF
2.x	4.x	1.x	1.8	4.x	5.0x (Win)	5.0 (Mac)	5.1+ (Mac)	5.5 (Win)	6	7	6	7	8	1.x	2.0	2.9+	3.x	1.0
[1]			●				●		●	●	●	●	●		●		●	●

[1] keine Druckausgabe möglich

Die `page-break-inside`-Eigenschaft legt fest, ob innerhalb eines Elements überhaupt ein Seitenumbruch erfolgen darf. Sie können innerhalb des Elements keinen Umbruch erzwingen, wohl aber mit dem Wert `avoid` vermeiden, dass innerhalb eines Absatzes ein Umbruch erfolgt.

CSS-Element: `page-break-inside`	*Mögliche Werte: auto*, `avoid`, `inherit`
CSS-Version: CSS 2, CSS 3	*Zulässig für folgende (X)HTML-Elemente:* alle Blockelemente
Medium: seitenorientierte Medien	*Vererbt:* Ja

Palm	NN	Mozilla		Internet Explorer							Opera			Safari		iCab	Kq	FF
2.x	4.x	1.x	1.8	4.x	5.0x (Win)	5.0 (Mac)	5.1+ (Mac)	5.5 (Win)	6	7	6	7	8	1.x	2.0	2.9+	3.x	1.0
[1]							●		●	●	●	●	●					●

[1] keine Druckausgabe möglich

Bedeutung der möglichen Werte

- `auto`: Der Seitenumbruch wird automatisch vom ausgebenden Gerät gesteuert. Er wird nicht unterdrückt, wird aber auch nicht erzwungen.
- `always`: Vor bzw. hinter der Box des Blockelements wird ein Seitenumbruch eingefügt.
- `avoid`: verhindert einen Seitenumbruch vor (hinter, in) dem Element.

- `left`: Vor bzw. hinter der Box werden mindestens ein, maximal zwei Seitenumbrüche eingefügt, so dass die nächste Seite eine linke Seite ist.
- `right`: Vor bzw. hinter der Box werden mindestens ein, maximal zwei Seitenumbrüche eingefügt, so dass die nächste Seite eine rechte Seite ist.

Problemlösung

Zusammen mit den Eigenschaften `page-break-inside` und `page-break-after` können Sie Einstellungen festlegen, die sich widersprechen. Nehmen Sie an, Sie legen für einen Absatz `page-break-after:always` fest, für den folgenden Absatz jedoch `page-break-before:avoid`. Dann muss es natürlich Regeln zur Konfliktlösung geben. Grundsätzlich gilt dabei, dass `always`, `left` und `right` Priorität vor dem Wert `avoid` haben.

Sie finden das Beispiel auf der Buch-CD in der Datei BSP/K10/page_break.html.

Mit dem Stil:

```
h2 { page-break-before: always }
```

könnten Sie beispielsweise definieren, dass vor jeder Überschrift zweiter Ordnung ein Seitenumbruch erfolgen soll.

10.3 Nicht/schlecht unterstützte Attribute

In dieser Rubrik finden Sie solche Attribute, die von vielen oder zumindest von sehr wichtigen Browser nicht oder fehlerhaft unterstützt werden und die somit noch nicht praxistauglich sind.

Hurenkinder (widows)

Von Hurenkindern bzw. Witwen spricht man, wenn die letzte Zeilen eines Absatzes nach einem Seitenumbruch allein auf der neuen Seite steht. Mit der `widows`-Eigenschaft können Sie festlegen, wie viele Zeilen mindestens auf der neuen Seite stehen sollen.

CSS-Element: `widows`	*Mögliche Werte:* `inherit`, Zahl
CSS-Version: CSS 2, 2.1, 3	*Zulässig für folgende (X)HTML-Elemente:* alle Blockelemente
Medium: seitenorientierte Medien	*Vererbt:* Ja

Palm	NN	Mozilla		Internet Explorer							Opera			Safari		iCab	Kq	FF
2.x	4.x	1.x	1.8	4.x	5.0x (Win)	5.0 (Mac)	5.1+ (Mac)	5.5 (Win)	6	7	6	7	8	1.x	2.0	2.9+	3.x	1.0
[1]																		

[1] keine Druckausgabe möglich

Sie finden das Beispiel auf der Buch-CD in der Datei BSP/K10/widows.html.

Mit `p {widows:4}` legen Sie fest, dass am Seitenanfang mindestens 4 Zeilen stehen bleiben müssen.

Der Standardwert der Eigenschaft ist 2.

Pseudo-Klassen :first, :right, :left

Mit den Pseudo-Klassen `:first` (erste Seite), `:right` (rechte Seite) und `:left` (linke Seite) und der `@page`-Regel können Sie festlegen, welche Formatierungen für besondere Seiten gelten sollen.

Es gilt dabei, dass allgemeine Formatierungen (`@page` ohne Pseudo-Klasse) von den Formatierungen mit der Pseudo-Klasse überschrieben werden. `:first` überschreibt überdies die Formatierungen mit `:left` bzw. `:right`, abhängig davon, ob die erste Seite eine linke oder rechte Seite ist.

***C**SS-Element:* `:first`, `:left`, `:right`	*Mögliche Werte:*
CSS-Version: CSS 2-3	*Zulässig für folgende (X)HTML-Elemente:* `@page`-Regel
Medium: seitenorientierte Medien	*Vererbt:* Nein

Palm	NN	Mozilla		Internet Explorer							Opera			Safari		iCab	Kq	FF
2.x	4.x	1.x	1.8	4.x	5.0x (Win)	5.0 (Mac)	5.1+ (Mac)	5.5 (Win)	6	7	6	7	8	1.x	2.0	2.9+	3.x	1.0
[1]											●	●	●					

[1] keine Druckausgabe möglich

Sie finden das Beispiel auf der Buch-CD in der Datei BSP/K10/pseudoklassen.html.

Das Beispiel definiert für die erste Seite einen einheitlichen Seitenrand von 6 cm, für die linken Seiten einen rechten Rand von 6 cm und einen linken von 3 cm. Für die rechten Seiten werden umgekehrte Werte definiert.

```
@page :first { margin:6cm }
@page :left  { margin-right:6cm; margin-left:3cm }
@page :right { margin-left:6cm ;margin-right:3cm }
```

Schneidemarken (marks)

Mit der `marks`-Eigenschaft können Sie den Browser anweisen, Schneidemarken mit anzuzeigen. Sie werden außerhalb des Seitenbereichs ausgegeben und dienen zum korrekten Zuschnitt der Seite (`crop`) bzw. zum korrekten Ausrichten der Seiten (`cross`) bei der maschinellen Weiterverarbeitung der ausgegebenen Seiten.

Damit die Markierungen sichtbar sind, müssen Sie eine absolute Seitengröße angeben.

Mehr dazu finden Sie im Abschnitt »Seitengröße (size)«.

CSS-Element: `marks`	*Mögliche Werte:* `crop` und/oder `cross`, `none`, `inherit`
CSS-Version: CSS 2, CSS 3	*Zulässig für folgende (X)HTML-Elemente:* `@page`-Regel
Medium: seitenorientierte Medien	*Vererbt:* Nein

Palm	NN	Mozilla		Internet Explorer							Opera			Safari		iCab	Kq	FF
2.x	4.x	1.x	1.8	4.x	5.0x (Win)	5.0 (Mac)	5.1+ (Mac)	5.5 (Win)	6	7	6	7	8	1.x	2.0	2.9	3.x	1.0
[1]																		

[1] keine Druckausgabe möglich

Mögliche Werte

- `crop`: erzeugt eine Schnittmarke, die aus einer senkrechten bzw. waagerechten Linie besteht
- `cross`: erzeugt eine Kreuzmarkierung für die Ausrichtung

Die Werte `crop` *und* `cross` *können Sie beide gleichzeitig angeben.*

- `none`: es wird keine Markierung eingefügt.

Sie finden das Beispiel auf der Buch-CD in der Datei BSP/K10/marks.html.

Beispiel

Folgender Codeausschnitt zeigt die Anwendung der Eigenschaft. Über die `size`-Eigenschaft wird eine Seitengröße von 10 x 10 cm definiert. Die `marks`-Eigenschaft legt dann fest, dass Schneide- und Kreuzmarkierungen angezeigt werden sollen.

```
@page { size:10cm 10cm; marks:crop cross }
```

Schusterjungen (orphans)

Von Schusterjungen spricht man, wenn durch einem Seitenumbruch die erste Zeile eines Absatzes allein auf der letzten Seite zurückbleibt. Mit der `orphans`-Eigenschaft können Sie festlegen, wie viele Zeilen mindestens zurückbleiben sollen. Kann diese Anzahl Zeilen nicht eingehalten werden, wird der ganze Absatz auf die nächste Zeile verschoben.

CSS-Element: `orphans`	*Mögliche Werte:* `inherit`, Zahl
CSS-Version: CSS 2, 2.1, 3	*Zulässig für folgende (X)HTML-Elemente:* alle Blockelemente
Medium: seitenorientierte Medien	*Vererbt:* Ja

Palm	NN	Mozilla		Internet Explorer							Opera			Safari		iCab	Kq	FF
2.x	4.x	1.x	1.8	4.x	5.0x (Win)	5.0 (Mac)	5.1+ (Mac)	5.5 (Win)	6	7	6	7	8	1.x	2.0	2.9+	3.x	1.0
[1]																		

[1] keine Druckausgabe möglich

Sie finden das Beispiel auf der Buch-CD in der Datei BSP/K10/orphans.html.

Mit `p {orphans:4)` legen Sie fest, dass am Seitenende mindestens 4 Zeilen stehen bleiben müssen.

Der Standardwert der Eigenschaft ist 2.

Seitengröße (size)

Mit der Eigenschaft `size` können Sie die Größe und Ausrichtung einer Seite festlegen. Berücksichtigt wird die Angabe nur bei Ausgaben auf seitenorientierten Medien. Angewendet wird sie in der `@page`-Regel für eine, alle oder bestimmte Seiten.

CSS-Element: `size`	*Mögliche Werte:* numerische Werte mit Einheit, *auto*, `portrait`, `landscape`, `inherit`, (in CSS 3 auch die vordefinierten Größen `A5`, `A4`, `A3`, `B5`, `B4`, `letter`, `legal` und `ledger`)
CSS-Version: CSS 2, CSS 3	*Zulässig für folgende (X)HTML-Elemente:* `@page`-Regel
Medium: seitenorientierte Medien	*Vererbt:* Nein

Palm	NN	Mozilla		Internet Explorer							Opera			Safari		iCab	Kq	FF
2.x	4.x	1.x	1.8	4.x	5.0x (Win)	5.0 (Mac)	5.1+ (Mac)	5.5 (Win)	6	7	6	7	8	1.x	2.0	2.9+	3.x	1.0
[2]							⊙ [1]				⊙ [1]	⊙ [1]	⊙ [1]					●

[1] siehe unten
[2] keine Druckausgabe möglich

Sie finden das Beispiel auf der Buch-CD in den Dateien BSP/K10/size.html *und* BSP/K10/size2.html.

Sie können die Größe absolut oder relativ definieren. Wenn Sie relative Größen verwenden, kann der Browser das Dokument skalieren, um es auf die gewünschte Seitengröße des Ausgabemediums anzupassen.

Mögliche Werte

Relative Größenangaben können Sie mit folgenden Werten erreichen:

- `auto`: Hiermit wird die Größe der Seite an die Größe des Ausgabemediums angepasst.
- `landscape`: Die Größe der Seitenbox wird an die Größe des Ausgabemediums angepasst und die Seitenbox im Querformat ausgerichtet. Das Querformat ist dadurch definiert, dass die längere Seitenkante horizontal ausgerichtet wird.
- `portrait`: Die Größe der Seitenbox wird an die Größe des Ausgabemediums angepasst. Die Ausgabe erfolgt aber im Hochformat.

Absolute Größen definieren Sie mit numerischen Werten, die eine Einheit haben müssen. Dabei geben Sie mit dem ersten Wert die Breite und mit dem zweiten die Höhe an.

Im folgenden Beispiel wird eine Seitengröße definiert, die dem A5-Format entspricht, also 14,8 x 21 cm. Da die Breite geringer ist als die Höhe, müsste die Ausgabe im Hochformat erfolgen, was auch auf der Standardseitengröße A4 Hochformat der meisten Drucker problemlos möglich sein soll. Zusätzlich wurde hier ein grauer Seitenhintergrund definiert, der von einem Rahmen umgeben wurde, damit Sie erkennen können, wie groß der Inhaltsbereich definiert ist.

Listing 10.1: Anwendung der size-Eigenschaft

```
<style type="text/css" media="print">
    <!--
    @page { size: 14.8cm 21cm; margin:0 }
    body  { color:red; background-color:silver;
            border:1px dotted black;
            width:100%; height:100%; margin:0; }
    -->
</style>
```

Browserunterstützung

Derzeit ist die Browserunterstützung allerdings eher dürftig. Der Opera-Browser ab Version 6 hält zwar die definierte Seitengröße ein, bricht aber den Inhalt der Seite nicht um, auch dann nicht, wenn Sie für das `body`-Element eine Größe von 100% festlegen, die schließlich den für den Seitenbereich verfügbaren Raum ausfüllen müsste.

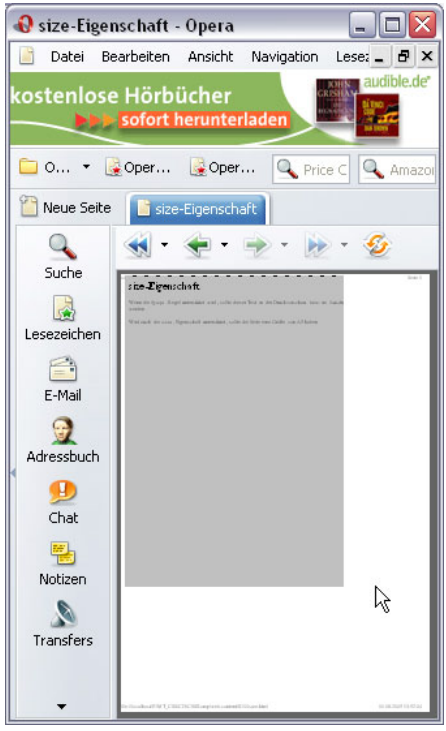


Abbildung 10.2: Korrekte Seitengröße und Ausrichtung im Opera-Browser, aber fehlender Umbruch des Inhalts

Die Mac-Version des Internet Explorers wendet zwar die Seitengröße korrekt an, nicht aber die Seitenausrichtung. Die Seite wird in diesem Fall im Querformat ausgegeben. Das kann nur bedingt als fehlerhaft angesehen werden, da es den Browsern gemäß CSS-Standard erlaubt ist, den Benutzern zu ermöglichen, in die Übertragung der Seitenbox auf das Ausgabemedium einzugreifen. So könnte beispielsweise durch Standard- und Benutzereinstellungen des Browsers die Ausgabe im Querformat erfolgen.

Mac

Seitenrand (margin)

Mit der `margin`-Eigenschaft können Sie bei Anwendung auf die `@page`-Regel den Seitenrand definieren. Im Prinzip funktioniert die `margin`-Eigenschaft hier wie bei normalen Elementen.

Mehr zu den `margin`*-Eigenschaften finden Sie in Kapitel 5, »Größen, Abstände und Positionierung«.*

REF

CSS-Element: `margin`	*Mögliche Werte:* numerische Werte mit Einheit, prozentuale Werte
CSS-Version: CSS 2-3	*Zulässig für folgende (X)HTML-Elemente:* `@page`-Regel
Medium: seitenorientierte Medien	*Vererbt:* Nein

Palm	NN	Mozilla		Internet Explorer							Opera			Safari		iCab	Kq	FF
2.x	4.x	1.x	1.8	4.x	5.0x (Win)	5.0 (Mac)	5.1+ (Mac)	5.5 (Win)	6	7	6	7	8	1.x	2.0	2.9+	3.x	1.0
[1]											⊙	⊙	⊙					

[1] keine Druckausgabe möglich

Sie finden das Beispiel auf der Buch-CD in der Datei BSP/K10/margin.html.

Der Opera-Browser ist der einzige Browser, der zurzeit den Seitenrand unterstützt, das allerdings fehlerhaft. Bei der Seitengröße `size:auto` wird trotzdem der Seiteninhalt abgeschnitten, wenn ein Seitenrand gesetzt wird. Eigentlich müsste der Browser die Ausgabegröße an das Ausgabemedium anpassen.

Abbildung 10.3: Der Text wird abgeschnitten, die Ränder werden allerdings korrekt eingehalten.

Seitentyp (page)

Mit der `page`-Eigenschaft können Sie einen bestimmten Seitentyp in der `@page`-Regel auswählen.

CSS-Element: `page`	*Mögliche Werte: auto*, Bezeichner
CSS-Version: CSS 2, CSS 3	*Zulässig für folgende (X)HTML-Elemente:* alle Blockelemente
Medium: seitenorientierte Medien	*Vererbt:* Ja

Palm	NN	Mozilla		Internet Explorer							Opera			Safari		iCab	Kq	FF
2.x	4.x	1.x	1.8	4.x	5.0x (Win)	5.0 (Mac)	5.1+ (Mac)	5.5 (Win)	6	7	6	7	8	1.x	2.0	2.9+	3.x	1.0
[1]																		

[1] keine Druckausgabe möglich

Als Bezeichner geben Sie den Namen eines zuvor mit `@page` definierten Seitentyps an. Als Bezeichner sind prinzipiell die gleichen Bezeichner erlaubt, die in CSS auch für Klassen- und ID-Stile zulässig sind.

Mehr zur Namensgebung von Stilen finden Sie in Kapitel 2, »Stile definieren«.

Sie finden das Beispiel auf der Buch-CD in der Datei BSP/K10/page.html.

Beispiel

Mit der `page`-Eigenschaft wählen Sie innerhalb des zu druckenden Elements den Namen des Seitentyps aus. Die Angabe `style="page:querformat"` würde beispielsweise bewirken, dass für die Ausgabe des Absatzes der Seitentyp `querformat` verwendet wird. Den definieren Sie wiederum mit der `@page`-Regel und weisen dem Typ dann die entsprechenden Regeln in den geschweiften Klammern zu.

Logischerweise kann sich die Seitenausrichtung nicht innerhalb einer Seite ändern, sondern nur von einer Seite zu nächsten. Normalerweise sollte der Browser selbstständig für einen Seitenumbruch sorgen, wenn zwei aufeinander folgende Blockelemente mit unterschiedlichen `page`*-Werten ausgegeben werden. Zur Sicherheit sorgen im Beispiel die Angaben* `page-break-before:always` *und* `page-break-after:always` *jedoch noch einmal dafür, dass vor und nach den Absätzen ein Seitenumbruch erfolgt.*

Listing 10.2: Verwendung der page-Eigenschaft

```
<head>
    <title>page-Eigenschaft</title>
    <style type="text/css" media="print">
    <!--
        @page hochformat { size:portrait; }
        @page querformat { size:landscape; }
        body             { background-color:silver;
                           border:1px dotted black;
                           width:100%; height:100%;
                           margin:0; color:red
        }
    -->
    </style>
</head>
<body>
    <h1>page-Eigenschaft</h1>
    <p style="page:querformat;page-break-before:always;page-break-
        after:always">Falls die page-Eigenschaft unterst&uuml;tzt wird,
        sollte dieser Absatz im Querformat ausgegeben werden.</p>
    <p style="page:hochformat;page-break-before:always;page-break-
        after:always">Dieser Absatz sollte hingegen im Hochformat
        ausgegeben werden. </p>
</body>
```

10.4 Einführung Sprachausgabe

Die Sprachausgabe funktioniert so, dass sie keine visuelle Ausgabe der Seite erzeugt, sondern eine akustische. Dies geschieht in der Regel dadurch, dass der Inhalt der Seite in reinen Text konvertiert und vorgelesen wird. Um dennoch wichtige Stellen hervorheben zu können, gibt es die Möglichkeit akustische Icons zu definieren, mit denen Sie beispielsweise Links kennzeichnen können.

Die akustische Darstellung eines Dokuments erfolgt innerhalb einer Klangumgebung, die dreidimensional und temporär ist. Letzteres bedeutet, dass Sie die Reihenfolge bestimmen können, in der Klänge erzeugt werden, Sie können dabei nicht nur festlegen, in welcher Lautstärke ein Element vorgelesen werden soll, sondern auch mit welcher Stimme, Stimmlage und in welcher Geschwindigkeit. Die Möglichkeiten sind also enorm.

Anwendungsmöglichkeiten

Sprachausgabe wird im Augenblick vor allem in Zusammenhang mit barrierefreien Webseiten genannt, weil es sich um eine Technik handelt, die vor allem Blinden entgegenkommt. Aber auch für andere Einsatzgebiete wie Telekommunikation, Dokumentationssysteme und Informationssysteme (u.a. Fahrplanauskünfte, Telefonauskünfte) ist eine akustische Ausgabe denkbar. Praxistauglich ist aber bisher nur die Sprachausgabe für Blinde, die schon recht häufig genutzt wird, dazu werden allerdings bisher vor allem Screenreader eingesetzt und die lesen nur den Inhalt des Bildschirms vor, ohne selbst die Webseite zu parsen und dann auch den CSS-Code zu berücksichtigen.

Browserunterstützung

Die Browserunterstützung ist eher dürftig. Keiner der großen Browser unterstützt derzeit eine Sprachausgabe. Das heißt jedoch nicht, dass Sie die Befehle nicht einsetzen können und sollten.

Der einzige mir bekannte Browser, der eine Sprachausgabe unterstützt, ist iCab 2.9.x und höher. Allerdings unterstützt er die CSS-Befehle für die Sprachausgabe noch nicht, sondern liest die Seite einfach mit Standardeinstellungen vor.

Auch wenn also im nachfolgenden Referenzteil die Browserunterstützung als nicht vorhanden angegeben wird, können Sie die Befehle auch heute schon einsetzen, weil diese die visuelle Darstellung der Seite für die normale Anzeige im Browser nicht negativ beeinflussen.

Damit ein Ausgabeprogramm die Wörter korrekt aussprechen kann, ist es wichtig, dass Sie für Ihre Seite und die auszugebenden Elemente deren Sprache mit dem `lang`*-Attribut definieren. Schließlich macht es auch in der Bedeutung einen Unterschied, ob das Programm das englische Wort »sun« (sprich: san) englisch korrekt oder wie im Deutschen »sun« ausspricht. Damit geht der ganze Sinn verloren. Selbstverständlich gilt das auch für fremdsprachliche Wörter in Texten einer anderen Sprache. Wenn Sie beispielsweise für das* `body`*-Element mit* `lang="de"` *eine deutsche Sprache definiert haben, können Sie englische Wörter im Fließtext, beispielsweise wie folgt kennzeichnen:* `<h1><span lang="en">Biker-</span>Club</h1>`.

10.5 Eigenschaften zur Sprachausgabe

Nachfolgend finden Sie die Eigenschaften, die CSS für die Steuerung der Sprachausgabe zur Verfügung stellt. Die aktuellen großen Browser unterstützen sie zwar nicht, dennoch gibt es Software, insbesondere solche für Blinde, die diese Teile des CSS-Standards umsetzen.

Akustische Icons (cue-before, cue-after und cue)

Mit den Eigenschaften `cue-before` und `cue-after` können Sie vor und nach einem Element ein akustisches Icon implementieren. So können Sie beispielsweise vor und nach besonders wichtigen Abschnitten einen Ton abspielen lassen, der dem Benutzer die Wichtigkeit deutlich macht.

Die Eigenschaft `cue` stellt wiederum eine Kurzform dar. Der erste Wert von `cue` gilt für `cue-before`, der zweite für `cue-after`.

CSS-Element: `cue`, `cue-before`, `cue-after`	*Mögliche Werte: none*, URI, `inherit`
CSS-Version: CSS 2, CSS 3	*Zulässig für folgende (X)HTML-Elemente:* alle
Medium: Aural	*Vererbt:* Nein

Palm	NN	Mozilla		Internet Explorer							Opera			Safari		iCab	Kq	FF
2.x	4.x	1.x	1.8	4.x	5.0x (Win)	5.0 (Mac)	5.1+ (Mac)	5.5 (Win)	6	7	6	7	8	1.x	2.0	2.9+	3.x	1.0

Mögliche Werte

- URI: Mit diesem Wert geben Sie den URL der Klangdatei an, der als akustisches Icon abgespielt werden soll. Sollte es sich bei der angegebenen Datei nicht um eine Klangdatei handeln, soll die Angabe ignoriert werden.
- `none`: Es wird keine Klangdatei abgespielt.

Wenn Sie für ein Element sowohl eine Pause (siehe Abschnitt »Pausen (pause-before, pause-after und pause)«) als auch einen Klang definieren, werden die Eigenschaften in folgender Reihenfolge berücksichtigt:

1. *Klang vor dem Element*
2. *Pause vor dem Element*
3. *Element*
4. *Pause nach dem Element*
5. *Klang nach dem Element*

Sie finden das Beispiel auf der Buch-CD in der Datei BSP/K10/sprachausgabe.html.

Wenn Sie vor Zitaten einen Klang und eine Pause abspielen möchten, könnten Sie dazu folgenden Stil definieren:

```
cite { pause-before:1000ms; pause-after:1s;
       cue-before:url("ding.wav") }
```

Aussprache der Interpunktion (speak-punctuation)

Die Eigenschaft `speak-punctuation` legt fest, wie Interpunktionszeichen (d.h. Punkt und Komma etc.) auszusprechen sind.

CSS-Element: `speak-punctuation`	*Mögliche Werte: none*, `code`, `inherit`
CSS-Version: CSS 2, CSS 3	*Zulässig für folgende (X)HTML-Elemente:* alle
Medium: Aural	*Vererbt:* Ja

Palm	NN	Mozilla		Internet Explorer							Opera			Safari		iCab	Kq	FF
2.x	4.x	1.x	1.8	4.x	5.0x (Win)	5.0 (Mac)	5.1+ (Mac)	5.5 (Win)	6	7	6	7	8	1.x	2.0	2.9+	3.x	1.0

Mögliche Werte

- `code`: Die Interpunktionszeichen werden ausgesprochen. Ein Satz wie »Ich bin zu spät gekommen, weil die Bahn Verspätung hatte.«, würde wie folgt gesprochen: »Ich bin zu spät gekommen KOMMA weil die Bahn Verspätung hatte PUNKT«.
- `none`: Die Interpunktion wird durch natürliche Pausen angezeigt.

Sie finden das Beispiel auf der Buch-CD in der Datei BSP/K10/sprachausgabe.html. *Erläuterungen zum Beispiel finden Sie im Abschnitt »Aussprache von Zahlen (speak-numeral)«.*

Aussprache von Überschriften (speak-header)

Die Eigenschaft `speak-header` bestimmt, wie Tabellen akustisch ausgegeben werden. Sie legt fest, wie Spalten und Zeilenüberschriften gesprochen werden, ob sie vor jedem Wert der Spalte/Zeile oder nur einmal am Anfang der Tabellenstruktur vorgelesen werden.

CSS-Element: `speak-header`	*Mögliche Werte: once*, `always`, `inherit`
CSS-Version: CSS 2, CSS 3	*Zulässig für folgende (X)HTML-Elemente:* alle Elemente, die Spalten- oder Zeilentitel sind bzw. mit `display` als solche formatiert sind
Medium: Aural	*Vererbt:* Ja

Palm	NN	Mozilla		Internet Explorer							Opera			Safari		iCab	Kq	FF
2.x	4.x	1.x	1.8	4.x	5.0x (Win)	5.0 (Mac)	5.1+ (Mac)	5.5 (Win)	6	7	6	7	8	1.x	2.0	2.9+	3.x	1.0

Mögliche Werte

- `once`: Die Überschrift wird nur einmal gesprochen.
- `always`: Die Überschrift wird vor jeder Zelle gesprochen, die der Überschrift zugeordnet werden kann.

Sie finden das Beispiel auf der Buch-CD in der Datei BSP/K10/sprachausgabe.html.

Beispiel

Im Beispiel wird eine Tabelle wie in Abbildung 10.4 definiert.

Abbildung 10.4: Die definierte Tabelle

	1. Halbjahr	2. Halbjahr
Verkäufer 1:	20.278	25.293
Verkäufer 2:	7.293	18.283

Mit dem Elementstil

```
th { speak-header: always }
```

wird festgelegt, wie die Tabelle vorgelesen wird. Mit dem Wert `always` wird die Tabelle folgendermaßen vorgelesen:

»Verkäufer 1: 1. Halbjahr 20.278 2. Halbjahr 25.293 Verkäufer 2: 1. Halbjahr 7.293 2. Halbjahr 18.283«

Würden Sie hingegen den Wert `once` verwenden, würde die Tabelle folgendermaßen ausgegeben werden:

»1. Halbjahr 2. Halbjahr Verkäufer 1: 20.278 25.293 Verkäufer 2: 7.293 18.283«

Aussprache von Zahlen (speak-numeral)

Die Eigenschaft `speak-numeral` definiert, wie Zahlen ausgesprochen werden.

CSS-Element: `speak-numeral`	*Mögliche Werte:* `digits`, *continuous*, `inherit`
CSS-Version: CSS 2, CSS 3	*Zulässig für folgende (X)HTML-Elemente:* alle
Medium: Aural	*Vererbt:* Ja

Palm	NN	Mozilla		Internet Explorer							Opera			Safari		iCab	Kq	FF
2.x	4.x	1.x	1.8	4.x	5.0x (Win)	5.0 (Mac)	5.1+ (Mac)	5.5 (Win)	6	7	6	7	8	1.x	2.0	2.9+	3.x	1.0

Die korrekte Aussprache von Zahlen ist vor allem wichtig, wenn Ihr Text auch Codes und ähnliches enthält, weil dieser als Zahl vorgelesen wenig Sinn ergäbe. Verweisen Sie beispielsweise im Text auf bestimmte nummerierte Abschnitte, ist es natürlich günstiger, den Abschnitt 1.2 mit »Eins PUNKT Zwei« ausgeben zu lassen, als mit 1 2. Um eine korrekte Aussprache zu erreichen, müssen Sie daher nicht nur den Wert der Eigenschaft `speak-numeral` *korrekt setzen, sondern auch die Eigenschaft* `speak-punctuation`.

Mögliche Werte

- `digits`: Zahlen werden als Ziffern ausgesprochen. Die Zahl 100 würde also mit »Eins Null Null« vorgelesen.
- `continuous`: Zahlen werden als Wörter ausgesprochen. Die Zahl 100 würde mit »Einhundert« vorgelesen werden. Wie die Zahlen als Wörter vorgelesen werden, ist abhängig von der Sprache des Textes.

Sie finden das Beispiel auf der Buch-CD in der Datei BSP/K10/sprachausgabe.html.

Haben Sie im Code beispielsweise folgende Liste definiert:

Listing 10.3: Listendefinition

```
<div class="numliste">
    <p>1. Die Nummerierungen vor den Einträgen</p>
    <p>2. sollten als einzelne Ziffern ausgesprochen werden. </p>
    <p>2.1 Die Punkte dazwischen sollten ebenfalls als Wörter
        ausgesprochen werden.</p>
</div>
```

benötigen Sie folgende CSS-Klasse, um die Listennummern korrekt auszusprechen. Sie werden dann mit »Eins Punkt«, »Zwei Punkt« und »Zwei Punkt Eins Punkt« vorgelesen.

```
.numliste { speak-punctuation: code;
            speak-numeral: digits }
```

Die Eigenschaft `speak-punctuation:code` sorgt dafür, dass bei Zahlen mit Punkt der »Punkt« mit ausgesprochen wird, `speak-numeral` sorgt dafür, dass die Zahlen als Ziffern ausgesprochen werden.

Frequenzbereich (pitch-range)

Die Eigenschaft `pitch-range` legt den Frequenzbereich der Stimme fest, das heißt genauer, die Abweichung von der durchschnittlichen Stimmlage. Generell gilt dabei, je kleiner der Wert für die Eigenschaft ist, desto monotoner ist die Stimme. Je höher der Wert ist, desto lebhafter ist die Stimme. Der Standardwert der Eigenschaft ist 50.

CSS-Element: `pitch-range`	*Mögliche Werte:* numerischer Wert, `inherit`
CSS-Version: CSS 2, CSS 3	*Zulässig für folgende (X)HTML-Elemente:* alle
Medium: Aural	*Vererbt:* Ja

Palm	NN	Mozilla		Internet Explorer							Opera			Safari		iCab	Kq	FF
2.x	4.x	1.x	1.8	4.x	5.0x (Win)	5.0 (Mac)	5.1+ (Mac)	5.5 (Win)	6	7	6	7	8	1.x	2.0	2.9+	3.x	1.0

Mögliche Werte

Der Wert der Eigenschaft darf eine ganze Zahl zwischen 0 und 100 (einschließlich) sein.

Sie finden das Beispiel auf der Buch-CD in der Datei BSP/K10/sprachausgabe.html.

Frequenzspitze (stress)

Mit der `stress`-Eigenschaft definieren Sie die Abweichung in der Höhe der Stimme für betonte Silben und Satzteile. Der Standardwert ist 50.

CSS-Element: `stress`	*Mögliche Werte:* numerischer Wert, `inherit`
CSS-Version: CSS 2, CSS 3	*Zulässig für folgende (X)HTML-Elemente:* alle
Medium: Aural	*Vererbt:* Ja

Palm	NN	Mozilla		Internet Explorer							Opera			Safari		iCab	Kq	FF
2.x	4.x	1.x	1.8	4.x	5.0x (Win)	5.0 (Mac)	5.1+ (Mac)	5.5 (Win)	6	7	6	7	8	1.x	2.0	2.9+	3.x	1.0

Wie bei der Eigenschaft `pitch-range` sind als Werte ganze Zahlen von 0 bis 100 zulässig. Allerdings ist das Ergebnis ein- und desselben Wertes abhängig von der Stimme und der gesprochenen Sprache.

Sie finden das Beispiel auf der Buch-CD in der Datei BSP/K10/sprachausgabe.html.

Hintergrundsound (play-during)

Mit der Eigenschaft `play-during` können Sie einen Hintergrundsound für ein Element definieren. Er wird abgespielt, während der Inhalt des Elements ausgesprochen wird.

CSS-Element: `play-during`	*Mögliche Werte:* URI, `mix`, `repeat`, *auto*, `none`, `inherit`
CSS-Version: CSS 2, CSS 3	*Zulässig für folgende (X)HTML-Elemente:* alle
Medium: Aural	*Vererbt:* Nein

Palm	NN	Mozilla		Internet Explorer							Opera			Safari		iCab	Kq	FF
2.x	4.x	1.x	1.8	4.x	5.0x (Win)	5.0 (Mac)	5.1+ (Mac)	5.5 (Win)	6	7	6	7	8	1.x	2.0	2.9+	3.x	1.0

Mögliche Werte

- URI: gibt den URL der abzuspielenden Sound-Datei an. Der URL muss angegeben werden, wenn eines der Schlüsselwörter `mix` oder `repeat` angegeben wurde.
- `mix`: definiert, dass der Hintergrundsound des übergeordneten Elements weiter abgespielt wird und mit der im URL angegebenen Sound-Datei gemischt wird. Falls Sie `mix` nicht angeben, ersetzt der im URL angegebene Hintergrundsound den Hintergrundsound des übergeordneten Elements.
- `repeat`: Bei Angabe dieses Schlüsselwortes wird der Sound wiederholt, bis der gesamte Inhalt des Elements vorgelesen ist. Falls der Sound zu lang ist, wird er beendet, wenn das Element vollständig ausgegeben wurde.
- `auto`: Bei diesem Wert wird der Sound des übergeordneten Elements fortgesetzt.
- `none`: Hat das übergeordnete Element einen Sound, wird dieser abgebrochen und gegebenenfalls nach dem aktuellen Element weiter fortgesetzt.

Die Werte für URL, `mix` *und* `repeat` *können Sie gleichzeitig angeben.*

Sie finden das Beispiel auf der Buch-CD in der Datei BSP/K10/sprachausgabe.html.

Beispiel

Folgender Stil definiert einen Hintergrundsound für die ganze Seite, der wiederholt wird.

```
body { play-during:url("wellen.wav") repeat }
```

Horizontale Sound-Position (azimuth)

Die Eigenschaft `azimuth` definiert die Position der Ausgabe innerhalb einer 3D-Klangarena. Sie dient dazu, räumliche Klänge zu erzeugen. Wie die Ausgabe von der verwendeten Hardware und/oder dem Ausgabeprogramm erzeugt wird, legt der CSS-Standard jedoch nicht fest.

CSS-Element: `azimuth`	*Mögliche Werte:* Winkel, `left-side`, `far-left`, `left`, `center-left`, `center`, `center-right`, `right`, `far-right`, `right-side`, `behind`, `leftwards`, `rightwards`, `inherit`
CSS-Version: CSS 2, CSS 3	*Zulässig für folgende (X)HTML-Elemente:* alle
Medium: Aural	*Vererbt:* Ja

Palm	NN	Mozilla		Internet Explorer							Opera			Safari		iCab	Kq	FF
2.x	4.x	1.x	1.8	4.x	5.0x (Win)	5.0 (Mac)	5.1+ (Mac)	5.5 (Win)	6	7	6	7	8	1.x	2.0	2.9	3.x	1.0

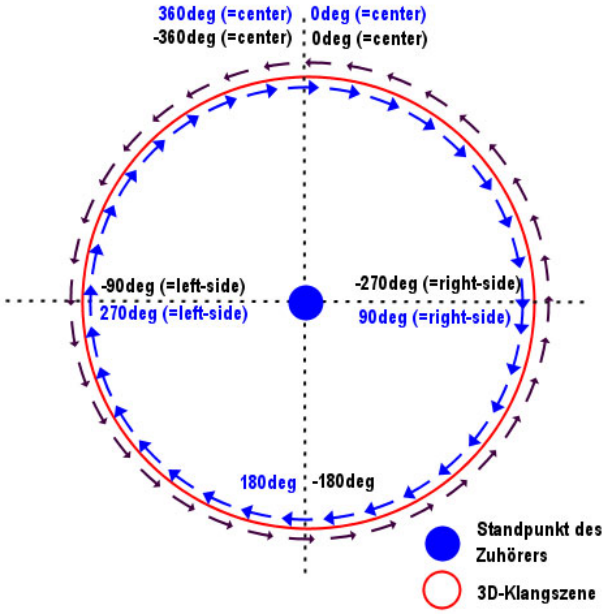
Mögliche Werte

- Winkel: Als Wert geben Sie einen Winkel mit der Einheit »deg« an. Der Bereich geht von `-360 deg` bis `+360 deg`. Der Wert 0 repräsentiert dabei den vorderen Mittelpunkt der Klangszene.

INFO

Folgendes Bild verdeutlicht die Klangszene. Im Mittelpunkt eines virtuellen Kreises, auf dem die Klangquellen angeordnet werden können, steht der Zuhörer. Über die Werte der `azimuth`*-Eigenschaft können Sie nun festlegen, wo genau die Klangquelle für die Ausgabe stehen soll. Dazu gibt es zwei Möglichkeiten. Sie bewegen sich ausgehend vom Wert 0, der direkt vor dem Betrachter liegt, im Uhrzeigersinn um den Betrachter herum. Dann geben Sie positive Gradzahlen an. Falls Sie entgegen dem Uhrzeigersinn gehen, sind es negative Werte. Die Winkelangaben* `-270 deg` *und* `90 deg` *führen also zum gleichen Ergebnis.* `-360 deg`*,* `+360 deg` *und* `0 deg` *sind mit dem Schlüsselwort* `center` *identisch.*

Abbildung 10.5: Aufbau der 3D-Klangszene



- `left-side`: entspricht `270 deg`
- `far-left`: entspricht `300 deg`
- `left`: entspricht `320 deg`
- `center-left`: entspricht `340 deg`
- `center`: entspricht `0 deg`
- `center-right`: entspricht `20 deg`
- `right`: entspricht `40 deg`
- `far-right`: entspricht `60 deg`
- `right-side`: entspricht `90 deg`
- `leftwards`: verschiebt den Sound entgegen der Uhrzeigerrichtung um 20 Grad
- `rightwards`: verschiebt den Sound in Uhrzeigerrichtung um 20 Grad

CD

Sie finden das Beispiel auf der Buch-CD in der Datei BSP/K10/sprachausgabe.html.

Lautstärke (volume)

Mit der Eigenschaft `volume` können Sie die Lautstärke der Sprachausgabe angeben. Zulässig sind dabei sowohl numerische und prozentuale Werte als auch Schlüsselwörter.

Gemäß CSS-Standard sollte das Ausgabeprogramm den Benutzern die Möglichkeit bieten, die Werte 0 und 100 manuell zu bestimmen, um festzulegen, welche Lautstärke die leiseste (0) und lauteste (100) Ausgabe hat.

CSS-Element: `volume`	*Mögliche Werte:* numerischer Wert, Prozentwert, `silent`, `x-soft`, `soft`, *medium*, `loud`, `x-loud`, `inherit`
CSS-Version: CSS 2, CSS 3	*Zulässig für folgende (X)HTML-Elemente:* alle
Medium: Aural	*Vererbt:* Ja

Palm	NN	Mozilla		Internet Explorer							Opera			Safari		iCab	Kq	FF
2.x	4.x	1.x	1.8	4.x	5.0x (Win)	5.0 (Mac)	5.1+ (Mac)	5.5 (Win)	6	7	6	7	8	1.x	2.0	2.9+	3.x	1.0

Mögliche Werte

- Prozentzahlen: beziehen sich auf den geerbten Wert und können Werte von 0 bis 100 repräsentieren.
- Numerische Werte: eine Zahl zwischen 0 und 100. Bei null ist die Ausgabe gerade noch hörbar, bei 100 gerade noch erträglich.
- `silent`: Die Ausgabe ist stumm geschaltet.
- `x-soft`: entspricht dem Wert 0.
- `soft`: leise, entspricht dem Wert 25.
- `medium`: mittel, entspricht dem Wert 50.
- `loud`: laut, entspricht dem Wert 75.
- `x-loud`: sehr laut, entspricht dem Wert 100.

Sie finden das Beispiel auf der Buch-CD in der Datei BSP/K10/sprachausgabe.html.

Beispiel

Folgender Code ermöglicht es Ihnen beispielsweise, alle Elemente mit der Klasse `.navigation` sowie die Überschriften erster und zweiter Ordnung mit der Lautstärke »Mittel« und den übrigen Inhalt mit der Lautstärke »Leise« auszugeben:

```
p                  { volume:soft }
h1,h2, .navigation { volume:medium }
```

Pausen (pause-before, pause-after und pause)

Mit den Eigenschaften `pause-before` (Pause vor dem Element), `pause-after` (Pause nach dem Element) und `pause` (Kurzform) können Sie bestimmen, ob und wann Pausen bei der Sprachausgabe gemacht werden. Der Standardwert ist jeweils 0.

CSS-Element: `pause`, `pause-before`, `pause-after`	*Mögliche Werte:* Zeitwert, Prozentwerte, `inherit`
CSS-Version: CSS 2, CSS 3	*Zulässig für folgende (X)HTML-Elemente:* alle
Medium: Aural	*Vererbt:* Nein

Palm	NN	Mozilla		Internet Explorer							Opera			Safari		iCab	Kq	FF
2.x	4.x	1.x	1.8	4.x	5.0x (Win)	5.0 (Mac)	5.1+ (Mac)	5.5 (Win)	6	7	6	7	8	1.x	2.0	2.9+	3.x	1.0

Werte für `pause-after` *und* `pause-before`

- Zeitwert: gibt die Länge der Pause in Sekunden und Millisekunden an.
- Prozentwert: bezieht sich auf die Sprechgeschwindigkeit, die mit der Eigenschaft `speech-rate` festgelegt wird. Ein Wert von 100% bewirkt, dass die Pause genauso lang ist, wie es durchschnittlich dauert, ein einzelnes Wort zu sprechen.

Beispiel

Das folgende Beispiel definiert eine Pause von 1 Sekunde vor und nach einem Zitat – einmal in Millisekunden und einmal in Sekunden.

```
cite { pause-before:1000ms; pause-after:1s }
```

Sie finden das Beispiel auf der Buch-CD in der Datei BSP/K10/sprachausgabe.html.

pause-*Eigenschaft*

Die Eigenschaft `pause` stellt eine Abkürzung dar. Sie können dazu einen oder zwei Werte übergeben. Geben Sie zwei an, ist der erste der Wert für `pause-before`, der zweite der für `pause-after`. Bei nur einem Wert bekommen beide Eigenschaften diesen einen Wert.

Das vorstehende Beispiel könnten Sie also auch ersetzen durch:

```
cite { pause:1000ms 1s }
```

oder

```
cite { pause:1s }
```

CSS-Element: `pause`	*Mögliche Werte:* Zeitwert, Prozentwerte
CSS-Version: CSS 2, CSS 3	*Zulässig für folgende (X)HTML-Elemente:* alle
Medium: Aural	*Vererbt:* Nein

Palm	NN	Mozilla		Internet Explorer							Opera			Safari		iCab	Kq	FF
2.x	4.x	1.x	1.8	4.x	5.0x (Win)	5.0 (Mac)	5.1+ (Mac)	5.5 (Win)	6	7	6	7	8	1.x	2.0	2.9+	3.x	1.0

Spracheigenschaften (speak)

Mit der Eigenschaft `speak` können Sie festlegen, wie die Elemente sprachlich dargestellt werden sollen.

CSS-Element: `speak`	*Mögliche Werte: normal*, `none`, `spell-out`, `inherit`
CSS-Version: CSS 2, CSS 3	*Zulässig für folgende (X)HTML-Elemente:* alle
Medium: Aural	*Vererbt:* Ja

Palm	NN	Mozilla		Internet Explorer							Opera			Safari		iCab	Kq	FF
2.x	4.x	1.x	1.8	4.x	5.0x (Win)	5.0 (Mac)	5.1+ (Mac)	5.5 (Win)	6	7	6	7	8	1.x	2.0	2.9	3.x	1.0

Mögliche Werte

- `none`: unterdrückt die Ausgabe. Im Gegensatz zu `volume:silent` wird dafür aber keine Zeit beansprucht.

Untergeordnete Elemente eines mit `speak:none` *beschriebenen Elements können den Wert der* `speak`*-Eigenschaft überschreiben, so dass sie dennoch gesprochen werden. Bei der* `display`*-Eigenschaft ist das nicht der Fall. Auch* `display:none` *bewirkt, dass ein Element nicht dargestellt wird, weder akustisch noch visuell. Sie sollten daher* `display:none` *verwenden, wenn Sie sichergehen möchten, dass auch untergeordnete Elemente nicht dargestellt werden.*

Mehr zur `display`*-Eigenschaft finde Sie in Kapitel 5, »Größen, Abstände und Positionierung«.*

- `normal`: Es werden sprachabhängig Regeln zur Aussprache des Elements und seiner untergeordneten Elemente verwendet.
- `spell-out`: Der Wert sorgt dafür, dass der Text buchstabiert wird. Das ist beispielsweise sehr nützlich für Abkürzungen, Codes etc., die keine aussprechbare Zeichenfolge darstellen.

Sie finden das Beispiel auf der Buch-CD in der Datei BSP/K10/sprachausgabe.html.

Nehmen Sie an, in Ihrem Text kommt das Wort »ABC-Schützen« vor. Dann sollten Sie dies im Code wie folgt definieren:

```
<p><span style="speak:spell-out">ABC</span>-Schützen</p>
```

Sprechgeschwindigkeit (speech-rate)

Die Eigenschaft `speech-rate` bestimmt die Sprechgeschwindigkeit und stellt dazu analog zur Eigenschaft `font-size` absolute und relative Schlüsselwörter für die Geschwindigkeit zur Verfügung.

CSS-Element: `speech-rate`	*Mögliche Werte:* numerischer Wert, `x-slow`, `slow`, ***medium***, `fast`, `x-fast`, `faster`, `slower`, `inherit`
CSS-Version: CSS 2, CSS 3	*Zulässig für folgende (X)HTML-Elemente:* alle
Medium: Aural	*Vererbt:* Ja

Palm	NN	Mozilla		Internet Explorer							Opera			Safari		iCab	Kq	FF
2.x	4.x	1.x	1.8	4.x	5.0x (Win)	5.0 (Mac)	5.1+ (Mac)	5.5 (Win)	6	7	6	7	8	1.x	2.0	2.9	3.x	1.0

Mögliche Werte

- numerischer Wert: Mit einem numerischen Wert geben Sie die Anzahl Wörter pro Minute an.

!! STOP

Beachten Sie, dass bei Angabe eines numerischen Wertes die Geschwindigkeit abhängig von der Sprache unterschiedlich ausfallen kann. Bei Sprachen, die verhältnismäßig viele kurze Wörter haben, ist die Ausgabe mit dem Wert 500 natürlich langsamer als bei Sprachen, die im Schnitt sehr lange Wörter haben. Um hier ebenso 500 Wörter pro Minute zu sprechen, muss jedes einzelne Wort natürlich schneller ausgesprochen werden.

- `x-slow`: ca. 80 Wörter pro Minute
- `slow`: ca. 120 Wörter pro Minute
- `medium`: ca. 190 Wörter pro Minute
- `fast`: ca. 300 Wörter pro Minute
- `x-fast`: ca. 500 Wörter pro Minute
- `faster`: ca. 40 Wörter pro Minute mehr als die aktuelle (geerbte) Sprechgeschwindigkeit
- `slower`: ca. 40 Wörter pro Minute weniger als die aktuelle (geerbte) Sprechgeschwindigkeit

Sie finden das Beispiel auf der Buch-CD in der Datei BSP/K10/sprachausgabe.html.

Stimmfamilie (voice-family)

Mit der Eigenschaft `voice-family` können Sie den Sprechertyp festlegen. Sie ist damit vergleichbar mit der `font-family`-Eigenschaft für die visuelle Ausgabe.

CSS-Element: `voice-family`	*Mögliche Werte:* Stimme, generische Stimme, `inherit`
CSS-Version: CSS 2, CSS 3	*Zulässig für folgende (X)HTML-Elemente:* alle
Medium: Aural	*Vererbt:* Ja

Palm	NN	Mozilla		Internet Explorer							Opera			Safari		iCab	Kq	FF
2.x	4.x	1.x	1.8	4.x	5.0x (Win)	5.0 (Mac)	5.1+ (Mac)	5.5 (Win)	6	7	6	7	8	1.x	2.0	2.9+	3.x	1.0
															○ 1		○ 1	●

Als Wert geben Sie eine oder mehrere alternative Stimmen an, die Sie durch Kommata trennen. Zunächst wird die zuerst angegebene Stimme verwendet, ist die nicht verfügbar, die als zweites genannte etc. Daher gilt hier, wie auch bei der Eigenschaft `font-family`, dass Sie als Letztes immer eine generische Stimme angeben sollten.

Mehr zur `font-family`*-Eigenschaft finden Sie in Kapitel 4, »Textformatierungen«.*

Generische Stimmen

CSS definiert drei generische Stimmen, nämlich

- `male`: männlich
- `female`: weiblich
- `child`: Kind

Spezielle Stimmen

Spezielle Stimmen sind vergleichbar mit den Schriftarten der `font-family`-Eigenschaft. Sie stellen jeweils eine charakteristische Stimme dar. Es kann sich dabei z.B. um die Stimmen von Schauspielern, Comicfiguren etc. handeln, aber auch künstliche Stimmen sind denkbar.

Sie finden das Beispiel auf der Buch-CD in der Datei BSP/K10/sprachausgabe.html.

Beispiel

Folgender Stil definiert für alle Elemente innerhalb des `body`-Elements eine Stimme namens »Sam« und als alternative generische Stimme eine männliche Stimme:

```
body { play-during:url("wellen.wav") repeat;
       azimuth:-90deg; elevation:-90deg;
       speech-rate: fast; voice-family:Sam, male }
```

Stimmen, deren Namen nicht den Namensregeln von CSS entsprechen, weil sie beispielsweise Sonderzeichen, Bindestriche etc. enthalten, sollten Sie in Anführungszeichen einfassen. Gleiches gilt für Namen von Stimmen, die Leerzeichen enthalten.

Stimmfrequenz (pitch)

Mit der `pitch`-Eigenschaft können Sie die durchschnittliche Stimmlage festlegen, die durch ihre Frequenz bestimmt wird. Das Ergebnis ist allerdings abhängig von der verwendeten Stimme. Sie können die Stimmfrequenz absolut oder mit relativen Schlüsselwörtern auch relativ zur Standardfrequenz der verwendeten Stimme festlegen.

CSS-Element: `pitch`	*Mögliche Werte:* Frequenz, `x-low`, `low`, *medium*, `high`, `x-high`, `inherit`
CSS-Version: CSS 2, CSS 3	*Zulässig für folgende (X)HTML-Elemente:* alle
Medium: Aural	*Vererbt:* Ja

Palm	NN	Mozilla		Internet Explorer							Opera			Safari		iCab	Kq	FF
2.x	4.x	1.x	1.8	4.x	5.0x (Win)	5.0 (Mac)	5.1+ (Mac)	5.5 (Win)	6	7	6	7	8	1.x	2.0	2.9+	3.x	1.0

Mögliche Werte

- Frequenz: Als Frequenz geben Sie einen numerischen Wert mit der Einheit `Hz` (für Hertz) an, der die durchschnittliche Frequenz der Stimme bestimmt.
- `x-low`: sehr niedrig
- `low`: niedrig

- `medium`: mittel
- `high`: hoch
- `x-high`: sehr hoch

Diese Schlüsselwörter sind relative Frequenzen, die von den Browsern in absolute Frequenzen umgesetzt werden sollen. Definiert ist nur folgende Beziehung:

```
x-low < low < medium < high < x-high
```

Sie finden das Beispiel auf der Buch-CD in der Datei BSP/K10/sprachausgabe.html.

Stimmumfang (richness)

Der Stimmumfang bzw. das Stimmvolumen bestimmt, wie gut die Stimme in einem größeren Raum trägt bzw. wie sanft eine Stimme ist. Die Eigenschaft `richness` legt das Stimmvolumen fest.

CSS-Element: `richness`	*Mögliche Werte:* numerischer Wert, `inherit`
CSS-Version: CSS 2, CSS 3	*Zulässig für folgende (X)HTML-Elemente:* alle
Medium: Aural	*Vererbt:* Ja

Palm	NN	Mozilla		Internet Explorer							Opera			Safari		iCab	Kq	FF
2.x	4.x	1.x	1.8	4.x	5.0x (Win)	5.0 (Mac)	5.1+ (Mac)	5.5 (Win)	6	7	6	7	8	1.x	2.0	2.9+	3.x	1.0

Als numerischen Wert geben Sie eine ganze Zahl zwischen 0 und 100 an. Je niedriger der Wert ist, desto sanfter ist die Stimme. Der Standardwert ist 50.

Sie finden das Beispiel auf der Buch-CD in der Datei BSP/K10/sprachausgabe.html.

Vertikale Sound-Position (elevation)

Mit der Eigenschaft `elevation` legen Sie die vertikale Position des Sounds in einer 3D-Sound-Arena fest.

CSS-Element: `elevation`	*Mögliche Werte:* Winkel, `below`, *level*, `above`, `higher`, `lower`, `inherit`
CSS-Version: CSS 2, CSS 3	*Zulässig für folgende (X)HTML-Elemente:* alle
Medium: Aural	*Vererbt:* Ja

Palm	NN	Mozilla		Internet Explorer							Opera			Safari		iCab	Kq	FF
2.x	4.x	1.x	1.8	4.x	5.0x (Win)	5.0 (Mac)	5.1+ (Mac)	5.5 (Win)	6	7	6	7	8	1.x	2.0	2.9+	3.x	1.0

Mögliche Werte

- Winkel: gibt eine Steigung als Winkel an, die zwischen -90 deg und 90 deg liegt. Die Höhe des Zuhörers ist definiert als 0 deg.

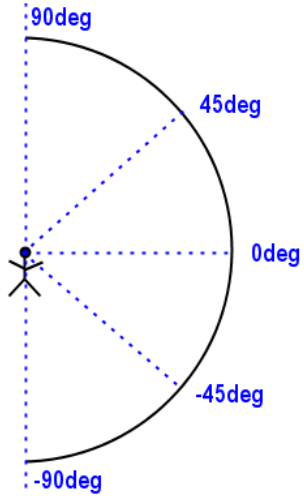


Abbildung 10.6: Darstellung des vertikalen Winkels für die Soundausgabe

- `below`: unterhalb, entspricht -90 deg
- `level`: entspricht 0 deg
- `above`: oberhalb, entspricht 90 deg
- `higher`: addiert 10 Grad zur aktuellen (vererbten) Position

Sie finden das Beispiel auf der Buch-CD in der Datei BSP/K10/sprachausgabe.html.

11 Sonstige Formatierungen, Pseudo-Elemente und -Klassen

Der wesentliche Teil dieses Kapitels beschäftigt sich mit Pseudo-Klassen und Pseudo-Elementen. Darüber hinaus werden noch einige CSS-Eigenschaften beschrieben, die sich nicht eindeutig speziellen Bereichen zuordnen lassen.

11.1 Einführung

Pseudo-Elemente und -Klassen

Das Konzept der Pseudo-Elemente und Pseudo-Klassen wurde bereits mit CSS 1.0 in Form der Pseudo-Klassen `:link`, `:visited` etc. eingeführt. Sie erlauben auch Formatierungen für solche Elemente festzulegen, die nicht durch eine Definition im Quellcode, sondern erst durch die Darstellung im Browser und Benutzeraktionen entstehen. Ob es sich um ein Pseudo-Element oder eine Pseudo-Klasse handelt, hängt davon ab, auf welche Weise die Elemente entstehen.

Pseudo-Elemente sind Elemente, die nicht über den (X)HTML-Dokumentenbaum entstehen, aber davon abgeleitet werden können. Ein gutes Beispiel dafür ist das Pseudo-Element `first-letter`*. Es gibt zwar keinen Tag* `<first-letter>`*, aber durch die Definition der Elementinhalte bestimmt der Dokumentenbaum, welches der erste Buchstabe eines Elements ist. Daher kann aus dem Dokumentenbaum der Inhalt eines Pseudo-Elements abgeleitet werden. Das trifft genauso auf die Pseudo-Elemente* `:before` *und* `:after` *zu. Aus der Dokumentstruktur ergibt sich nämlich auch, was vor und was nach einem Element ist. Folglich können Sie mit den Pseudo-Elementen* `:before` *und* `:after` *eine Position innerhalb des Dokumentenbaums relativ zu einem Element bestimmen. Pseudo-Elemente sind also:*

- `:first-line`
- `:first-letter`
- `:before`
- `:after`

Pseudo-Klassen entsprechen immer Elementen, die im Dokumentenbaum definiert sind. Allerdings werden diese klassifiziert und zwar nach bestimmten Eigenschaften. Dabei handelt es sich normalerweise um Eigenschaften, die sich nicht vom Dokumentenbaum ableiten lassen (z.B. ob ein Link bereits besucht wurde, gehovert wird oder den Fokus hat). Zu den CSS-Klassen gehören:

- `:first-child`
- `:link`
- `:visited`
- `:hover`
- `:active`
- `:focus`
- `:lang`

11.2 Praxistaugliche Formatierungen

Nachfolgend finden Sie CSS-Attribute, die Sie problemlos oder mit nur geringen Einschränkungen in der Praxis einsetzen können. Entweder ist die Browserunterstützung sehr gut oder aber ein Nichtausführen der CSS-Anweisung wirkt sich nicht so negativ aus, dass die Seite nicht mehr bedienbar ist.

Mauszeiger (cursor)

Mit der `cursor`-Eigenschaft können Sie bestimmen, wie der Mauszeiger aussehen soll. Als Wert können Sie einen oder mehrere Werte angeben, die Sie durch Komma trennen. Sie können beispielsweise zwei verschiedene Bild-URLs angeben und danach noch einen Standard-Cursor.

CSS-Element: `cursor`	*Mögliche Werte:* URL, *auto*, `crosshair`, `default`, `pointer`, `move`, `e-resize`, `ne-resize`, `nw-resize`, `n-resize`, `se-resize`, `sw-resize`, `s-resize`, `w-resize`, `text`, `wait`, `help`, `inherit`, `progress`[7]
CSS-Version: CSS 2, CSS 2.1, CSS 3	*Zulässig für folgende (X)HTML-Elemente:* alle
Medium: Visual, Interaktiv	*Vererbt:* Ja

Palm	NN	Mozilla		Internet Explorer							Opera			Safari		iCab	Kq	FF
2.x	4.x	1.x	1.8	4.x	5.0x (Win)	5.0 (Mac)	5.1+ (Mac)	5.5 (Win)	6	7	6	7	8	1.x	2.0	2.9+	3.x	1.0
		○ 1+3+6	○ 1+3	● 1+6	● 1+6	○ 4+6	○ 4+6	● 1+6	● 1+6	● 1		○ 2+3	○ 2+3		○ 4	○ 2+3+5	○ 4	○ 1+3

1 `crosshair` wird wie `text` dargestellt.
2 `crosshair` wird wie `default` dargestellt, bzw. nicht unterstützt.
3 ohne URL
4 URL und `crosshair` werden wie `text` dargestellt.
5 ab Version 3.0
6 Kein `progress`, bzw. `progress` wird wie `text` dargestellt.
7 nur in CSS 2.1

Für die Eigenschaft stehen folgende Werte zur Verfügung. ***Mögliche Werte***

- `auto`: Der Browser zeigt einen passenden Cursor an.
- `crosshair`: Der Cursor wird als Fadenkreuz angezeigt.
- `default`: Der Standardmauszeiger des Systems, in der Regel ein Pfeil.
- `pointer`: Ein Mauszeiger, der einen Hyperlink anzeigt.
- `move`: Ein Mauszeiger, der anzeigt, dass ein Element des Bildschirms verschoben wird.
- `e-resize`, `ne-resize`, `nw-resize`, `n-resize`, `se-resize`, `sw-resize`, `s-resize`, `w-resize`: Diese Werte zeigen einen Mauszeiger an, der eine Größenänderung kennzeichnet. Die Buchstaben `e`, `n`, `w` und `s` stehen für East (Osten), North (Norden), West (Westen), South (Süden) und kennzeichnen die Richtung des Mauszeigers.
- `text`: Der Mauszeiger ist ein Texteingabecursor. In der Regel ist das ein senkrechter Strich.
- `wait`: Ein Mauszeiger, der anzeigt, dass das System beschäftigt ist. Normalerweise wird dazu eine Sanduhr verwendet.
- `progress`: Ein Mauszeiger, der anzeigt, dass das System beschäftigt ist. Gemäß CSS-Standard soll `progress` einen anderen Mauszeiger anzeigen als `wait`.
- `help`: Ein Mauszeiger, der anzeigt, dass Hilfe verfügbar ist. Unter Windows ist dies in der Regel ein Fragezeichen.

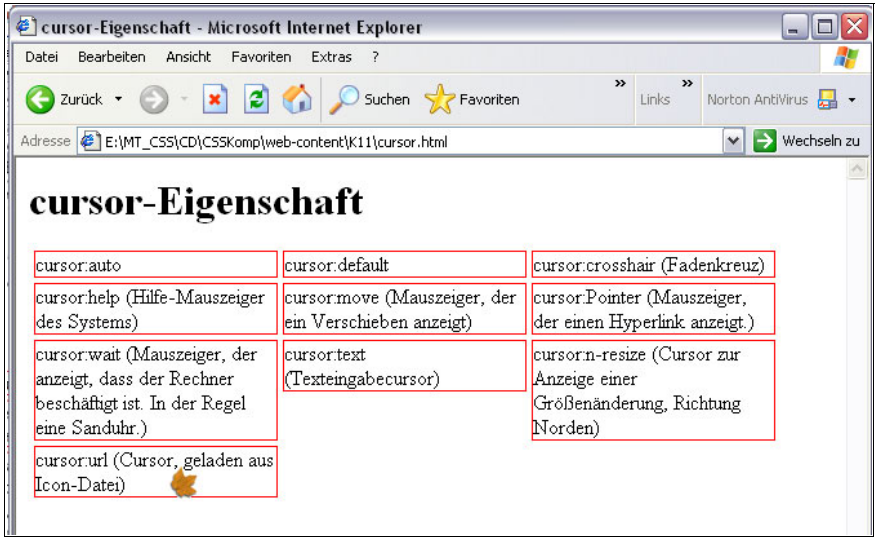
- URL: Gibt einen URL zu einer gültigen Grafikdatei an, aus der der Mauszeiger geladen wird. Das können Dateien in folgenden Formaten sein:
  - ANI: Animierte Cursor
  - ICO: Icon-Dateien
  - CUR: Cursor-Dateien

Sie finden das Beispiel auf der Buch-CD in der Datei BSP/K11/cursor.html.

Möchten Sie einen Cursor von einem URL laden, sieht der notwendige Stil dazu wie folgt aus:

```
<div style="cursor:url(mein.ico)">cursor:url (Cursor, geladen aus
Icon-Datei)</div>
```

Abbildung 11.1: Einzig der Internet Explorer 6+ für Windows lädt den Cursor aus dem angegebenen URL.



Pseudo-Klasse :active

Mithilfe der Pseudo-Klasse `:active` können Sie ein Element formatieren, das gerade aktiv ist. Was »aktiv« bedeutet, ist jedoch nicht eindeutig definiert. Die meisten Browser betrachten ein Element beispielsweise dann als aktiv, wenn mit der Maus darauf geklickt wird – bis die Maustaste wieder losgelassen wird.

Die Anwendung von `:active` *ist genau wie* `:hover` *nicht auf Hyperlinks beschränkt. Im Prinzip dürfen Sie die Klasse auf beliebige Elemente anwenden. Allerdings ist es derzeit so, dass die meisten Browser* `:hover` *nur für Links unterstützen.*

CSS-Element: `:active`	*Mögliche Werte:*
CSS-Version: CSS 1-3, TV, Mobile	*Zulässig für folgende (X)HTML-Elemente:* alle
Medium: alle	*Vererbt:*

Palm	NN	Mozilla		Internet Explorer							Opera			Safari		iCab	Kq	FF
2.x	4.x	1.x	1.8	4.x	5.0x (Win)	5.0 (Mac)	5.1+ (Mac)	5.5 (Win)	6	7	6	7	8	1.x	2.0	2.9+	3.x	1.0
		●	●	○ [1]	○ [1]	○ [1]	○ [1]	○ [1]	○ [1]	○ [1]		●	●	●	●	● [2]	●	●

[1] nur für Hyperlinks
[2] ab Version 3.0

Sie finden das Beispiel auf der Buch-CD in der Datei BSP/K11/link.html.

Beispiel

Der folgende Stil färbt mit der Klasse `.hovern` alle Elemente rot ein, solange diese angeklickt werden und die Maustaste festgehalten wird.

```
.hovern:active { background-color:red; color:white }
```

Die Pseudo-Klassen `:hover`, `:active` *und* `:focus` *schließen sich gegenseitig nicht aus. Ein Element kann also gleichzeitig mit den Pseudo-Klassen* `:hover` *und* `:active` *formatiert werden. Das sollten Sie bedenken, wenn Sie die Formatierungen für die Stile festlegen.*

Pseudo-Element :after

Das Pseudo-Element `:after` definiert Inhalt, der hinter einem Element angezeigt wird. Diesen Inhalt definieren Sie mit der `content`-Eigenschaft.

Mehr zur `content`*-Eigenschaft finden Sie in Kapitel 6, »Listen und Aufzählungen«.*

Solange der mit `:after` *erzeugte Inhalt Bestandteil der ersten Zeile ist, kann er mit dem Pseudo-Element* `:first-line` *formatiert werden.*

CSS-Element: `:after`	*Mögliche Werte:*
CSS-Version: CSS 2-3	*Zulässig für folgende (X)HTML-Elemente:* alle
Medium: alle	*Vererbt:*

Palm	NN	Mozilla		Internet Explorer							Opera			Safari		iCab	Kq	FF
2.x	4.x	1.x	1.8	4.x	5.0x (Win)	5.0 (Mac)	5.1+ (Mac)	5.5 (Win)	6	7	6	7	8	1.x	2.0	2.9+	3.x	1.0
		●[1]	●									⊙	⊙		⊙	●[2]	⊙	●

[1] ab Version 1.4
[2] ab Version 3.0

Sie finden das Beispiel auf der Buch-CD in der Datei BSP/K11/before_after.html.

Erläuterungen zum Beispiel finden Sie im Abschnitt »Pseudo-Element :before«.

Pseudo-Element :before

Das Pseudo-Element `:before` definiert Inhalt, der vor einem Element angezeigt wird. Diesen Inhalt definieren Sie mit der `content`-Eigenschaft.

Mehr zur content*-Eigenschaft finden Sie in Kapitel 6, »Listen und Aufzählungen«.*

Fügen Sie mit dem Pseudo-Element ein oder mehrere Zeichen vor einem Element ein, kann das erste Zeichen über das Pseudo-Element :first-letter *formatiert werden.* :first-letter *formatiert dann also nicht mehr das erste Zeichen des definierten Textes.*

CSS-Element: :before	*Mögliche Werte:*
CSS-Version: CSS 2-3	*Zulässig für folgende (X)HTML-Elemente:* alle
Medium: alle	*Vererbt:*

Palm	NN	Mozilla		Internet Explorer							Opera			Safari		iCab	Kq	FF
2.x	4.x	1.x	1.8	4.x	5.0x (Win)	5.0 (Mac)	5.1+ (Mac)	5.5 (Win)	6	7	6	7	8	1.x	2.0	2.9+	3.x	1.0
		●[1]	●									⊙	⊙		⊙	●[2]	⊙	●

[1] ab Version 1.4
[2] ab Version 3.0

Beispiel

Auch mit dem :before-Element können Sie, wie mit :first-letter, Initialen erzeugen; allerdings mit dem Unterschied, dass Sie das Initial dazu erst erzeugen müssen. Sie geben dazu das Zeichen mit der content-Eigenschaft an, und können es über weitere CSS-Eigenschaften formatieren.

Sie finden das Beispiel auf der Buch-CD in der Datei BSP/K11/before_after.html.

Folgende Stile fügen vor und nach einem Absatz ein Anführungszeichen ein und formatieren es in Blau und einer größeren Schrift.

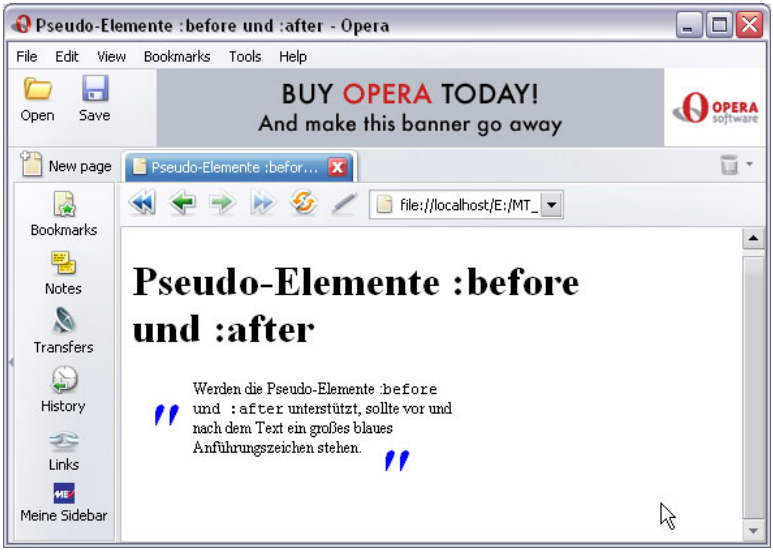
```
p:before {
   content:'"';
   font-weight:bold;
   color:blue;
   font-style:italic;
   font-size:4em;
   float:left;
   margin:10px; margin-bottom:0;
}
```

```
p:after {
   content:'"';
   font-weight:bold;
   color:blue;
   font-style:italic;
   font-size:4em;
   float:none;
   vertical-align:top;
   margin:10px;margin-bottom:0
}
```

Abbildung 11.2: Korrekt formatiert sollte die Beispielseite so aussehen.

Der Opera-Browser und Safari fügen den Text zwar ein, wenden aber nicht alle Formatierungen korrekt an. Beispielsweise wird die `vertical-align`-Eigenschaft nicht auf den mit `:before` erzeugten Inhalt angewendet.

Abbildung 11.3: Opera und Safari richten das mit :before erzeugte Anführungszeichen nicht korrekt aus.



Pseudo-Klasse :first-child

Die Pseudo-Klasse `:first-child` identifiziert Elemente, die das erste Element eines übergeordneten Elements sind.

CSS-Element: `:first-child`	*Mögliche Werte:*
CSS-Version: CSS 2, 2.1, 3, TV	*Zulässig für folgende (X)HTML-Elemente:* alle
Medium: alle	*Vererbt:*

Palm	NN	Mozilla		Internet Explorer							Opera			Safari		iCab	Kq	FF
2.x	4.x	1.x	1.8	4.x	5.0x (Win)	5.0 (Mac)	5.1+ (Mac)	5.5 (Win)	6	7	6	7	8	1.x	2.0	2.9+	3.x	1.0
			●				● [1]					● [2]	●		●	● [3]	●	●

1 ab Version 5.2
2 mindestens ab Version 7.4x
3 ab Version 3.0

Beispiel

Listing 11.1: Der Beispielcode

```
<head>
    <title>Pseudo-Klasse :first-child</title>
    <style type="text/css">
    <!--
        p                 { color:blue }
        p:first-child     { color:red }
        p span:first-child { text-decoration: underline; color:lime }
        code:first-child  { font-weight:bold;
                            text-decoration: underline }
    -->
    </style>
</head>
<body>
  <h1>Pseudo-Klasse :first-child</h1>
  <div>
    <p>Wird die Pseudo-Klasse <code>:first-child</code> unterst&uuml;tzt,
       sollten alle Abs&auml;tze blau formatiert sein, nur der erste
       nicht.</p>
    <p>Dies ist der zweite Absatz der Seite.</p>
    <p><span>Dieser Text wird gr&uuml;n</span> formatiert, wenn der
       Browser <code>:first-child</code> unterst&uuml;tzt.</p>
```

```
    <p>Außerdem werden <code>code</code>-Elemente, die als erstes Kind in
       einem &uuml;bergeordneten Element liegen, fett formatiert und
       unterstrichen, wenn der Browser <code>:first-child</code>
       unterst&uuml;tzt.</p>
  </div>
</body>
```

Sie finden das Beispiel auf der Buch-CD in der Datei BSP/K11/first_child.html.

Pseudo-Element :first-letter

Mit dem Pseudo-Element `:first-letter` können Sie auf das erste Zeichen eines Elements zugreifen. Allerdings sind nicht alle CSS-Eigenschaften für einen Stil mit `:first-letter` zulässig. Nur die folgenden Eigenschaften dürfen Sie verwenden:

- Schrifteigenschaften
- Vordergrundfarbe
- Hintergrundeigenschaften
- `text-decoration`
- `vertical-align`, falls `float:none` gesetzt ist
- `text-transform`
- `line-height`
- Außen- und Innenabstände
- Rahmeneigenschaften
- `float`
- `text-shadow`
- `clear`

CSS-Element: `:first-letter`	*Mögliche Werte:*
C*SS-Version:* CSS 1-3, TV	*Zulässig für folgende (X)HTML-Elemente:* alle
Medium: alle	*Vererbt:*

Palm	NN	Mozilla		Internet Explorer							Opera			Safari		iCab	Kq	FF
2.x	4.x	1.x	1.8	4.x	5.0x (Win)	5.0 (Mac)	5.1+ (Mac)	5.5 (Win)	6	7	6	7	8	1.x	2.0	2.9	3.x	1.0
		●[1]	●			●	●				●	●	●		●	●[2]	●	●

[1] ab Version 1.4+
[2] ab Version 3.0

Sie finden das Beispiel auf der Buch-CD in der Datei BSP/K11/first_letter.html.

Beispiel

Mit Hilfe des Pseudo-Elements `:first-letter` können Sie Initialen erstellen und formatieren, indem Sie das erste Zeichen in einer größeren Schrift formatieren und den folgenden Text bei Bedarf mit `float:left` um das erste Zeichen fließen lassen. Dazu legen Sie zunächst die allgemeinen Schrifteinstellungen fest und anschließend die Eigenschaften für das erste Zeichen.

```
p {
  color:black;
  font-size:12px;width:150px
}
p:first-letter {
  font-weight:bold;
  color:blue;
  font-style:italic;
  font-size:4em;
  float:left;
  margin:10px;margin-bottom:0
}
```

Beachten Sie, dass die Auswahl des ersten Buchstabens auch sprachabhängig ist. In bestimmten Sprachen gelten gewisse Buchstabenkombinationen als ein Zeichen mit besonderer Aussprache. Diese Zeichenkombinationen sollten dann von den Browsern als ein Zeichen gewertet werden.

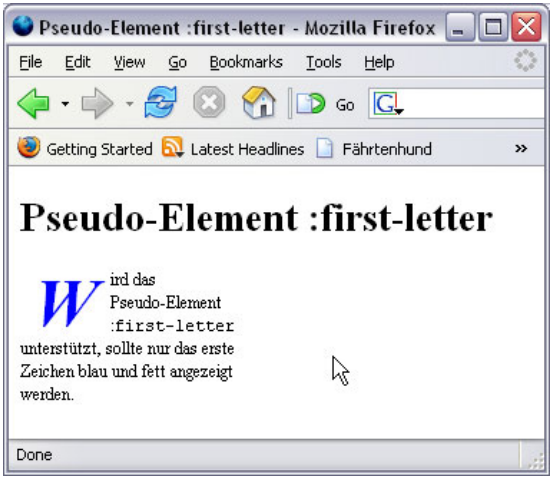


Abbildung 11.4: So sollten die beiden Stile der Beispielseite angewendet werden.

Pseudo-Element :first-line

Mit dem Pseudo-Element `:first-line` können Sie auf die erste Zeile eines Elements zugreifen. Wie viele Wörter diese Zeile umfasst, ermittelt der Browser anhand des erfolgten Zeilenumbruchs automatisch.

INFO

Der Inhalt der ersten Zeile wird vom Browser wie ein Inline-Element behandelt. Allerdings gibt es da diverse Einschränkungen. Sie können für die so selektierten Teile des Blockelements nur folgende Formatierungen festlegen:

- *Schrifteigenschaften und Textformatierungen*
- *Vordergrundfarbe*
- *Hintergrundeinstellungen*
- `clear`-*Eigenschaft*

Alle anderen Eigenschaften sollten vom Browser ignoriert werden.

CSS-Element: `:first-line`	*Mögliche Werte:*
CSS-Version: CSS 1-3, TV	*Zulässig für folgende (X)HTML-Elemente:* alle Blockelemente
Medium: alle	*Vererbt:*

Palm	NN	Mozilla		Internet Explorer							Opera			Safari		iCab	Kq	FF
2.x	4.x	1.x	1.8	4.x	5.0x (Win)	5.0 (Mac)	5.1+ (Mac)	5.5 (Win)	6	7	6	7	8	1.x	2.0	2.9+	3.x	1.0
		● [1]	●			●	●					● [2]	●		●	● [3]	●	●

1 ab Version 1.4+
2 mindestens ab Version 7.4+
3 ab Version 3.0

Beispiel

Mit dem folgenden Code können Sie festlegen, dass die erste Zeile eines jeden Absatzes in blauer Schrift und fett formatiert werden soll.

```
p:first-line { font-weight:bold; color:blue; }
```

Sie finden das Beispiel auf der Buch-CD in der Datei BSP/K11/first_line.html.

Abbildung 11.5: So sieht eine korrekte Darstellung des Beispiels aus (hier im Opera-Browser).

Pseudo-Klasse :focus

Die Pseudo-Klasse `:focus` ermöglicht es Ihnen, Elemente zu formatieren, die aktuell den Fokus haben, also Tastatureingaben etc. entgegennehmen können. Generell können Sie die Klasse auf beliebige Elemente anwenden. Besonders sinnvoll ist der Einsatz aber im Zusammenhang mit Formularfeldern.

CSS-Element: `:focus`	*Mögliche Werte:*
CSS-Version: CSS 2-3, TV, Mobile	*Zulässig für folgende (X)HTML-Elemente:* alle
Medium: alle	*Vererbt:*

Palm	NN	Mozilla		Internet Explorer							Opera			Safari		iCab	Kq	FF
2.x	4.x	1.x	1.8	4.x	5.0x (Win)	5.0 (Mac)	5.1+ (Mac)	5.5 (Win)	6	7	6	7	8	1.x	2.0	2.9	3.x	1.0
		●	●			○ 1	○ 1			5		○ 1	○ 1	○ 1+2	○ 1+2	○ 1+2+4	● 1+2	● 3

1 nur für Formularfelder
2 nicht für Formular-Schaltflächen
3 auf dem Mac nur Formularfelder, keine Schaltflächen und keine Links
4 ab Version 3.0 auch für Links
5 Einem Link, der den Fokus hat, wird die Pseudo-Klasse `:active` zugewiesen.

Sie finden das Beispiel auf der Buch-CD in der Datei BSP/K11/focus.html.

Beispiel

Wenn Sie alle Formularfelder, die den Fokus haben, mit einem rosa Hintergrund und einem roten Rahmen versehen möchten, könnten die dazu notwendigen Stile wie folgt lauten:

```
input:focus { background-color:rgb(255,125,125);
              border:1px solid red }
input[type="submit"]:focus {
             background-color:red; border:none }
```

Der erste Stil sorgt dafür, dass für alle `input`-Elemente der Rahmen und die Hintergrundfarbe gesetzt werden, wenn sie den Fokus haben. Mit dem zweiten Stil machen Sie die Rahmendefinition rückgängig, wenn das Element über das Attribut `type` mit dem Wert `submit` verfügt, weil es dann ein Button ist.

Nur Browser, die auch die Attribut-Selektoren unterstützen, stellen den zweiten Stil dar. Der Internet Explorer für Mac zeigt beispielsweise auch um den Submit-Button einen Rahmen an, weil er den Attribut-Selektor nicht beherrscht.

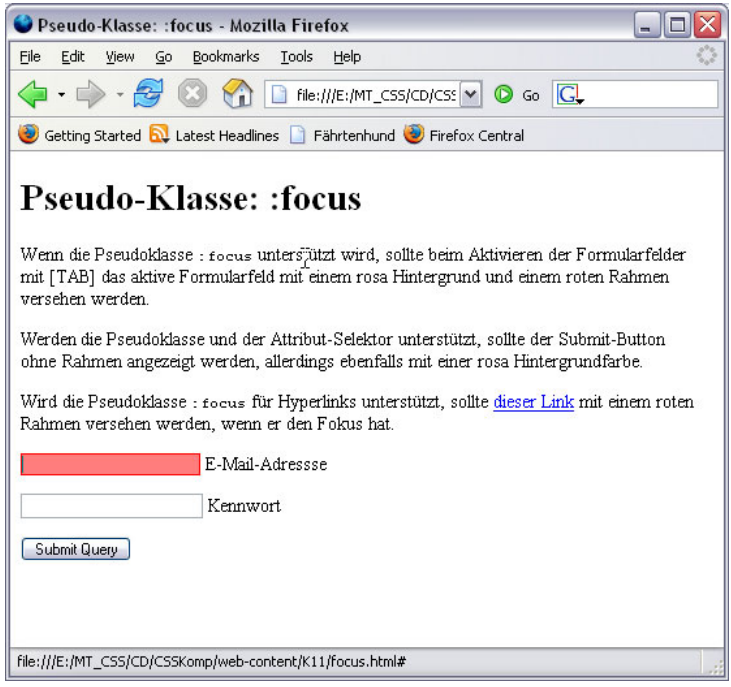


Abbildung 11.6: Korrekte Darstellung der Beispielseite in Firefox für Windows. Hier hat das obere Eingabefeld den Fokus.

Bei Opera-Browsern können Sie den Fokus für Hyperlinks nur über Zugriffstasten aktivieren. Sie müssen dazu den Link um das Attribut `accesskey` ergänzen und als Wert die gewünschte Taste angeben. Mit `<a href="#" accesskey="A">dieser Link</a>` würden Sie also einen Hyperlink definieren, der über die Taste A den Fokus bekommt. Formularfelder können Sie hingegen mit der ⇆-Taste anspringen und ihnen so den Fokus geben.

Auch iCab benötigt eine Zugriffstaste, damit Hyperlinks den Fokus bekommen können. Die Zugriffstaste müssen Sie dann zusammen mit Alt drücken. iCab zeigt einen definierten Accesskey in spitzen Klammern (ab Version 3.0 in eckigen Klammern) hinter dem Link an. Empfinden Sie das als störend, müssen Sie auf das `accesskey`-Attribut verzichten; dann können Links im Opera-Browser und iCab aber nicht den Fokus bekommen. iCab unterstützt die Pseudo-Klasse `:focus` auch für Hyperlinks und ist der einzige Browser, der um die Eingabefelder auch den Rahmen anzeigt.

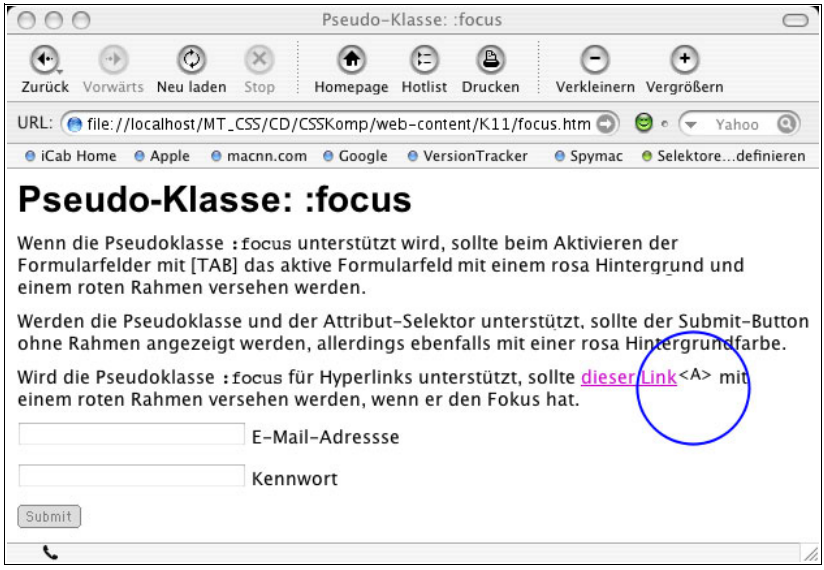


Abbildung 11.7: Anzeige der Zugriffstaste in iCab 2.9.x

Und noch eine Besonderheit gibt es in iCab 3.0. Offenbar kennt iCab zwei verschiedene Zustände, die dem »Fokus« eines Elements ähnlich sind. Ein Formularfeld, das mit der ⇆-Taste aktiviert wurde, bekommt zunächst den Standard-Markierungsrahmen von Mac OS X. Erst nach Anklicken des Formularfeldes wird dann die CSS-Klasse angewendet, zumindest die Rahmeneinstellungen. Die Hintergrundfarbe wird nicht angewendet.

Bei einem erneuten Drücken der ⇆-Taste erhält dann das nächste Formularfeld gemäß Mac OS X den Fokus, da es mit dem hellblauen Standardrahmen formatiert wurde. Gleichzeitig bleibt aber der Stil mit der Pseudo-Klasse `:focus` dem ersten Eingabefeld zugewiesen, bis erneut ein Element oder ein anderer Teil des Fensters angeklickt wird. Der Fensterteil, der also gemäß iCab den Fokus hat, muss dies nicht auch aus Sicht des Betriebssystems haben. Dies kann (muss aber nicht) noch ein Fehler der BETA-Version sein.

Mac +

Die Firefox-Version für den Mac und Safari ermöglichen ebenfalls keine Fokussierung von Hyperlinks (auch nicht über Zugriffstasten) und Formularschaltflächen über die Tastatur. Deshalb wird für diese Elemente die Pseudo-Klasse `:focus` auch nicht angewendet. Anwendbar ist `:focus` damit nur auf Eingabefelder in Formularen, und dort kommt lediglich die Hintergrundfarbe zum Einsatz. Rahmeneigenschaften werden verschluckt zugunsten des Standardrahmens von Mac OS X für fokussierte Elemente.

Pseudo-Klasse :hover

Mit Hilfe der Pseudo-Klasse `:hover` können Sie ein Element formatieren, über das der Benutzer mit der Maus fährt, ohne es jedoch anzuklicken.

Die Anwendung von `:hover` *ist allerdings nicht auf Hyperlinks beschränkt. Im Prinzip dürfen Sie die Klasse auf beliebige Elemente anwenden. Allerdings ist es derzeit so, dass die meisten Browser* `:hover` *nur für Links unterstützten.*

CSS-Element: `:hover`	*Mögliche Werte:*
CSS-Version: CSS 1-3	*Zulässig für folgende (X)HTML-Elemente:* alle
Medium: alle	*Vererbt:*

Palm	NN	Mozilla		Internet Explorer							Opera			Safari		iCab	Kq	FF
2.x	4.x	1.x	1.8	4.x	5.0x (Win)	5.0 (Mac)	5.1+ (Mac)	5.5 (Win)	6	7	6	7	8	1.x	2.0	2.9	3.x	1.0
		●	●	○ [1]	○ [1]	○ [1]	○ [1]	○ [1]	○ [1]	○ [1]		●	●	●	●	● [2]	●	●

[1] nur für Hyperlinks
[2] ab Version 3.0

Sie finden das Beispiel auf der Buch-CD in der Datei BSP/K11/link.html.

Beispiel

Mit Hilfe der Pseudo-Klasse `:hover` können Sie beispielsweise Hover-Buttons erstellen, ohne dass Sie ein `img`-Element benötigen. Sie definieren dazu einfach ein `div`-Element:

```
<div> </div>
```

Für dieses `div`-Element legen Sie dann über einen Elementstil oder einen Klassenstil Größe und Hintergrundbild fest:

```
div { width:50px; height:50px;
      background-image:url(grau.jpg);
      border:1px solid black }
```

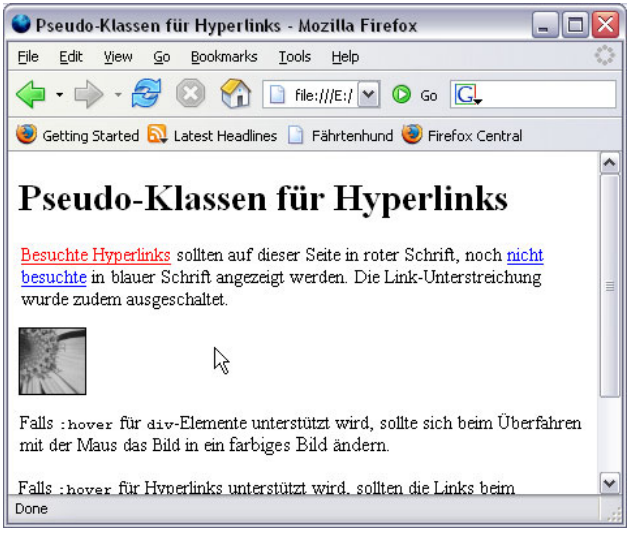
In einem zweiten Stil mit der Pseudo-Klasse `:hover` definieren Sie dann einfach ein anderes Hintergrundbild und einen stärkeren Rahmen:

```
div:hover { background-image:url(farbig.jpg);
            border:3px solid red }
```

!! STOP

Die Pseudo-Klassen `:hover`*,* `:active` *und* `:focus` *schließen sich gegenseitig nicht aus. Ein Element kann also gleichzeitig mit den Pseudo-Klassen* `:hover` *und* `:active` *formatiert werden. Das sollten Sie bedenken, wenn Sie die Formatierungen für die Stile festlegen.*

Abbildung 11.8: Das »Bild« in normalem Zustand in Schwarz-Weiß ...



Pseudo-Klasse :link

Mit der Pseudo-Klasse `:link` können Sie Links formatieren, die noch nicht besucht wurden. Ob ein Link besucht wurde, bestimmen die meisten Browser daran, ob die Zielseite bereits im Browsercache vorhanden ist. Sie können jedoch gemäß HTML-Standard den Status eines Links nach einer Weile zurücksetzen.

REF

Analog dazu verwenden Sie die Pseudo-Klasse `:visited`*, um die besuchten Links zu formatieren. Mehr zu dieser Pseudo-Klasse finden Sie im Abschnitt »Pseudo-Klasse :visited«.*

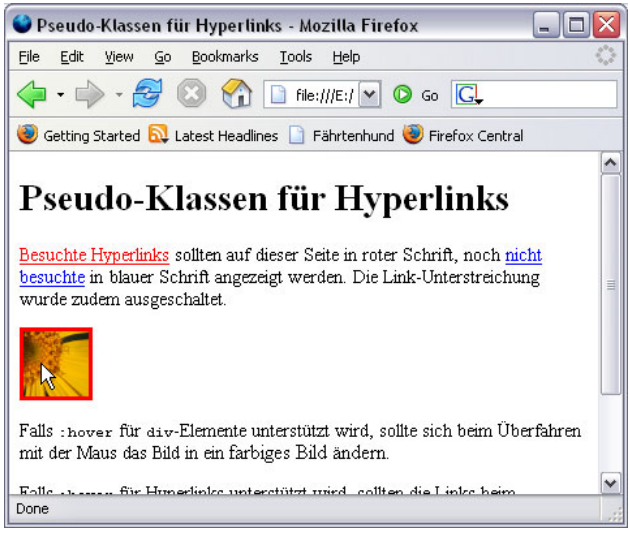


Abbildung 11.9: ... und gehovert in Farbe und mit rotem Rahmen

CSS-Element: `:link`	*Mögliche Werte:*
CSS-Version: CSS 1, 2, 2.1, 3, TV, Mobile	*Zulässig für folgende (X)HTML-Elemente:* Hyperlinks (`a`-Elemente mit `href`-Attribut)
Medium: alle	*Vererbt:*

Palm	NN	Mozilla		Internet Explorer							Opera			Safari		iCab	Kq	FF
2.x	4.x	1.x	1.8	4.x	5.0x (Win)	5.0 (Mac)	5.1+ (Mac)	5.5 (Win)	6	7	6	7	8	1.x	2.0	2.9+	3.x	1.0
●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●

Sie finden das Beispiel auf der Buch-CD in der Datei BSP/K11/link.html.

Beispiel

Mit den folgenden beiden Stilen definieren Sie unterschiedliche Textfarben für normale und besuchte Hyperlinks.

```
a:link    { color:blue }
a:visited { color:red }
```

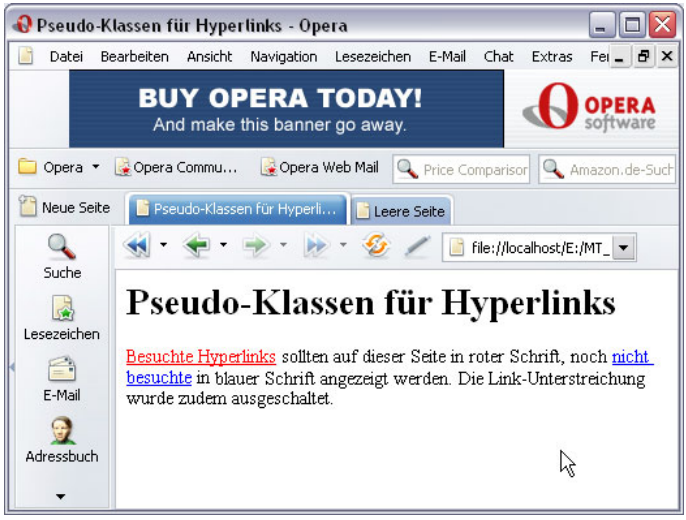


Abbildung 11.10: So sollte die Beispielseite dargestellt werden, vorausgesetzt der erste Link wurde schon einmal angeklickt, der zweite nicht.

Pseudo-Klasse :visited

Mit der Pseudo-Klasse `:visited` können Sie die besuchten Hyperlinks formatieren. Auch diese Pseudo-Klasse können Sie nur auf Hyperlinks, also `a`-Elemente anwenden.

CSS-Element: `:visited`	*Mögliche Werte:*
CSS-Version: CSS 1, 2, 2.1, 3, TV, Mobile	*Zulässig für folgende (X)HTML-Elemente:* alle `a`-Elemente mit `href`-Attribut
Medium: alle	*Vererbt:*

Palm	NN	Mozilla		Internet Explorer							Opera			Safari		iCab	Kq	FF
2.x	4.x	1.x	1.8	4.x	5.0x (Win)	5.0 (Mac)	5.1+ (Mac)	5.5 (Win)	6	7	6	7	8	1.x	2.0	2.9+	3.x	1.0
●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●

Sie finden das Beispiel auf der Buch-CD in der Datei BSP/K11/link.html.

11.3 Nicht/schlecht unterstützte Formatierungen

In dieser Rubrik finden Sie solche Attribute, die von vielen oder zumindest von sehr wichtigen Browsern nicht oder fehlerhaft unterstützt werden und die somit noch nicht praxistauglich sind.

Pseudo-Klasse :lang

Mit der Pseudo-Klasse `:lang` können Sie Formatierungen abhängig von der Sprache festlegen. Im Unterschied zum Einsatz des Attribut-Selektors, beispielsweise `body[lang="de"]`, müssen Sie dabei nicht wissen, mit welchem Attribut bzw. auf welche Weise die Sprache definiert wurde. Das kann nämlich abhängig von der eingesetzten Auszeichnungssprache (XHTML bzw. HTML) unterschiedlich geschehen.

An die Pseudo-Klasse übergeben Sie den Namen der Sprache als Parameter und setzen diesen in runde Klammern.

INFO

CSS-Element: `:lang(`*Sprache*`)`	*Mögliche Werte:* Kürzel für die Sprache, beispielsweise `de` (Deutsch), `en` (Englisch) etc.
CSS-Version: CSS 2-3	*Zulässig für folgende (X)HTML-Elemente:* alle
Medium: alle	*Vererbt:* Nein

Palm	NN	Mozilla		Internet Explorer							Opera			Safari		iCab	Kq	FF
2.x	4.x	1.x	1.8	4.x	5.0x (Win)	5.0 (Mac)	5.1+ (Mac)	5.5 (Win)	6	7	6	7	8	1.x	2.0	2.9+	3.x	1.0
		●[1]	●				●					●	●			●[2]		●

[1] ab Version 1.4+
[2] ab Version 3.0

Beispiel

Mit folgenden Stilen werden alle deutschen Elemente der Seite rot und fett, alle englischen blau und in einer kleineren Schrift formatiert. Da sie auf den Universal-Selektor angewendet werden, betreffen die Formatierungen beliebige Elemente der Seite.

```
:lang(en) { color:blue; font-size:0.9em }
:lang(de) { color:red; font-weight:bold }
```

Abbildung 11.11: So sollte die Beispielseite korrekt dargestellt werden.

Pseudo-Element :not

Das Pseudo-Element `:not` bietet die Möglichkeit, nach Elementen zu selektieren, die ein bestimmtes Attribut nicht haben. Sie können es beispielsweise nutzen, um Bilder ohne `alt`-Attribut zu finden oder alle Elemente ohne `class`-Attribut zu identifizieren.

CSS-Element: `:not([`*Attributname*`])`	*Mögliche Werte:* Attributnamen
CSS-Version: CSS 3	*Zulässig für folgende (X)HTML-Elemente:* alle
Medium:	*Vererbt:*

Palm	NN	Mozilla		Internet Explorer							Opera			Safari		iCab	Kq	FF
2.x	4.x	1.x	1.8	4.x	5.0x (Win)	5.0 (Mac)	5.1+ (Mac)	5.5 (Win)	6	7	6	7	8	1.x	2.0	2.9	3.x	1.0
		●[1]	●												●		●	●

[1] ab Version 1.4+, Netscape 7.1

Sie finden das Beispiel auf der Buch-CD in der Datei BSP/K11/not.html.

Mit dem folgenden Stil können Sie beispielsweise dafür sorgen, dass alle Absätze, die nicht über ein `class`-Attribut verfügen, mit roter, fetter Schrift formatiert werden.

```
p:not([class]) { color:red; font-weight:bold }
```

Abbildung 11.12: So sollte die Beispielseite dargestellt werden.

Schreib-/Leserichtung (direction)

Die Eigenschaft `direction` gibt an, in welcher Richtung Text gelesen und geschrieben wird, also ob von rechts nach links oder von links nach rechts. Sie wirkt sich aber nicht nur auf die Anordnung von Text aus, sondern auch auf die Positionierung und den Überlauf von Elementen sowie die Anordnung von Tabellenspalten. Sie ist also nicht speziell eine Eigenschaft zur Textformatierung.

INFO

Beachten Sie, dass Browser den Wert der Eigenschaft missachten dürfen, weil auch HTML 4 die Angabe der Leserichtung ermöglicht. Diese geht dann den CSS-Formatierungen vor.

Die Schreib-/Leserichtung bewirkt auch nicht, dass die Reihenfolge der Wörter oder Buchstaben geändert wird, sondern lediglich die Ausrichtung von Text und enthaltenen Elementen.

CSS-Element: `direction`	*Mögliche Werte: ltr*, `rtl`, `inherit`
CSS-Version: CSS 2-3	*Zulässig für folgende (X)HTML-Elemente:* alle
Medium: Visual	*Vererbt:* Ja

Palm	NN	Mozilla		Internet Explorer							Opera			Safari		iCab	Kq	FF
2.x	4.x	1.x	1.8	4.x	5.0x (Win)	5.0 (Mac)	5.1+ (Mac)	5.5 (Win)	6	7	6	7	8	1.x	2.0	2.9+	3.x	1.0
		● [1]	● [2]						●	●		●	●		●	⊙ [2]	●	● [2]

1 ab Version 1.4+
2 ab Version 3.0 teilweise und fehlerhaft

Mögliche Werte

- `ltr`: ist die Abkürzung für »left to right« und bewirkt folglich, dass von links nach rechts geschrieben wird.
- `rtl`: steht für »right to left« und bewirkt eine Schreibrichtung von rechts nach links.

INFO

Für Inline-Elemente wird der Wert nur berücksichtigt, wenn die `unicode-bidi`*-Eigenschaft einen der Werte* `embed` *oder* `override` *hat. Mehr zu diesen Werten finden Sie im Abschnitt »Unicode-bidirektionaler Modus (unicode-bidi)«.*

CD

Sie finden das Beispiel auf der Buch-CD in der Datei BSP/K11/direction.html.

Beispiel

Im folgenden Beispiel werden zwei Absätze mit einer fixen Breite von 250 Pixel definiert. Der obere wird mit `direction:ltr`, der zweite mit `direction:rtl` formatiert. Für beide Absätze wurde `text-align:justify` ausgewählt.

Listing 11.2: Code der Beispielseite

```
</head>
...
    <style type="text/css">
    <!--
        .vergleich       { color:black; font-size:12px;
                           width:250px; text-align:justify;
                           direction:ltr }
        .rechtsnachlinks { color:black; font-size:12px;
                           width:250px; text-align:justify;
                           direction:rtl }
     -->
```

```
    </style>
</head>
<body>
    <h1>direction- und unicode-bidi-Eigenschaft</h1>
    <p class="vergleich">Wenn die direction-Eigen ...</p>
    <p class="rechtsnachlinks">Wenn die direction-... </p>
</body>
```

Bei den auf Mozilla-Gecko basierenden Browsern gibt es Unterschiede in der Darstellung, die offenbar abhängig von der Version sind. Während ältere Mozilla-Browser (kleiner als Mozilla 1.8) nur den Text innerhalb des Absatzes entsprechend der `direction`-Eigenschaft ausrichten, richten die neueren Browser, wie Firefox (Windows und Mac) sowie Mozilla 1.8, die Absätze auch innerhalb der umgebenden Box entsprechend rechts- bzw. linksbündig aus.

Abbildung 11.13: Mozilla 1.8+ richtet die Absätze auch in der umgebenden Box unterschiedlich aus.

Der Opera-Browser, Safari und der Internet Explorer 6/7 verfahren jedoch wie die älteren Mozilla-Browser.

iCab ab der Version 3.0 richtet die mit `direction:rtl` formatierten Blockelemente rechts innerhalb des umgebenden Elements aus, wie dies auch Mozilla 1.8+ macht. Die Schreibrichtung wird jedoch nur bei Elementen mit Blocksatz deutlich. Da iCab noch keinen Blocksatz unterstützt, lässt sich nicht sagen, ob die `direction`-Eigenschaft sonst noch Auswirkungen hat.

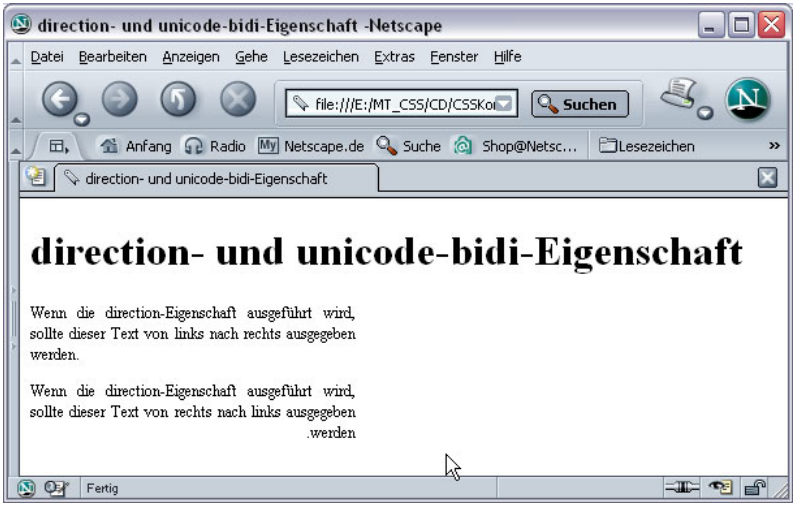


Abbildung 11.14: Mozilla-Browser (hier Netscape 7.1) sowie der Opera-Browser, IE und Safari verwenden den Wert von direction nur für den Inhalt des Absatzes.

Unicode-bidirektionaler Modus (unicode-bidi)

Es gibt Situationen, in denen es denkbar ist, dass verschiedene Schreibrichtungen innerhalb einer Seite und sogar innerhalb eines Elements verwendet werden. Dieser Modus wird als bidirektionaler Modus bezeichnet. Das Verhalten des Browsers in diesem Modus können Sie mit der Eigenschaft `unicode-bidi` einstellen.

CSS-Element: `unicode-bidi`	*Mögliche Werte: normal*, `embed`, `bidi-override`, `inherit`
CSS-Version: CSS 2-3	*Zulässig für folgende (X)HTML-Elemente:* alle
Medium: Visual	*Vererbt:* Nein

Palm	NN	Mozilla		Internet Explorer							Opera			Safari		iCab	Kq	FF
2.x	4.x	1.x	1.8	4.x	5.0x (Win)	5.0 (Mac)	5.1+ (Mac)	5.5 (Win)	6	7	6	7	8	1.x	2.0	2.9+	3.x	1.0

Mögliche Werte

- `normal`: Die Leserichtung wird durch den Wert der `direction`-Eigenschaft bestimmt.

Erläuterungen zur `direction`*-Eigenschaft finden Sie im Abschnitt »Schreib-/Leserichtung (direction)«.*

- `embed`: Das Inline-Element, das mit `unicode-bidi:embed` formatiert ist, wird in der angegebenen Leserichtung angezeigt, während das umgebende Element seine Leserichtung beibehält. Günstig ist das vor allem bei Zitaten in Sprachen, die eine andere Leserichtung haben.
- `bidi-override`: Die mit `direction` angegebene Leserichtung ersetzt die Leserichtungen der anderen Elemente.

Sie finden das Beispiel auf der Buch-CD in der Datei BSP/K11/direction.html.

Mit dem Stil:

```
.cite { direction:rtl; unicode-bidi:embed; }
```

werden alle Zitate von rechts nach links geschrieben und so in einen linksbündigen Kontext eingebettet.

11.4 Praxisbeispiel

Mit Hilfe von Pseudo-Elementen und Pseudo-Klassen lassen sich fantastische Stylesheets kreieren, die nicht nur die Fehlersuche erleichtern können, sondern auch die Formatierung mit CSS wesentlich vereinfachen. Das folgende, zugegebenermaßen sehr kurze Praxisbeispiel zeigt einige dieser Möglichkeiten.

Sie finden das Beispiel auf der Buch-CD in der Datei BSP/K11/praxisbeispiel.html.

Anker kenntlich machen

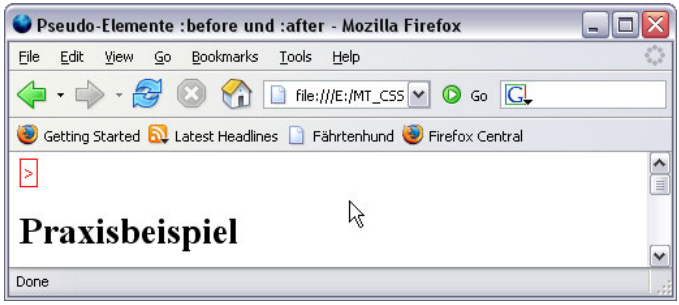
Anker zeichnen sich dadurch aus, dass es sich um `a`-Elemente handelt, denen das `href`-Attribut fehlt. Zudem sind sie unsichtbar, da sie in der Regel nicht über einen darstellbaren Inhalt verfügen. Dennoch kann es unter Umständen sinnvoll sein, sie sichtbar zu machen. Dabei hilft folgender Stil:

```
a:not([href]):before { content:">";
                       border:1px solid red;
                       color:red; padding:2px }
```

Er selektiert zunächst mit `a:not([href])` alle `a`-Elemente ohne `href`-Attribut. Mit der Pseudo-Klasse `:before` wird dann vor dem Element ein »>« in

roter Schrift und mit einem roten Rahmen eingefügt. Damit wird der Anker sichtbar.

Abbildung 11.15: Das Ergebnis in Firefox



Elemente mit fehlenden Attributen finden

Mit dem Pseudo-Element `:not` können Sie auch sehr schön fehlerhaften HTML-Code ausfindig machen, indem Sie alle Elemente markieren, denen notwendige Attribute fehlen. Beispielsweise können Sie damit Bilder markieren, die ohne `alt`-Attribut eingefügt wurden.

```
img:not([alt]) { border:3px solid red;
                 background-color:yellow }
```

Ungültige Elemente markieren

Noch einfacher ist es, ungültige HTML-Tags (z.B. aus HTML 3.2) zu markieren, nämlich mit Hilfe einfacher Elementstile. Folgender Stil zeigt, wie Sie in HTML 4 ungültige `center`-Elemente markieren können:

```
center { border:3px solid red;
         background-color:yellow }
```

Elemente zählen

Vielleicht möchten Sie auch einen schnellen Überblick über die Anzahl der Elemente eines bestimmten Typs haben? Interessant wäre das beispielsweise bei Tabellenverschachtelungen. Mit folgenden Stilen implementieren Sie für das `table`-Element einen Zähler `tab` und geben diesen gefolgt von einem Leerzeichen vor dem Element mit der CSS-Klasse `.AnzTab` aus.

```
table          { counter-increment:tab }
.AnzTab:before { content:counter(tab) ' ' }
```

Damit der Zähler angezeigt wird, sollten Sie nach der letzten Tabelle ein Element mit der CSS-Klasse einfügen:

```
<p class="AnzTab">Tabellen enthalten</p>
```

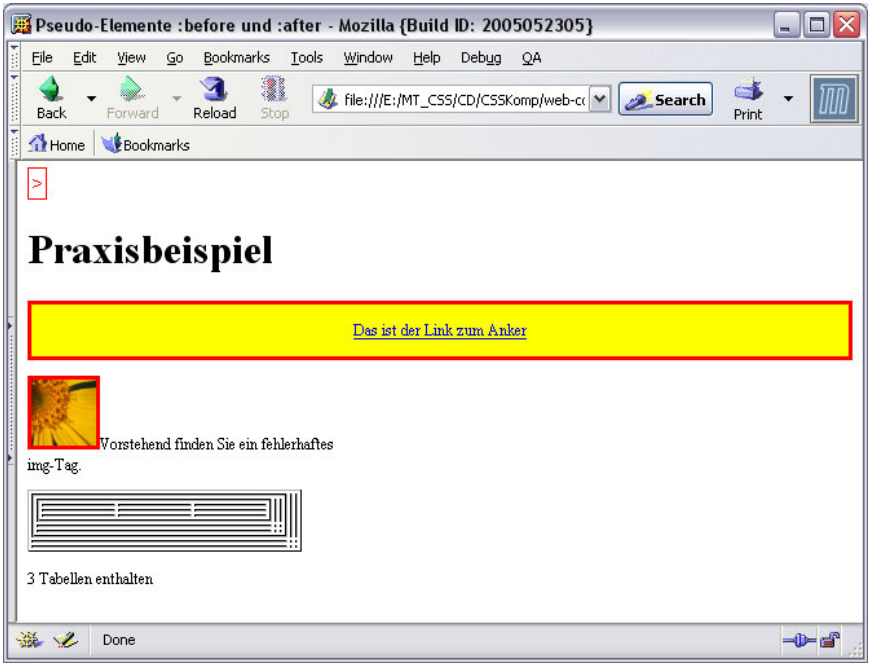


Abbildung 11.16: Die Beispielseite in Mozilla: fehlerhafte Elemente werden markiert und die Anzahl der enthaltenen Tabellen gezählt und ausgegeben.

Teil 3 CSS und Browserkompatibilität im Überblick

A Anhang

A.1 Verwendete Symbole und Bezeichnungen

Zur Darstellung der CSS-Kompatibilität verwenden die Tabellen im Buch einige Symbole, die einer Erklärung bedürfen.

Dieses Symbol bedeutet, dass der CSS-Standard zwar vollständig implementiert ist, aber dennoch so viele Fehler enthält, dass die Darstellung im Prinzip unbrauchbar ist. ○

Dieses Symbol zeigt an, dass der Browser den Standard nur teilweise unterstützt und dass die unterstützten Teile des Standards zudem zahlreiche Fehler enthalten. ⊙

Mit diesem Symbol werden Browser gekennzeichnet, die zwar nur Teile eines Standards beherrschen, diese Teile aber im Wesentlichen fehlerfrei. ⊙

Das Häkchen-Symbol zeigt an, dass der Browser den Standard nahezu vollständig beherrscht. Das schließt selbstverständlich nicht aus, dass der Browser den einen oder anderen Fehler macht. ●

Leere Tabellenzellen geben an, dass der Standard gar nicht berücksichtigt wird und ein »+« hinter der Version zeigt an, dass damit die genannte Version und höhere Versionen gemeint sind. Analog dazu gibt ein Browsername mit »-« hinter der Versionsnummer an, dass neben der angegebenen Version auch Vorversionen gemeint sind. Wenn nicht anders angegeben, beziehen sich die Angaben auf alle Betriebssystemversionen, für die der Browser verfügbar ist.

A.2 Unterstützte CSS-Standards

Browser	CSS 1	CSS 2	CSS 2.1	CSS 3
IE 3	⊙			
IE 4	○	⊙		

Tabelle A.1: Unterstützte CSS-Standards nach Browserversionen

Tabelle A.1: Unterstützte CSS-Standards nach Browserversionen (Forts.)

Browser	CSS 1	CSS 2	CSS 2.1	CSS 3
IE 5.0 (Mac)	●	●		
IE 5.1+ (Mac)	●	●	◎	
IE 5.0/5.01 (Windows)	●	⊙		
IE 5.5 (Windows)	●	●		
IE 6	●	●[1]	◎	
IE 7	●	●[1]	◎	
NN 4.x	○	⊙		
Mozilla 0.x/Netscape 6	●	○[1]		
Mozilla 1.x/Netscape 7	●	○[1]		
Mozilla 1.7/Camino 0.8	●	●[1]		
Mozilla 1.7.8/Netscape 7.1	●	●[1]	◎	◎
Mozilla 1.8	●	●[1]	◎	◎
Firefox 1.0	●	●[1]	◎	
Opera 4	●	●[1]		
Opera 5	●	●[1]		
Opera 6	●	●[1]		
Opera 7	●	●[1]	●	
Opera 8	●	●	●	
Safari 1.x	●	●[1]	◎	◎
Safari 2.0	●	●[1]	◎	◎
Konqueror 3.x	●	●[1]	◎	◎
iCab 2.x	●	◎		
iCab 3.0+	●	●[2]	●[2]	
Palm Web Browser 2.x	◎[2]	◎[2]		

[1] Sprachausgaben werden nicht unterstützt.

[2] Auch Teile von HTML werden nicht unterstützt, die CSS-Unterstützung selbst ist aber gar nicht so schlecht für einen PDA-Browser.

A.3 Mögliche DocType-Angaben

DocType-Angabe	Beschreibung
`<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN" "http://www.w3.org/TR/html4/strict.dtd">`	HTML 4.01 Strict
`<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">`	HTML 4.01 Transitional
`<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Frameset//EN" "http://www.w3.org/TR/html4/frameset.dtd">`	HTML 4.01 Frameset
`<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">`	XHTML 1.0 Strikt
`<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">`	XHTML 1.0 Transitional
`<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Frameset//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-frameset.dtd">`	XHTML 1.0 Frameset
`<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN" "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">`	XHTML 1.1 (entspricht XHTML 1.0 Strict)
`<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN">`	HTML 2.0
`<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 3.2//EN">`	HTML 3.2

Tabelle A.2: Mögliche DocType-Angaben

DocType-Angabe			Browser						
HTML/XML-Version	URI	XML-Prolog	IE 5.x (Mac)	IE 6+ (Win) Opera 7.0 - 7.03	Mozilla 0.6+ N 6.0+	Mozilla 1.1 und 1.01 N 7	Opera 7.1+	Safari 0.82+	Mozilla >1.1 N 7.1+
Kein			Q	Q	Q	Q	Q	Q	Q
Unbekannt			Q	S	S	S	S	S	S
Unbekannt	●		Q	S	S	S	S	S	S
HTML 2.0			Q	Q	Q	Q	Q	Q	Q
HTML 3.2			Q	Q	Q	Q	Q	Q	Q
HTML 3.2	●		Q	Q	Q	Q	Q	Q	Q
HTML 4.0 Strict			S	S	S	S	S	S	S
HTML 4.0 Strict	●		S	S	S	S	S	S	S
HTML 4.0 Transitional			Q	Q	Q	Q	Q	Q	Q

Tabelle A.3: Auswirkungen der DocType-Angabe auf den Anzeigemodus

Tabelle A.3: Auswirkungen der DocType-Angabe auf den Anzeigemodus (Forts.)

DocType-Angabe			Browser						
HTML/XML-Version	**URI**	**XML-Prolog**	**IE 5.x (Mac)**	**IE 6+ (Win) Opera 7.0 - 7.03**	**Mozilla 0.6+ N 6.0+**	**Mozilla 1.1 und 1.01 N 7**	**Opera 7.1+**	**Safari 0.82+**	**Mozilla >1.1 N 7.1+**
HTML 4.0 Transitional	●		S	S	Q	Q	S	Q	Q
HTML 4.0 Frameset			Q	Q	Q	Q	Q	Q	Q
HTML 4.0 Frameset	●		S	S	Q	Q	S	Q	Q
HTML 4.01 Strict			S	S	S	S	S	S	S
HTML 4.01 Strict	●		S	S	S	S	S	S	S
HTML 4.01 Transitional			Q	Q	Q	Q	Q	Q	Q
HTML 4.01 Transitional	●		S	S	S	A	S	Q	S
HTML 4.01 Frameset			Q	Q	Q	Q	Q	Q	Q
HTML 4.01 Frameset	●		S	S	S	A	S	Q	S
XHTML 1.0 Strict	●		S	S	S	S	S	S	S
XHTML 1.0 Transitional	●		S	S	S	A	S	Q	S
XHTML 1.0 Frameset	●		S	S	S	A	S	Q	S
XHTML 1.0 Strict	●	●	S	Q	S	S	S	S	S
XHTML 1.0 Transitional	●	●	S	Q	S	A	S	Q	S
XHTML 1.0 Frameset	●	●	S	Q	S	A	S	Q	S
XHTML 1.1			S	S	S	S	S	S	S
XHTML 1.1		●	S	Q[a]	S	S	S	S	S

a. Genauso verhalten sich die angegebenen Browser auch, wenn vor einer DocType-Angabe in einer HTML-Seite ein Kommentar steht.

Selektoren und Pseudo-Elemente

Selektor	Beispiel	CSS 1.0	CSS 2.0	CSS 2.1	CSS 3.0	CSS Mobile	CSS TV
Element-Selektor	`p`	●	●	●	●	●	●
ID-Selektor	`#banner`	●	●	●	●	●	●
Klassen-Selektor	`.wichtig`	●	●	●	●	●	●
Nachkommen-Selektor	`p span`	●	●	●	●	●	●
Kind-Selektor	`p>span`		●	●	●	●	●
Direkter Geschwister-Selektor	`h1+p`		●	●	●		
Indirekter Geschwister-Selektor	`h1~p`				●		
Universal-Selektor	`div*p`		●	●	●	●	●
Attribut-Selektor: `[Attribut]`	`*[lang]`		●	●	●		
Attribut-Selektor: `[Attribut=Wert]`	`*[lang=de]`		●	●	●		
Attribut-Selektor `[Attribut~=Wert]`	`img[alt~="Logo"]`		●	●	●		
Attribut-Selektor `[Attribut|=Wert]`	`img[alt|="Logo"]`		●	●	●		
Attribut-Selektor `[Attribut^="Anfang"]`	`a[href^= "ftp://"]`				●		
Attribut-Selektor `[Attribut$="Ende"]`	`a[href$="/"]`				●		
Attribut-Selektor `[Attribut*="Teilzeichenfolge"]`	`a[href*= "abc.de"]`				●		
Attribut-Selektor/Pseudo-Klasse `:not[Attribut]`	`img:not[width]`				●		
Kombinierter Attribut-Selektor `*[attributselector] [attributselector]`	`a[href][title] {border-bottom:1px dotted red}`		●	●	●		
Tabellen-Selektoren `table.row[zeile]` `table.column[spalte]`	`table.row[1]`				●		

Tabelle A.4: Verfügbarkeit der Selektoren in den verschiedenen CSS-Standards

Tabelle A.5: Verfügbarkeit der Selektoren in den verschiedenen Browsern

Selektor	Palm	NN	IE			Opera		Mozilla		Safari/ Konq	iCab
	2.x	4.x	5.x (Mac)	6 (Win)	7 (Win)	7	8	0.x+	1.0+	2.0+/ 3.x+	3.0
Element-Selektor	●	●	●	●	●	●	●				●
ID-Selektor	●	●	●	●	●	●	●				●
Klassen-Selektor	●	●	●	●	●	●	●				●
Nachkommen-Selektor	●	○	●	●	●	●	●				●
Indirekter Geschwister-Selektor											
Kind-Selektor	●		●			●	●	●	●	●	●
Direkter Geschwister-Selektor	●		●			●	●	●	●	●	●
Universal-Selektor	●	◎	●	●	●	●	●	●	●	●	◎
Attribut-Selektor `[Attribut]`						●	●	●	●	●	●
Attribut-Selektor `[Attribut=Wert]`						●	●	●	●	●	●
Attribut-Selektor `[Attribut~=Wert]`	●					●	●	●	●	●	●
Attribut-Selektor `[Attribut|=Wert]`						●	●	●	●	●	●
Attribut-Selektor `[Attribut^="Anfang"]`									●[1]		●
Attribut-Selektor `[Attribut$="Ende"]`									●[1]		●

Selektor	Palm	NN	IE			Opera		Mozilla		Safari/ Konq	iCab
	2.x	4.x	5.x (Mac)	6 (Win)	7 (Win)	7	8	0.x+	1.0+	2.0+/ 3.x+	3.0
Attribut-Selektor `[Attribut*= "Ende"]`									●[1]		●
Attribut-Selektor/ Pseudo-Klasse `:not[Attribut]`									●[1]	●	
Kombinierter Attribut-Selektor `*[attribut-selector] [attribut-selector]`						●[2]	●[2]	●[2]	●[2]	○[2]	●[2]
Tabellen-Selektoren `table.row [zeile]` `table.column [spalte]`											

[1] Unterstützung erst ab Mozilla 1.4+
[2] Vorausgesetzt, es werden nur Attribut-Selektoren verwendet, die der Browser unterstützt

Tabelle A.5: Verfügbarkeit der Selektoren in den verschiedenen Browsern (Forts.)

A.4 Farben

HTML-Farbname	Farbe
`aqua`	aquamarinblau
`black`	schwarz
`blue`	blau
`fuchsia`	fuchsia
`gray`	grau
`green`	grün
`lime`	limonengelb
`maroon`	kastanienbraun
`navy`	navygrün
`olive`	olivgrün

Tabelle A.6: HTML-Farbnamen zur Verwendung als Farbangabe

Tabelle A.6: HTML-Farbnamen zur Verwendung als Farbangabe (Forts.)

HTML-Farbname	Farbe
purple	violett
red	rot
silver	silber
teal	blau-grün (teal)
white	weiß
yellow	gelb

A.5 Einheiten

Tabelle A.7: Verfügbare CSS-Maßeinheiten

Maßeinheit	Zuordnung	Bedeutung
%	relativ	Prozent in Relation zum umgebenden Element oder der Standardgröße des Elements
cm	absolut	Zentimeter
em	relativ	Relativ zur elementeigenen Schrifthöhe, diese kann wiederum absolut festgelegt sein oder im Browser eingestellt werden.
ex	relativ	Relativ zur Höhe des Buchstabens »x«
in	absolut	Zoll (1 Zoll=2,54 cm)
mm	absolut	Millimeter
pc	absolut	Pica (12 Punkt [pt] entsprechen 1 Pica)
pt	absolut	Punkt (1 Punkt = 1/72 Zoll)
px	relativ	Pixel in Relation zur eingestellten Auflösung

A.6 Browserkompatibilität der @-Regeln

Tabelle A.8: Browserkompatibilität der @-Regeln

Selektor	Palm	NN	IE			Opera		Mozilla		Safari/ Konq	iCab
	2.x	4.x	5.x (Mac)	6 (Win)	7 (Win)	7	8	0.x+	1.0+	2.0+/ 3.x+	3.0
@media	●			●	●	●	●		●	●	●
@import (mit url())	●		●	●	●	●	●	●	●	●	●
@import (ohne url())	●		●	●	●	●	●	●	●	●	●

Selektor	Palm	NN	IE			Opera		Mozilla		Safari/ Konq	iCab
	2.x	4.x	5.x (Mac)	6 (Win)	7 (Win)	7	8	0.x+	1.0+	2.0+/ 3.x+	3.0
@import (mit url() und Medientyp)	●					●	●	●	●	●	●
@page			●	●	●	●	●	●	●	●	●
@font-face											

Tabelle A.8: Browserkompatibilität der @-Regeln (Forts.)

A.7 Textformatierungen

CSS-Eigenschaft	CSS	Palm	NN	IE			Opera		Mozilla		Safari/Konq	iCab
		2.x	4.x	5.x (Mac)	6 (Win)	7 Win)	7	8	0.x+	1.0+	2.0+/ 3.x+	2.9
color	1-3, TV, Mobile	◎	●	●	●	●	●	●	●	●	●	●
font-family	1-3, TV, Mobile		●	●	●	●	●	●	●	●	●	●
font-style	1-3, TV, Mobile	●	◎	●	●	●	●	●	●	●	●	●
font-variant	1-3, TV, Mobile			●	●	●	●	●	●	●	●	●
font-weight	1-3, TV, Mobile	◎	◎	●	●	●	●	●	●	●	●	●
font-stretch	1, 2, 3											●
font-size	1-3, TV, Mobile	◎	●	●	●	●	●	●	●	●	●	●
text-indent	1-3, TV, Mobile	◎	◎	●	●	●	●	●	●	●	●	●
text-align	1-3, TV, Mobile	◎	◎	◎	●	●	●	●	●	●	●	●
text-decoration	1-3, TV, Mobile	●	◎	◎	●	◎	●	●	●	●	●	●

Tabelle A.9: Textformatierungen

Tabelle A.9: Textformatierungen (Forts.)

CSS-Eigenschaft	CSS	Palm	NN	IE			Opera		Mozilla		Safari/Konq	iCab
		2.x	4.x	5.x (Mac)	6 (Win)	7 Win)	7	8	0.x+	1.0+	2.0+/ 3.x+	2.9
text-shadow	2, 3										◉	
letter-spacing	1-3			●	●	●	●	●	●	●	●	●
word-spacing	1-3			●	●	●	●	●	●	●	●	●
text-transform	1-3, TV, Mobile		⊙	●	●	●	●	●	●	●	●	●
white-space	1-3, TV, Mobile	⊙	◉	◉	◉	◉	◉	●	◉	◉	●	●
line-height	1-3, TV	●	●	●	●	●	●	●	●	●	●	●
vertical-align	2, 3 TV, Mobile	●	◉	●	●	●	●	●	●	●	●	●
text-align-last	3				◉	◉						
text-justify	3				●	●						
font-effect	3											
quotes	2-3			⊙			◉	◉	◉	◉	⊙	◉

A.8 Anzeigetyp (display-Eigenschaft)

Tabelle A.10: Anzeigetyp (display-Eigenschaft)

display-Eigenschaft	CSS	Palm	NN	IE			Opera		Mozilla		Safari/Konq	iCab
Wert		2.x	4.x	5.x (Mac)	6 (Win)	7 Win)	7	8	0.x+	1.0+	2.0+/ 3.x+	3.0
none	1-3, Mobile, TV	●	○	●	●	●	●	●	●	●	●	●
block	1-3, Mobile, TV	●	○	●	●	●	●	●	●	●	●	●
inline	1-3, Mobile, TV	●		●	●	●	●	●	●	●	●	●
list-item	1-3, Mobile, TV	●		●	●	●	●	●	●	●	●	●

display-Eigenschaft	CSS	Palm	NN	IE			Opera		Mozilla		Safari/Konq	iCab
Wert		**2.x**	**4.x**	**5.x (Mac)**	**6 (Win)**	**7 Win)**	**7**	**8**	**0.x+**	**1.0+**	**2.0+/ 3.x+**	**3.0**
inherit	1-3, TV	●		●			●	●	●	●	●	●
marker	2											
run-in	2 + 3	○		●			●	●			●	
compact	2.0	○										
table	2-3	○		●			●	●	●	●	●	
table-row	2-3	○					●	●	●	●	●	
table-cell	2-3	○					●	●	●	●	●	
table-column	2-3	●										
table-caption	2-3	○					●	●	●	●	●	
table-header-group	2-3	○					●	●	●	●	●	
table-footer-group	2-3	○					●	●	●	●	●	
table-column-group	2-3			○					●			
table-row-group	2-3			○					●			
inline-table	2-3						●	●			●	

Tabelle A.10: Anzeigetyp (display-Eigenschaft) (Forts.)

A.9 Größen und Positionierung

Eigenschaft	CSS-Version	Palm	NN	IE			Opera		Mozilla		Safari/Konq	iCab
		2.x	**4.x**	**5.x (Mac)**	**6 (Win)**	**7 Win)**	**7**	**8**	**0.x+**	**1.0+**	**2.0+/ 3.x+**	**3.0**
bottom	2-3, TV			●	●	●	●	●	●	●	●	●
top	2-3, TV			●	●	●	●	●	●	●	●	●
left	2-3, TV	●	⊙	○	●	●	●	●	●	●	●	●

Tabelle A.11: Größen und Positionierung

Tabelle A.11: Größen und Positionierung (Forts.)

Eigenschaft	CSS-Version	Palm	NN	IE			Opera		Mozilla		Safari/Konq	iCab
		2.x	4.x	5.x (Mac)	6 (Win)	7 Win)	7	8	0.x+	1.0+	2.0+/ 3.x+	3.0
right	2-3, TV	●		○	●	●	●	●	●	●	●	●
height	1-3, TV, Mobile			●	◉	◉	◉	◉	◉	◉	◉	●
max-height	2-3						◉	◉	◉	◉	◉	◉
min-height	2-3						◉	◉	◉	◉	◉	●
width	1-3, TV, Mobile	●	◉	●	●	●	●	●	●	●	◉	●
min-width	2-3						●	●	●	●	●	●
max-width	2-3						●	●	●	●	●	●
position	2-3, TV	⊙		●	◉	⊙	●	●	●	●	●	●
float	1-3, TV, Mobile	⊙		●	●	●	●	●	●	●	●	●
z-index	2-3, TV	◉	●	●	●	●	●	●	●	●	●	●
visibility	2-3, TV, Mobile	◉	◉	◉	◉		◉	◉	●	●	◉	◉
clear	1-3, TV, Mobile	●	●	●	●		●	●	●	●	●	●
overflow	2-3	◉		⊙	●	●	●	●	●	●	●	●
CLIP	2, 2.1, 3			●	●	●	●	●	●	●	●	●
padding	1-3, TV, Mobile	⊙	●	○	●	●	●	●	●	●	●	●
margin	1-3, TV, Mobile	⊙	●	●	●	●	●	●	●	●	●	●

A.10 Listen, Marker und Aufzählungen

Eigenschaft	CSS-Version	Palm	NN	IE			Opera		Mozilla		Safari/Konq	iCab
		2.x	4.x	5.x (Mac)	6 (Win)	7 Win)	7	8	0.x+	1.0+	2.0+/ 3.x+	3.0
list-style-type	1-3, TV, Mobile	◉	◉	◉	◉	◉	◉	◉	◉	◉	◉	◉
list-style-image	1-3, TV, Mobile	●		●	●	●	●	●	●	●	●	●
list-style-position	1-3, TV, Mobile	◉	◉	◉	◉	◉	◉	◉	◉	◉	◉	●
list-style	1-3, TV, Mobile	●	●	●	●	●	●	●	●	●	●	●
content	2-3						⊙	⊙	◉	◉	◉	●
counter-increment	2-3						●	●		◉[1]		●
counter-reset	2-3						●	●		◉[1]		●
marker-offset	2, 3											

[1] ab Version 1.8

Tabelle A.12: Listen, Marker und Aufzählungen

A.11 Bilder und Hintergründe

Eigenschaft	CSS-Version	Palm	NN	IE			Opera		Mozilla		Safari/Konq	iCab
		2.x	4.x	5.x (Mac)	6 (Win)	7 Win)	7	8	0.x+	1.0+	2.0+/ 3.x+	3.0
background-color	1-3, TV, Mobile	●	○	●	●	●	●	●	●	●	●	●
background-image	1-3, TV, Mobile	○	○	●	●	●	●	●	●	●	●	●
background-repeat	1-3, TV, Mobile	●	○	●	●	●	●	●	●	●	●	●

Tabelle A.13: Bilder und Hintergründe

Tabelle A.13: Bilder und Hintergründe (Forts.)

Eigenschaft	CSS-Version	Palm	NN	IE			Opera		Mozilla		Safari/Konq	iCab
		2.x	4.x	5.x (Mac)	6 (Win)	7 Win)	7	8	0.x+	1.0+	2.0+/ 3.x+	3.0
background-position	1-3, TV, Mobile	●		●	●	●	●	●	●	●	●	●
background-attachment	1-3	●		●	●la	●	●	●	●	●	●	●
background	1-3, TV, Mobile	●	●	●	●	●	●	●	●	●	●	●
opacity	3									●[1]	●	

[1] ab Version 1.4+

A.12 Linien und Rahmen

Tabelle A.14: Linien und Rahmen

Eigenschaft	CSS-Version	Palm	NN	IE			Opera		Mozilla		Safari/Konq	iCab
		2.x	4.x	5.x (Mac)	6 (Win)	7 Win)	7	8	0.x+	1.0+	2.0+/ 3.x+	3.0
border-style	1-3, TV, Mobile	◘	○	⊙	⊙	⊙	●	●	●	●	●	●
border-top-style	1-3, TV, Mobile	◘	○	⊙	⊙	⊙	●	●	●	●	●	●
border-right-style	1-3, TV, Mobile	◘	○	⊙	⊙	⊙	●	●	●	●	●	●
border-bottom-style	1-3, TV, Mobile	◘	○	⊙	⊙	⊙	●	●	●	●	●	●
border-left-style	1-3, TV, Mobile	◘	○	⊙	⊙	⊙	●	●	●	●	●	●
border-width	1-3, TV, Mobile	⊙	○	●	●	●	●	●	●	●	●	●

Eigenschaft	CSS-Version	Palm	NN	IE			Opera		Mozilla		Safari/Konq	iCab
		2.x	4.x	5.x (Mac)	6 (Win)	7 Win)	7	8	0.x+	1.0+	2.0+/ 3.x+	3.0
border-top-width	1-3, TV, Mobile	⊙	○	●	●	●	●	●	●	●	●	●
border-right-width	1-3, TV, Mobile	⊙	○	●	●	●	●	●	●	●	●	●
border-bottom-width	1-3, TV, Mobile	⊙	○	●	●	●	●	●	●	●	●	●
border-left-width	1-3, TV, Mobile	⊙	○	●	●	●	●	●	●	●	●	●
border-color	1-3, TV, Mobile	●	●	●	●	●	●	●	●	●	●	●
border-top-color	1-3, TV, Mobile	●	●	●	●	●	●	●	●	●	●	●
border-right-color	CSS 1-3, TV, Mobile	●	●	●	●	●	●	●	●	●	●	●
border-bottom-color	1-3, TV, Mobile	●	●	●	●	●	●	●	●	●	●	●
border-left-color	1-3, TV, Mobile	●	●	●	●	●	●	●	●	●	●	●
border	1-3, TV, Mobile	●	●	●	●	●	●	●	●	●	●	●
border-top	1-3, TV, Mobile	●		●	●	●	●	●	●	●	●	●
border-right	1-3, TV, Mobile	●		●	●	●	●	●	●	●	●	●
border-bottom	1-3, TV, Mobile	●		●	●	●	●	●	●	●	●	●

Tabelle A.14: Linien und Rahmen (Forts.)

Tabelle A.14: Linien und Rahmen (Forts.)

Eigenschaft	CSS-Version	Palm	NN	IE			Opera		Mozilla		Safari/Konq	iCab
		2.x	4.x	5.x (Mac)	6 (Win)	7 Win)	7	8	0.x+	1.0+	2.0+/ 3.x+	3.0
border-left	1-3, TV, Mobile	●		●	●	●	●	●	●	●	●	●
border-radius	3											
border-top-left-radius	3											
border-top-right-radius	3											
border-bottom-left-radius	3											
border-top-right-radius	3											
border-break	3											
box-shadow	3											
outline	2-3, TV			⊙			●	●		●[1]	⊙	
outline-width	2-3, TV			⊙			●	●		●[1]	⊙	
outline-style	2-3, TV			⊙			●	●		●[1]	⊙	
outline-color	2-3, TV			⊙			●	●		●[1]	⊙	

[1] ab Mozilla 1.8

A.13 Tabellen und Zellen

Eigenschaft	CSS-Version	Palm	NN	IE			Opera		Mozilla		Safari/Konq	iCab
		2.x	4.x	5.x (Mac)	6 (Win)	7 Win)	7	8	0.x+	1.0+	2.0+/ 3.x+	2.9+
table-layout	2-3			●	●	●	●	●	●	●	●	●
caption-side	2-3, TV	◎		◎			◎	◎	◎	●[1]	◎	◎
border-collapse	2-3	●			●	●	●	●		●[1]	●	●
border-spacing	2-3						◎	◎		◎[1]	◎	●
empty-cells	2-3			⊙			●	●	●	●	●	●
text-align	2, 3, TV, Mobile		◎	◎	◎	◎	◎	◎	◎	◎	◎	◎

[1] ab Version 1.4+

Tabelle A.15: Tabellen und Zellen

A.14 Seitenlayout – Druckausgabe

Eigenschaft	CSS-Version	Palm	NN	IE			Opera		Mozilla		Safari/Konq	iCab
		2.x+	4.x	5.x (Mac)	6 (Win)	7 Win)	7	8	0.x+	1.0+	2.0+/ 3.x+	3.0
size	2, 3			⊙			⊙	⊙				
page	2, 3											
marks	2, 3											
page-break-before	2, 3			●	●	●	●	●		●[1]	●	
page-break-after	2, 3			●	●	●	●	●		●[1]	●	
page-break-inside	2, 3				●	●	●	●				
orphans	2-3											
widows	2-3											
:first	2-3						●	●				

Tabelle A.16: Seitenlayout – Druckausgabe

Tabelle A.16: Seitenlayout – Druckausgabe (Forts.)

Eigenschaft	CSS-Version	Palm	NN	IE			Opera		Mozilla		Safari/Konq	iCab
		2.x+	4.x	5.x (Mac)	6 (Win)	7 Win)	7	8	0.x+	1.0+	2.0+/ 3.x+	3.0
:right	2-3						●	●				
:left	2-3						●	●				
margin	2-3						⊙	⊙				

+ keine Druckausgabe möglich
1 ab Version 1.4+

A.15 Sprachausgabe

Tabelle A.17: Sprachausgabe

Eigenschaft	CSS-Version	Palm	NN	IE			Opera		Mozilla		Safari/Konq	iCab
		2.x	4.x	5.x (Mac)	6 (Win)	7 Win)	7	8	0.x+	1.0+	2.0+/ 3.x+	3.0
cue-before	2, 3											
cue-after	2, 3											
cue	2, 3											
speak-punctuation	2, 3											
speak-header	2, 3											
speak-numeral	2, 3											
pitch-range	2-3											
stress	2-3											
play-during	2-3											
azimuth	2-3											
volume	2-3											
pause-before	2-3											
pause-after	2-3											
pause	2-3											
speak	2-3											
speech-rate	2-3											

Eigenschaft	CSS-Version	Palm	NN	IE			Opera		Mozilla		Safari/Konq	iCab
		2.x	4.x	5.x (Mac)	6 (Win)	7 Win)	7	8	0.x+	1.0+	2.0+/ 3.x+	3.0
richness	2-3											
elevation	2-3											
voice-family	2-3											
pitch	2-3											

Tabelle A.17: Sprachausgabe (Forts.)

A.16 Pseudo-Elemente und Pseudoklassen

Eigenschaft	CSS-Version	Palm	NN	IE			Opera		Mozilla		Safari/Konq	iCab
		2.x	4.x	5.x (Mac)	6 (Win)	7 Win)	7	8	0.x+	1.0+	2.0+/ 3.x+	3.0
:first-child	2-3, TV			●			●	●		●	●	●
:link	1-3, TV, Mobile	●	●	●	●	●	●	●	●	●	●	●
:visited	1-3, TV, Mobile	●	●	●	●	●	●	●	●	●	●	●
:hover	1-3			⊙	⊙	⊙	●	●	●	●	●	●
:active	1-3, TV, Mobile			⊙	⊙	⊙	●	●	●	●	●	●
:focus	2-3, TV, Mobile			⊙			⊙	⊙	●	●	⊙	⊙
:lang()	2-3			●			●	●		●[1]		●
:first-line	1-3, TV			●			●	●		●[1]	●	●
:first-letter	1-3, TV			●			●	●		●[1]	●	●
:before	2-3						⊙	⊙		●[1]	⊙	●
:after	2-3						⊙	⊙		●[1]	⊙	●
:not	3									●[1]	●	

[1] ab Version 1.4+

Tabelle A.18: Pseudo-Elemente und Pseudo-Klassen

A.17 Sonstige Formatierungen

Tabelle A.19: Sonstige Formatierungen

Eigenschaft	CSS-Version	Palm	NN	IE			Opera		Mozilla		Safari/Konq	iCab
		2.x	4.x	5.x (Mac)	6 (Win)	7 (Win)	7	8	0.x+	1.0+	2.0+/ 3.x+	3.0
direction	2-3				●	●	●	●		●	●	
unicode-bidi	2-3											
cursor	2-3			⊙	⊙		⊙	⊙	⊙	⊙	⊙	⊙

A.18 Der CSS 1.0-Standard

Tabelle A.20: Der CSS 1.0-Standard

CSS-Eigenschaft	CSS-Version	Palm	NN	Internet Explorer			Opera		Mozilla		Safari/ Konq	iCab
		2.x	4.x	5.x (Mac)	6 (Win)	7 (Win)	7	8	0.x+	1.0+	2.0+/ 3.x+	3.0
:active	1-3, TV, Mobile			⊙	⊙	⊙	●	●	●	●	●	●
:first-letter	1-3, TV			●			●	●		●[1]	●	●
:first-line	1-3, TV			●			●	●		●[1]	●	●
:hover	1-3			⊙	⊙	⊙	●	●	●	●	●	●
:link	1-3, TV, Mobile	●	●	●	●	●	●	●	●	●	●	●
:visited	1-3, TV, Mobile	●	●	●	●	●	●	●	●	●	●	●
background	1-3, TV, Mobile	●	●	●	●	●	●	●	●	●	●	●
background-attachment	1-3	●		●	●	●	●	●	●	●	●	●
background-color	1-3, TV, Mobile	●	○	●	●	●	●	●	●	●	●	●
background-image	1-3, TV, Mobile	○	○	●	●	●	●	●	●	●	●	●

Tabelle A.20: Der CSS 1.0-Standard (Forts.)

CSS-Eigenschaft	CSS-Version	Palm	NN	Internet Explorer			Opera		Mozilla		Safari/ Konq	iCab
		2.x	4.x	5.x (Mac)	6 (Win)	7 (Win)	7	8	0.x+	1.0+	2.0+/ 3.x+	3.0
background-position	1-3, TV, Mobile	●		●	●	●	●	●	●	●	●	●
background-repeat	1-3, TV, Mobile	●	○	●	●	●	●	●	●	●	●	●
border	1-3, TV, Mobile	●	●	●	●	●	●	●	●	●	●	●
border-bottom	1-3, TV, Mobile	●		●	●	●	●	●	●	●	●	●
border-bottom-color	1-3, TV, Mobile	●	●	●	●	●	●	●	●	●	●	●
border-bottom-style	1-3, TV, Mobile	◘	○	⊙	⊙	⊙	●	●	●	●	●	●
border-bottom-width	1-3, TV, Mobile	⊙	○	●	●	●	●	●	●	●	●	●
border-color	1-3, TV, Mobile	●	●	●	●	●	●	●	●	●	●	●
border-left	1-3, TV, Mobile	●		●	●	●	●	●	●	●	●	●
border-left-color	1-3, TV, Mobile	●	●	●	●	●	●	●	●	●	●	●
border-left-style	1-3, TV, Mobile	◘	○	⊙	⊙	⊙	●	●	●	●	●	●
border-left-width	1-3, TV, Mobile	⊙	○	●	●	●	●	●	●	●	●	●
border-right	1-3, TV, Mobile	●		●	●	●	●	●	●	●	●	●

Tabelle A.20: Der CSS 1.0-Standard (Forts.)

CSS-Eigenschaft	CSS-Version	Palm	NN	Internet Explorer			Opera		Mozilla		Safari/ Konq	iCab
		2.x	4.x	5.x (Mac)	6 (Win)	7 (Win)	7	8	0.x+	1.0+	2.0+/ 3.x+	3.0
`border-right-color`	1-3, TV, Mobile	●	●	●	●	●	●	●	●	●	●	●
`border-right-style`	1-3, TV, Mobile	◉	○	⊙	⊙	⊙	●	●	●	●	●	●
`border-right-width`	1-3, TV, Mobile	⊙	○	●	●	●	●	●	●	●	●	●
`border-style`	1-3, TV, Mobile	◉	○	⊙	⊙	⊙	●	●	●	●	●	●
`border-style`	1-3, TV, Mobile	◉	○	⊙	⊙	⊙	●	●	●	●	●	●
`border-top`	1-3, TV, Mobile	●		●	●	●	●	●	●	●	●	●
`border-top-color`	1-3, TV, Mobile	●	●	●	●	●	●	●	●	●	●	●
`border-top-style`	1-3, TV, Mobile	◉	○	⊙	⊙	⊙	●	●	●	●	●	●
`border-top-width`	1-3, TV, Mobile	⊙	○	●	●	●	●	●	●	●	●	●
`border-width`	1-3, TV, Mobile	⊙	○	●	●	●	●	●	●	●	●	●
`clear`	1-3, TV, Mobile	●	●	●	●		●	●	●	●	●	●
`color`	1-3, TV, Mobile	◉	●	●	●	●	●	●	●	●	●	●
`display`	1-3, Mobile, TV	◉	◉	◉	◉	◉	◉	◉	◉	◉	◉	◉

Tabelle A.20: Der CSS 1.0-Standard (Forts.)

CSS-Eigenschaft	CSS-Version	Palm	NN	Internet Explorer			Opera		Mozilla		Safari/ Konq	iCab
		2.x	4.x	5.x (Mac)	6 (Win)	7 (Win)	7	8	0.x+	1.0+	2.0+/ 3.x+	3.0
float	1-3, TV, Mobile	⊙		●	●	●	●	●	●	●	●	●
font-family	1-3, TV, Mobile		●	●	●	●	●	●	●	●	●	●
font-size	1-3, TV, Mobile	○	●	●	●	●	●	●	●	●	●	●
font-stretch	1, 2, 3											●
font-style	1-3, TV, Mobile		○	●	●	●	●	●	●	●	●	●
font-variant	1-3, TV, Mobile			●	●	●	●	●	●	●	●	●
font-weight	1-3, TV, Mobile	○	○	●	●	●	●	●	●	●	●	●
height	1-3, TV, Mobile			●	○	○	○	○	○	○	○	●
letter-spacing	1-3			●	●	●	●	●	●	●	●	●
line-height	1-3, TV	●	●	●	●	●	●	●	●	●	●	●
list-style	1-3, TV, Mobile	●	●	●	●	●	●	●	●	●	●	●
list-style-image	1-3, TV, Mobile	●		●	●	●	●	●	●	●	●	●
list-style-position	1-3, TV, Mobile	○	○	○	○	○	○	○	○	○	○	●
list-style-type	1-3, TV, Mobile	○	○	○	○	○	○	○	○	○	○	○
margin	1-3, TV, Mobile	⊙	●	●	●	●	●	●	●	●	●	●

Tabelle A.20: Der CSS 1.0-Standard (Forts.)

CSS-Eigenschaft	CSS-Version	Palm	NN	Internet Explorer			Opera		Mozilla		Safari/ Konq	iCab
		2.x	4.x	5.x (Mac)	6 (Win)	7 (Win)	7	8	0.x+	1.0+	2.0+/ 3.x+	3.0
padding	1-3, TV, Mobile	⊙	●	○	●	●	●	●	●	●	●	●
text-align	1-3, TV, Mobile	◎	◎	◎	●	●	●	●	●	●	●	●
text-decoration	1-3, TV, Mobile	●	◎	◎	●	◎	●	●	●	●	●	●
text-indent	1-3, TV, Mobile	◎	⊙	●	●	●	●	●	●	●	●	●
text-transform	1-3, TV, Mobile		⊙	●	●	●	●	●	●	●	●	●
vertical-align	2, 3, TV, Mobile	●	◎	●	●	●	●	●	●	●	●	●
white-space	1-3, TV, Mobile	⊙	⊙	●	●	●	●	●	●	●	●	●
width	1-3, TV, Mobile	●	◎	●	●	●	●	●	●	●	◎	●
word-spacing	1-3			●	●	●	●	●	●	●	●	●

[1] ab Version 1.4+
[2] ab Version 1.8+

A.19 Der CSS 2.0-Standard

Tabelle A.21: Der CSS 2.0-Standard

CSS-Eigenschaft	CSS-Version	Palm	NN	Internet Explorer			Opera		Mozilla		Safari/ Konq	iCab
		2.x	4.x	5.x (Mac)	6 (Win)	7 (Win)	7	8	0.x+	1.0+	2.0+/ 3.x+	3.0
:active	1-3, TV, Mobile			◎	◎	◎	●	●	●	●	●	●
:after	2-3						⊙	⊙		●[1]	⊙	●

Tabelle A.21: Der CSS 2.0-Standard (Forts.)

CSS-Eigenschaft	CSS-Version	Palm	NN	Internet Explorer			Opera		Mozilla		Safari/ Konq	iCab
		2.x	4.x	5.x (Mac)	6 (Win)	7 (Win)	7	8	0.x+	1.0+	2.0+/ 3.x+	3.0
:before	2-3						⊙	⊙		●[1]	⊙	●
:first	2-3						●	●				
:first-child	2-3, TV			●			●	●		●	●	●
:first-letter	1-3, TV			●			●	●		●[1]	●	●
:first-line	1-3, TV			●			●	●		●[1]	●	●
:focus	2-3, TV, Mobile			⊙			⊙	⊙	●	●	⊙	⊙
:hover	1-3			⊙	⊙	⊙	●	●	●	●	●	●
:lang()	2-3			●			●	●		●[1]		●
:left	2-3						●	●				
:link	1-3, TV, Mobile	●	●	●	●	●	●	●	●	●	●	●
:right	2-3						●	●				
:visited	1-3, TV, Mobile	●	●	●	●	●	●	●	●	●	●	●
azimuth	2-3											
background	1-3, TV, Mobile	●	●	●	●	●	●	●	●	●	●	●
background-attachment	1-3	●		●	●	●	●	●	●	●	●	●
background-color	1-3, TV, Mobile	●	○	●	●	●	●	●	●	●	●	●
background-image	1-3, TV, Mobile	○	○	●	●	●	●	●	●	●	●	●
background-position	1-3, TV, Mobile	●		●	●	●	●	●	●	●	●	●

Tabelle A.21: Der CSS 2.0-Standard (Forts.)

CSS-Eigenschaft	CSS-Version	Palm	NN	Internet Explorer			Opera		Mozilla		Safari/ Konq	iCab
		2.x	4.x	5.x (Mac)	6 (Win)	7 (Win)	7	8	0.x+	1.0+	2.0+/ 3.x+	3.0
background-repeat	1-3, TV, Mobile	●	○	●	●	●	●	●	●	●	●	●
border	1-3, TV, Mobile	●	●	●	●	●	●	●	●	●	●	●
border-bottom	1-3, TV, Mobile	●		●	●	●	●	●	●	●	●	●
border-bottom-color	1-3, TV, Mobile	●	●	●	●	●	●	●	●	●	●	●
border-bottom-style	1-3, TV, Mobile	◉	○	⊙	⊙	⊙	●	●	●	●	●	●
border-bottom-width	1-3, TV, Mobile	⊙	○	●	●	●	●	●	●	●	●	●
border-collapse	2-3	●			●	●	●	●		●[1]	●	●
border-color	1-3, TV, Mobile	●	●	●	●	●	●	●	●	●	●	●
border-left	1-3, TV, Mobile	●		●	●	●	●	●	●	●	●	●
border-left-color	1-3, TV, Mobile	●	●	●	●	●	●	●	●	●	●	●
border-left-style	1-3, TV, Mobile	◉	○	⊙	⊙	⊙	●	●	●	●	●	●
border-left-width	1-3, TV, Mobile	⊙	○	●	●	●	●	●	●	●	●	●
border-right	1-3, TV, Mobile	●		●	●	●	●	●	●	●	●	●
border-right-color	1-3, TV, Mobile	●	●	●	●	●	●	●	●	●	●	●

Tabelle A.21: Der CSS 2.0-Standard (Forts.)

CSS-Eigenschaft	CSS-Version	Palm	NN	Internet Explorer			Opera		Mozilla		Safari/ Konq	iCab
		2.x	4.x	5.x (Mac)	6 (Win)	7 (Win)	7	8	0.x+	1.0+	2.0+/ 3.x+	3.0
border-right-style	1-3, TV, Mobile	◉	○	⊙	⊙	⊙	●	●	●	●	●	●
border-right-width	1-3, TV, Mobile	⊙	○	●	●	●	●	●	●	●	●	●
border-spacing	2-3						◉	◉		◉[1]	◉	●
border-style	1-3, TV, Mobile	◉	○	⊙	⊙	⊙	●	●	●	●	●	●
border-style	1-3, TV, Mobile	◉	○	⊙	⊙	⊙	●	●	●	●	●	●
border-top	1-3, TV, Mobile	●		●	●	●	●	●	●	●	●	●
border-top-color	1-3, TV, Mobile	●	●	●	●	●	●	●	●	●	●	●
border-top-style	1-3, TV, Mobile	◉	○	⊙	⊙	⊙	●	●	●	●	●	●
border-top-width	1-3, TV, Mobile	⊙	○	●	●	●	●	●	●	●	●	●
border-width	1-3, TV, Mobile	⊙	○	●	●	●	●	●	●	●	●	●
bottom	2-3, TV			●	●	●	●	●	●	●	●	●
caption-side	2-3,	◉		◉			◉	◉	◉	●[1]	◉	◉
clear	1-3, TV, Mobile	●	●	●	●		●	●	●	●	●	●
clip	2, 2.1, 3			●	●	●	●	●	●	●	●	●
color	1-3, TV, Mobile	◉	●	●	●	●	●	●	●	●	●	●

Tabelle A.21: Der CSS 2.0-Standard (Forts.)

CSS-Eigenschaft	CSS-Version	Palm	NN	Internet Explorer			Opera		Mozilla		Safari/ Konq	iCab
		2.x	4.x	5.x (Mac)	6 (Win)	7 (Win)	7	8	0.x+	1.0+	2.0+/ 3.x+	3.0
content	2-3						⊙	⊙	○	○	○	●
counter-increment	2-3						●	●		○[2]		●
counter-reset	2-3						●	●		○[2]		●
cue	2, 3											
cue-after	2, 3											
cue-before	2, 3											
cursor	2-3			○	○		○	○	○	○	○	○
display	1-3, TV, Mobile	○	○	○	○	○	○	○	○	○	○	○
elevation	2-3											
empty-cells	2-3			⊙			●	●	●	●	●	●
float	1-3, TV, Mobile	⊙		●	●	●	●	●	●	●	●	●
font-family	1-3, TV, Mobile		●	●	●	●	●	●	●	●	●	●
font-size	1-3, TV, Mobile	○	●	●	●	●	●	●	●	●	●	●
font-stretch	1, 2, 3											●
font-style	1-3, TV, Mobile		○	●	●	●	●	●	●	●	●	●
font-variant	1-3, TV, Mobile			●	●	●	●	●	●	●	●	●
font-weight	1-3, TV, Mobile	○	○	●	●	●	●	●	●	●	●	●
height	1-3, TV, Mobile			●	○	○	○	○	○	○	○	●
left	2-3, TV	●	○	○	●	●	●	●	●	●	●	●

Tabelle A.21: Der CSS 2.0-Standard (Forts.)

CSS-Eigenschaft	CSS-Version	Palm	NN	Internet Explorer			Opera		Mozilla		Safari/ Konq	iCab
		2.x	4.x	5.x (Mac)	6 (Win)	7 (Win)	7	8	0.x+	1.0+	2.0+/ 3.x+	3.0
letter-spacing	1-3			●	●	●	●	●	●	●	●	●
line-height	1-3, TV	●	●	●	●	●	●	●	●	●	●	●
list-style	1-3, TV, Mobile	●	●	●	●	●	●	●	●	●	●	●
list-style-image	1-3, TV, Mobile	●		●	●	●	●	●	●	●	●	●
list-style-position	1-3, TV, Mobile	◎	◎	◎	◎	◎	◎	◎	◎	◎	◎	●
list-style-type	1-3, TV, Mobile	◎	◎	◎	◎	◎	◎	◎	◎	◎	◎	◎
margin	1-3, TV, Mobile	⊙	●	●	●	●	●	●	●	●	●	●
margin	2-3						⊙	⊙				
marker-offset	2, 3											
marks	2, 3											
max-height	2-3						◎	◎	◎	◎	◎	●
max-width	2-3						●	●	●	●	●	●
min-height	2-3						◎	◎	◎	◎	◎	●
min-width	2-3						●	●	●	●	●	●
orphans	2-3											
outline	2-3, TV			◎			●	●		●[2]	◎	
outline-color	2-3, TV			◎			●	●		●[2]	◎	
outline-style	2-3, TV			◎			●	●		●[2]	◎	
outline-width	2-3, TV			◎			●	●		●[2]	◎	

Tabelle A.21: Der CSS 2.0-Standard (Forts.)

CSS-Eigenschaft	CSS-Version	Palm	NN	Internet Explorer			Opera		Mozilla		Safari/ Konq	iCab
		2.x	4.x	5.x (Mac)	6 (Win)	7 (Win)	7	8	0.x+	1.0+	2.0+/ 3.x+	3.0
overflow	2-3	◉		⊙	●	●	●	●	●	●	●	●
padding	1-3, TV, Mobile	⊙	●	○	●	●	●	●	●	●	●	●
page	2, 3											
page-break-after	2, 3			●	●	●	●	●		●[1]	●	
page-break-before	2, 3			●	●	●	●	●		●[1]	●	
page-break-inside	2, 3				●	●	●	●				
pause	2-3											
pause-after	2-3											
pause-before	2-3											
pitch	2-3											
pitch-range	2-3											
play-during	2-3											
position	2-3, TV	⊙		●	◉	⊙	●	●	●	●	●	●
quotes	2-3			⊙			◉	◉	◉	◉	⊙	◉
richness	2-3											
right	2-3, TV	●		○	●	●	●	●	●	●	●	●
size	2, 3			⊙			⊙	⊙				
speak	2-3											
speak-header	2, 3											
speak-numeral	2, 3											
speak-punctuation	2, 3											
speech-rate	2-3											
stress	2-3											
table-layout	2-3			●	●	●	●	●	●	●	●	●

Tabelle A.21: Der CSS 2.0-Standard (Forts.)

CSS-Eigenschaft	CSS-Version	Palm	NN	Internet Explorer			Opera		Mozilla		Safari/ Konq	iCab
		2.x	4.x	5.x (Mac)	6 (Win)	7 (Win)	7	8	0.x+	1.0+	2.0+/ 3.x+	3.0
text-align	2, 3, TV, Mobile		◉	◉	◉	◉	◉	◉	◉	◉	◉	◉
text-align	1-3, TV, Mobile	◉	◉	◉	●	●	●	●	●	●	●	●
text-decoration	1-3, TV, Mobile	●	◉	◉	●	◉	●	●	●	●	●	●
text-indent	1-3, TV, Mobile	◉	⊙	●	●	●	●	●	●	●	●	●
text-shadow	2, 3										◉	
text-transform	1-3, TV, Mobile		⊙	●	●	●	●	●	●	●	●	●
top	2-3, TV			●	●	●	●	●	●	●	●	●
vertical-align	2, 3, TV, Mobile	●	◉	●	●	●	●	●	●	●	●	●
visibility	2-3, TV, Mobile	◉	◉	◉	◉		◉	◉	●	●	◉	◉
voice-family	2-3											
volume	2-3											
white-space	1-3, TV, Mobile	⊙	⊙	●	●	●	●	●	●	●	●	●
widows	2-3											
width	1-3, TV, Mobile	●	◉	●	●	●	●	●	●	●	◉	●
word-spacing	1-3			●	●	●	●	●	●	●	●	●
z-index	2-3, TV	◉	●	●	●	●	●	●	●	●	●	●

[1] ab Version 1.4+
[2] ab Version 1.8+

A.20 Der CSS 2.1-Standard

Tabelle A.22: Der CSS 2.1-Standard

CSS-Eigenschaft	CSS-Version	Palm	NN	Internet Explorer			Opera		Mozilla		Safari/ Konq	iCab
		2.x	4.x	5.x (Mac)	6 (Win)	7 (Win)	7	8	0.x+	1.0 +	2.0+/ 3.x+	3.0
:active	1-3, TV, Mobile			⊙	⊙	⊙	●	●	●	●	●	●
:after	2-3						⊙	⊙		●[1]	⊙	●
:before	2-3						⊙	⊙		●[1]	⊙	●
:first	2-3						●	●				
:first-child	2-3, TV			●			●	●		●	●	●
:first-letter	1-3, TV			●			●	●		●[1]	●	●
:first-line	1-3, TV			●			●	●		●[1]	●	●
:focus	2-3, TV, Mobile			⊙			⊙	⊙	●	●	⊙	⊙
:hover	1-3			⊙	⊙	⊙	●	●	●	●	●	●
:lang()	2-3			●			●	●		●[1]		●
:left	2-3						●	●				
:link	1-3, TV, Mobile	●	●	●	●	●	●	●	●	●	●	●
:right	2-3						●	●				
:visited	1-3, TV, Mobile	●	●	●	●	●	●	●	●	●	●	●
azimuth	2-3											
background	1-3, TV, Mobile	●	●	●	●	●	●	●	●	●	●	●
background-attachment	1-3	●		●	●	●	●	●	●	●	●	●
background-color	1-3, TV, Mobile	●	○	●	●	●	●	●	●	●	●	●

CSS-Eigenschaft	CSS-Version	Palm	NN	Internet Explorer			Opera		Mozilla		Safari/ Konq	iCab
		2.x	4.x	5.x (Mac)	6 (Win)	7 (Win)	7	8	0.x+	1.0 +	2.0+/ 3.x+	3.0
background-image	1-3, TV, Mobile	○	○	●	●	●	●	●	●	●	●	●
background-position	1-3, TV, Mobile	●		●	●	●	●	●	●	●	●	●
background-repeat	1-3, TV, Mobile	●	○	●	●	●	●	●	●	●	●	●
border	1-3, TV, Mobile	●	●	●	●	●	●	●	●	●	●	●
border-bottom	1-3, TV, Mobile	●		●	●	●	●	●	●	●	●	●
border-bottom-color	1-3, TV, Mobile	●	●	●	●	●	●	●	●	●	●	●
border-bottom-style	1-3, TV, Mobile	◘	○	⊙	⊙	⊙	●	●	●	●	●	●
border-bottom-width	1-3, TV, Mobile	●	○	●	●	●	●	●	●	●	●	●
border-collapse	2-3	●			●	●	●	●		●[1]	●	●
border-color	1-3, TV, Mobile	●	●	●	●	●	●	●	●	●	●	●
border-left	1-3, TV, Mobile	●		●	●	●	●	●	●	●	●	●
border-left-color	1-3, TV, Mobile	●	●	●	●	●	●	●	●	●	●	●
border-left-style	1-3, TV, Mobile	◘	○	⊙	⊙	⊙	●	●	●	●	●	●
border-left-width	1-3, TV, Mobile	●	○	●	●	●	●	●	●	●	●	●

Tabelle A.22: Der CSS 2.1-Standard (Forts.)

Tabelle A.22: Der CSS 2.1-Standard (Forts.)

CSS-Eigenschaft	CSS-Version	Palm	NN	Internet Explorer			Opera		Mozilla		Safari/ Konq	iCab
		2.x	4.x	5.x (Mac)	6 (Win)	7 (Win)	7	8	0.x+	1.0 +	2.0+/ 3.x+	3.0
border-right	1-3, TV, Mobile	●		●	●	●	●	●	●	●	●	●
border-right-color	1-3, TV, Mobile	●	●	●	●	●	●	●	●	●	●	●
border-right-style	1-3, TV, Mobile	◉	○	⊙	⊙	⊙	●	●	●	●	●	●
border-right-width	1-3, TV, Mobile	●	○	●	●	●	●	●	●	●	●	●
border-spacing	2-3						◉	◉		◉[1]	◉	●
border-style	1-3, TV, Mobile	◉	○	⊙	⊙	⊙	●	●	●	●	●	●
border-style	1-3, TV, Mobile	◉	○	⊙	⊙	⊙	●	●	●	●	●	●
border-top	1-3, TV, Mobile	●		●	●	●	●	●	●	●	●	●
border-top-color	1-3, TV, Mobile	●	●	●	●	●	●	●	●	●	●	●
border-top-style	1-3, TV, Mobile	◉	○	⊙	⊙	⊙	●	●	●	●	●	●
border-top-width	1-3, TV, Mobile	●	○	●	●	●	●	●	●	●	●	●
border-width	1-3, TV, Mobile	●	○	●	●	●	●	●	●	●	●	●
bottom	2-3, TV			◉	◉	◉	●	●	●	●	●	●
caption-side	2-3	◉		◉			◉	◉	◉	●[1]	◉	◉
clear	1-3, TV, Mobile	●	●	●	●		●	●	●	●	●	●

CSS-Eigenschaft	CSS-Version	Palm	NN	Internet Explorer			Opera		Mozilla		Safari/ Konq	iCab
		2.x	4.x	5.x (Mac)	6 (Win)	7 (Win)	7	8	0.x+	1.0 +	2.0+/ 3.x+	3.0
clip	2-3			●	●	●	●	●	●	●	●	●
color	1-3, TV, Mobile	◉	●	●	●	●	●	●	●	●	●	●
content	2-3						⊙	⊙	◉	◉	◉	●
counter-increment	2-3						●	●		◉[2]		●
counter-reset	2-3						●	●		◉[2]		●
cursor	2-3			◉	◉		◉	◉	◉	◉	◉	◉
display	1-3, TV, Mobile	◉	◉	◉	◉	◉	◉	◉	◉	◉	◉	◉
elevation	2-3											
empty-cells	2-3			⊙			●	●	●	●	●	●
float	1-3, TV, Mobile	⊙		●	●	●	●	●	●	●	●	●
font-family	1-3, TV, Mobile		●	●	●	●	●	●	●	●	●	●
font-size	1-3, TV, Mobile	◉	●	●	●	●	●	●	●	●	●	●
font-stretch	1, 2, 3											●
font-style	1-3, TV, Mobile		◉	●	●	●	●	●	●	●	●	●
font-variant	1-3, TV, Mobile			●	●	●	●	●	●	●	●	●
font-weight	1-3, TV, Mobile	◉	◉	●	●	●	●	●	●	●	●	●
height	1-3, TV, Mobile			●	◉	◉	◉	◉	◉	◉	◉	●

Tabelle A.22: Der CSS 2.1-Standard (Forts.)

Tabelle A.22: Der CSS 2.1-Standard (Forts.)

CSS-Eigenschaft	CSS-Version	Palm	NN	Internet Explorer			Opera		Mozilla		Safari/ Konq	iCab
		2.x	4.x	5.x (Mac)	6 (Win)	7 (Win)	7	8	0.x+	1.0 +	2.0+/ 3.x+	3.0
left	2-3, TV	●	◎	○	●	●	●	●	●	●	●	●
letter-spacing	1-3			●	●	●	●	●	●	●	●	●
line-height	1-3, TV	●	●	●	●	●	●	●	●	●	●	●
list-style	1-3, TV, Mobile	●	●	●	●	●	●	●	●	●	●	●
list-style-image	1-3, TV, Mobile	●		●	●	●	●	●	●	●	●	●
list-style-position	1-3, TV, Mobile	◎	◎	◎	◎	◎	◎	◎	◎	◎	◎	●
list-style-type	1-3, TV, Mobile	◎	◎	◎	◎	◎	◎	◎	◎	◎	◎	◎
margin	1-3, TV, Mobile	⊙	●	●	●	●	●	●	●	●	●	●
margin	2-3						⊙	⊙				
marks	2, 3											
max-height	2-3						◎	◎	◎	◎	◎	●
max-width	2-3						●	●	●	●	●	●
min-height	2-3						◎	◎	◎	◎	◎	●
min-width	2-3						●	●	●	●	●	●
orphans	2-3											
outline	2-3, TV			◎			●	●		●[2]	◎	
outline-color	2-3, TV			◎			●	●		●[2]	◎	
outline-style	2-3, TV			◎			●	●		●[2]	◎	
outline-width	2-3, TV			◎			●	●		●[2]	◎	

CSS-Eigenschaft	CSS-Version	Palm	NN	Internet Explorer			Opera		Mozilla		Safari/ Konq	iCab
		2.x	4.x	5.x (Mac)	6 (Win)	7 (Win)	7	8	0.x+	1.0 +	2.0+/ 3.x+	3.0
overflow	2-3	◉		⊙	●	●	●	●	●	●	●	●
padding	1-3, TV, Mobile	⊙	●	○	●	●	●	●	●	●	●	●
pause	2-3											
pause-after	2-3											
pause-before	2-3											
pitch	2-3											
pitch-range	2-3											
play-during	2-3											
position	2-3, TV	⊙		●	◉	⊙	●	●	●	●	●	●
quotes	2-3			⊙			◉	◉	◉	◉	⊙	◉
richness	2-3											
right	2-3, TV	●		○	●	●	●	●	●	●	●	●
speak	2-3											
speech-rate	2-3											
stress	2-3											
table-layout	2-3			●	●	●	●	●	●	●	●	●
text-align	1-3, TV, Mobile	◉	◉	◉	●	●	●	●	●	●	●	●
text-decoration	1-3, TV, Mobile	●	◉	◉	●	◉	●	●	●	●	●	●
text-indent	1-3, TV, Mobile	◉	⊙	●	●	●	●	●	●	●	●	●
text-transform	1-3, TV, Mobile		⊙	●	●	●	●	●	●	●	●	●
top	2-3, TV			●	●	●	●	●	●	●	●	●

Tabelle A.22: Der CSS 2.1-Standard (Forts.)

Tabelle A.22: Der CSS 2.1-Standard (Forts.)

CSS-Eigenschaft	CSS-Version	Palm	NN	Internet Explorer			Opera		Mozilla		Safari/ Konq	iCab
		2.x	4.x	5.x (Mac)	6 (Win)	7 (Win)	7	8	0.x+	1.0 +	2.0+/ 3.x+	3.0
vertical-align	2, 3, TV, Mobile	●	◎	●	●	●	●	●	●	●	●	●
visibility	2-3, TV, Mobile	◎	◎	◎	◎		◎	◎	●	●	◎	◎
voice-family	2-3											
volume	2-3											
white-space	1-3, TV, Mobile	⊙	◎	◎	◎	◎	◎	●	◎	◎	●	●
widows	2-3											
width	1-3, TV, Mobile	●	⊙	●	●	●	●	●	●	●	◎	●
word-spacing	1-3			●	●	●	●	●	●	●	●	●
z-index	2-3, TV	◎	●	●	●	●	●	●	●	●	●	●

[1] ab Version 1.4+
[2] ab Version 1.8+

A.21 Erläuterte neue CSS 3.0-Eigenschaften

Bitte bedenken Sie, dass CSS 3.0 noch nicht verabschiedet ist. Daher kann sich noch viel ändern. Zudem stellen die nachfolgenden Eigenschaften und Elemente nur einen Auszug aus CSS 3.0 dar.

Tabelle A.23: Erläuterte neue CSS 3.0-Eigenschaften

CSS-Eigenschaft	CSS-Version	Palm	NN	Internet Explorer			Opera		Mozilla		Safari/ Konq	iCab
		2.x	4.x	5.x (Mac)	6 (Win)	7 (Win)	7	8	0.x+	1.0+	2.0+/ 3.x+	3.0
:not	3									●[1]	●	
border-bottom-left-radius	3											
border-break	3											
border-radius	3											

CSS-Eigenschaft	CSS-Version	Palm	NN	Internet Explorer			Opera		Mozilla		Safari/ Konq	iCab
		2.x	4.x	5.x (Mac)	6 (Win)	7 (Win)	7	8	0.x+	1.0+	2.0+/ 3.x+	3.0
border-top-left-radius	3											
border-top-right-radius	3											
border-top-right-radius	3											
box-shadow	3											
font-effect	3											
opacity	3									●[1]	●	
text-align-last	3				○	○						
text-justify	3				●	●						

[1] ab Version 1.4+
[2] ab Version 1.8+

Tabelle A.23: Erläuterte neue CSS 3.0-Eigenschaften (Forts.)

A.22 Die CSS-Mobile-Spezifikation

CSS-Eigenschaft	CSS-Version	Palm	NN	Internet Explorer			Opera		Mozilla		Safari/ Konq	iCab
		2.x	4.x	5.x (Mac)	6 (Win)	7 (Win)	7	8	0.x+	1.0+	2.0+/ 3.x+	3.0
:active	1-3, TV, Mobile			○	○	○	●	●	●	●	●	●
:focus	2-3, TV, Mobile			○			○	○	●	●	○	○
:link	1-3, TV, Mobile	●	●	●	●	●	●	●	●	●	●	●
:visited	1-3, TV, Mobile	●	●	●	●	●	●	●	●	●	●	●
background	1-3, TV, Mobile	●	●	●	●	●	●	●	●	●	●	●

Tabelle A.24: Die CSS-Mobile-Spezifikation

Tabelle A.24: Die CSS-Mobile-Spezifikation (Forts.)

CSS-Eigenschaft	CSS-Version	Palm	NN	Internet Explorer			Opera		Mozilla		Safari/ Konq	iCab
		2.x	4.x	5.x (Mac)	6 (Win)	7 (Win)	7	8	0.x+	1.0+	2.0+/ 3.x+	3.0
background-color	1-3, TV, Mobile	●	○	●	●	●	●	●	●	●	●	●
background-image	1-3, TV, Mobile	○	○	●	●	●	●	●	●	●	●	◉
background-position	1-3, TV, Mobile	●		●	●	●	●	●	●	●	●	●
background-repeat	1-3, TV, Mobile	●	○	●	●	●	●	●	●	●	●	●
border	1-3, TV, Mobile	●	●	●	●	●	●	●	●	●	●	●
border-bottom	1-3, TV, Mobile	●		●	●	●	●	●	●	●	●	●
border-bottom-color	1-3, TV, Mobile	●	●	●	●	●	●	●	●	●	●	●
border-bottom-style	1-3, TV, Mobile	◉	○	⊙	⊙	⊙	●	●	●	●	●	●
border-bottom-width	1-3, TV, Mobile	●	○	●	●	●	●	●	●	●	●	●
border-color	1-3, TV, Mobile	●	●	●	●	●	●	●	●	●	●	●
border-left	1-3, TV, Mobile	●		●	●	●	●	●	●	●	●	●
border-left-color	1-3, TV, Mobile	●	●	●	●	●	●	●	●	●	●	●
border-left-style	1-3, TV, Mobile	◉	○	⊙	⊙	⊙	●	●	●	●	●	●

Tabelle A.24: Die CSS-Mobile-Spezifikation (Forts.)

CSS-Eigenschaft	CSS-Version	Palm	NN	Internet Explorer			Opera		Mozilla		Safari/ Konq	iCab
		2.x	4.x	5.x (Mac)	6 (Win)	7 (Win)	7	8	0.x+	1.0+	2.0+/ 3.x+	3.0
border-left-width	1-3, TV, Mobile	●	○	●	●	●	●	●	●	●	●	●
border-right	1-3, TV, Mobile	●		●	●	●	●	●	●	●	●	●
border-right-color	1-3, TV, Mobile	●	●	●	●	●	●	●	●	●	●	●
border-right-style	1-3, TV, Mobile	◉	○	⊙	⊙	⊙	●	●	●	●	●	●
border-right-width	1-3, TV, Mobile	●	○	●	●	●	●	●	●	●	●	●
border-style	1-3, TV, Mobile	◉	○	⊙	⊙	⊙	●	●	●	●	●	●
border-style	1-3, TV, Mobile	◉	○	⊙	⊙	⊙	●	●	●	●	●	●
border-top	1-3, TV, Mobile	●		●	●	●	●	●	●	●	●	●
border-top-color	1-3, TV, Mobile	●	●	●	●	●	●	●	●	●	●	●
border-top-style	1-3, TV, Mobile	◉	○	⊙	⊙	⊙	●	●	●	●	●	●
border-top-width	1-3, TV, Mobile	●	○	●	●	●	●	●	●	●	●	●
border-width	1-3, TV, Mobile	●	○	●	●	●	●	●	●	●	●	●
clear	1-3, TV, Mobile	●	●	●	●		●	●	●	●	●	●

Tabelle A.24: Die CSS-Mobile-Spezifikation (Forts.)

CSS-Eigenschaft	CSS-Version	Palm	NN	Internet Explorer			Opera		Mozilla		Safari/ Konq	iCab
		2.x	4.x	5.x (Mac)	6 (Win)	7 (Win)	7	8	0.x+	1.0+	2.0+/ 3.x+	3.0
color	1-3, TV, Mobile	◉	●	●	●	●	●	●	●	●	●	●
display	1-3, TV, Mobile	◉	◉	◉	◉	◉	◉	◉	◉	◉	◉	◉
float	1-3, TV, Mobile	⊙		●	●	●	●	●	●	●	●	●
font-family	1-3, TV, Mobile		●	●	●	●	●	●	●	●	●	●
font-size	1-3, TV, Mobile	◉	●	●	●	●	●	●	●	●	●	●
font-style	1-3, TV, Mobile		◉	●	●	●	●	●	●	●	●	●
font-variant	1-3, TV, Mobile			●	●	●	●	●	●	●	●	●
font-weight	1-3, TV, Mobile	◉	◉	●	●	●	●	●	●	●	●	●
height	1-3, TV, Mobile			●	◉	◉	◉	◉	◉	◉	◉	●
list-style	1-3, TV, Mobile	●	●	●	●	●	●	●	●	●	●	●
list-style-image	1-3, TV, Mobile	●		●	●	●	●	●	●	●	●	●
list-style-position	1-3, TV, Mobile	◉	◉	◉	◉	◉	◉	◉	◉	◉	◉	●
list-style-type	1-3, TV, Mobile	◉	◉	◉	◉	◉	◉	◉	◉	◉	◉	◉

Tabelle A.24: Die CSS-Mobile-Spezifikation (Forts.)

CSS-Eigenschaft	CSS-Version	Palm	NN	Internet Explorer			Opera		Mozilla		Safari/ Konq	iCab
		2.x	4.x	5.x (Mac)	6 (Win)	7 (Win)	7	8	0.x+	1.0+	2.0+/ 3.x+	3.0
margin	1-3, TV, Mobile	⊙	●	●	●	●	●	●	●	●	●	●
padding	1-3, TV, Mobile	⊙	●	○	●	●	●	●	●	●	●	●
text-align	2, 3, TV, Mobile		◉	◉	◉	◉	◉	◉	◉	◉	◉	◉
text-align	1-3, TV, Mobile	◉	◉	◉	●	●	●	●	●	●	●	●
text-decoration	1-3, TV, Mobile	●	◉	◉	●	◉	●	●	●	●	●	●
text-indent	1-3, TV, Mobile	◉	⊙	●	●	●	●	●	●	●	●	●
text-transform	1-3, TV, Mobile		⊙	●	●	●	●	●	●	●	●	●
vertical-align	2, 3, TV, Mobile	●	◉	●	●	●	●	●	●	●	●	●
visibility	2-3, TV, Mobile	◉	◉	◉	◉		◉	◉	●	●	◉	◉
white-space	1-3, TV, Mobile	⊙	⊙	●	●	●	●	●	●	●	●	●
width	1-3, TV, Mobile	●	⊙	●	●	●	●	●	●	●	◉	●

[1] ab Version 1.4+
[2] ab Version 1.8+

A.23 Die CSS-TV-Spezifikation

Tabelle A.25: Die CSS-TV-Spezifikation

CSS-Eigenschaft	CSS-Version	Palm	NN	Internet Explorer			Opera		Mozilla		Safari/ Konq	iCab
		2.x	4.x	5.x (Mac)	6 (Win)	7 (Win)	7	8	0.x+	1.0+	2.0+/ 3.x+	3.0
:active	1-3, TV, Mobile			◎	◎	◎	●	●	●	●	●	●
:first-child	2-3, TV			●			●	●		●	●	●
:first-letter	1-3, TV			●			●	●		●[1]	●	●
:first-line	1-3, TV			●			●	●		●[1]	●	●
:focus	2-3, TV, Mobile			◎			◎	◎	●	●	◎	◎
:link	1-3, TV, Mobile	●	●	●	●	●	●	●	●	●	●	●
:visited	1-3, TV, Mobile	●	●	●	●	●	●	●	●	●	●	●
background	1-3, TV, Mobile	●	●	●	●	●	●	●	●	●	●	●
background-color	1-3, TV, Mobile	●	○	●	●	●	●	●	●	●	●	●
background-image	1-3, TV, Mobile	○	○	●	●	●	●	●	●	●	●	●
background-position	1-3, TV, Mobile	●		●	●	●	●	●	●	●	●	●
background-repeat	1-3, TV, Mobile	●	○	●	●	●	●	●	●	●	●	●
border	1-3, TV, Mobile	●	●	●	●	●	●	●	●	●	●	●

Tabelle A.25: Die CSS-TV-Spezifikation (Forts.)

CSS-Eigenschaft	CSS-Version	Palm	NN	Internet Explorer			Opera		Mozilla		Safari/ Konq	iCab
		2.x	4.x	5.x (Mac)	6 (Win)	7 (Win)	7	8	0.x+	1.0+	2.0+/ 3.x+	3.0
border-bottom	1-3, TV, Mobile	●		●	●	●	●	●	●	●	●	●
border-bottom-color	1-3, TV, Mobile	●	●	●	●	●	●	●	●	●	●	●
border-bottom-style	1-3, TV, Mobile	◉	○	⊙	⊙	⊙	●	●	●	●	●	●
border-bottom-width	1-3, TV, Mobile	●	○	●	●	●	●	●	●	●	●	●
border-color	1-3, TV, Mobile	●	●	●	●	●	●	●	●	●	●	●
border-left	1-3, TV, Mobile	●		●	●	●	●	●	●	●	●	●
border-left-color	1-3, TV, Mobile	●	●	●	●	●	●	●	●	●	●	●
border-left-style	1-3, TV, Mobile	◉	○	⊙	⊙	⊙	●	●	●	●	●	●
border-left-width	1-3, TV, Mobile	●	○	●	●	●	●	●	●	●	●	●
border-right	1-3, TV, Mobile	●		●	●	●	●	●	●	●	●	●
border-right-color	1-3, TV, Mobile	●	●	●	●	●	●	●	●	●	●	●
border-right-style	1-3, TV, Mobile	◉	○	⊙	⊙	⊙	●	●	●	●	●	●
border-right-width	1-3, TV, Mobile	●	○	●	●	●	●	●	●	●	●	●

Tabelle A.25: Die CSS-TV-Spezifikation (Forts.)

CSS-Eigenschaft	CSS-Version	Palm	NN	Internet Explorer			Opera		Mozilla		Safari/ Konq	iCab
		2.x	4.x	5.x (Mac)	6 (Win)	7 (Win)	7	8	0.x+	1.0+	2.0+/ 3.x+	3.0
border-style	1-3, TV, Mobile	◉	○	⊙	⊙	⊙	●	●	●	●	●	●
border-style	1-3, TV, Mobile	◉	○	⊙	⊙	⊙	●	●	●	●	●	●
border-top	1-3, TV, Mobile	●		●	●	●	●	●	●	●	●	●
border-top-color	1-3, TV, Mobile	●	●	●	●	●	●	●	●	●	●	●
border-top-style	1-3, TV, Mobile	◉	○	⊙	⊙	⊙	●	●	●	●	●	●
border-top-width	1-3, TV, Mobile	●	○	●	●	●	●	●	●	●	●	●
border-width	1-3, TV, Mobile	●	○	●	●	●	●	●	●	●	●	●
bottom	2-3, TV			◉	◉	◉	●	●	●	●	●	●
clear	1-3, TV, Mobile	●	●	●	●		●	●	●	●	●	●
color	1-3, TV, Mobile	◉	●	●	●	●	●	●	●	●	●	●
display	1-3, TV, Mobile	◉	◉	◉	◉	◉	◉	◉	◉	◉	◉	◉
float	1-3, TV, Mobile	⊙		●	●	●	●	●	●	●	●	●
font-family	1-3, TV, Mobile		●	●	●	●	●	●	●	●	●	●
font-size	1-3, TV, Mobile	◉	●	●	●	●	●	●	●	●	●	●

CSS-Eigenschaft	CSS-Version	Palm	NN	Internet Explorer			Opera		Mozilla		Safari/ Konq	iCab
		2.x	4.x	5.x (Mac)	6 (Win)	7 (Win)	7	8	0.x+	1.0+	2.0+/ 3.x+	3.0
font-style	1-3, TV, Mobile		⊙	●	●	●	●	●	●	●	●	●
font-variant	1-3, TV, Mobile			●	●	●	●	●	●	●	●	●
font-weight	1-3, TV, Mobile	⊙	⊙	●	●	●	●	●	●	●	●	●
height	1-3, TV, Mobile			●	⊙	⊙	⊙	⊙	⊙	⊙	⊙	●
left	2-3, TV	●	⊙	○	●	●	●	●	●	●	●	●
line-height	1-3, TV	●	●	●	●	●	●	●	●	●	●	●
list-style	1-3, TV, Mobile	●	●	●	●	●	●	●	●	●	●	●
list-style-image	1-3, TV, Mobile	●		●	●	●	●	●	●	●	●	●
list-style-position	1-3, TV, Mobile	⊙	⊙	⊙	⊙	⊙	⊙	⊙	⊙	⊙	⊙	●
list-style-type	1-3, TV, Mobile	⊙	⊙	⊙	⊙	⊙	⊙	⊙	⊙	⊙	⊙	⊙
margin	1-3, TV, Mobile	⊙	●	●	●	●	●	●	●	●	●	●
outline	2-3, TV			⊙			●	●		●[2]	⊙	
outline-color	2-3, TV			⊙			●	●		●[2]	⊙	
outline-style	2-3, TV			⊙			●	●		●[2]	⊙	
outline-width	2-3, TV			⊙			●	●		●[2]	⊙	

Tabelle A.25: Die CSS-TV-Spezifikation (Forts.)

Tabelle A.25: Die CSS-TV-Spezifikation (Forts.)

CSS-Eigenschaft	CSS-Version	Palm	NN	Internet Explorer			Opera		Mozilla		Safari/ Konq	iCab
		2.x	4.x	5.x (Mac)	6 (Win)	7 (Win)	7	8	0.x+	1.0+	2.0+/ 3.x+	3.0
padding	1-3, TV, Mobile	⊙	●	○	●	●	●	●	●	●	●	●
position	2-3, TV	⊙		●	◉	⊙	●	●	●	●	●	●
right	2-3, TV	●		○	●	●	●	●	●	●	●	●
text-align	2, 3, TV, Mobile		◉	◉	◉	◉	◉	◉	◉	◉	◉	◉
text-align	1-3, TV, Mobile	◉	◉	◉	●	●	●	●	●	●	●	●
text-decoration	1-3, TV, Mobile	●	◉	◉	●	◉	●	●	●	●	●	●
text-indent	1-3, TV, Mobile	◉	⊙	●	●	●	●	●	●	●	●	●
text-transform	1-3, TV, Mobile		⊙	●	●	●	●	●	●	●	●	●
top	2-3, TV			◉	◉	◉	●	●	●	●	●	●
vertical-align	2, 3, TV, Mobile	●	◉	●	●	●	●	●	●	●	●	●
visibility	2-3, TV, Mobile	◉	◉	◉	◉		◉	◉	●	●	◉	◉
white-space	1-3, TV, Mobile	⊙	⊙	●	●	●	●	●	●	●	●	●
width	1-3, TV, Mobile	●	⊙	●	●	●	●	●	●	●	◉	●
z-index	2-3, TV	◉	●	●	●	●	●	●	●	●	●	●

[1] ab Version 1.4+
[2] ab Version 1.8+

A.24 CD-Inhalte

Tabelle A.26: Inhalt der Buch-CD

Programm	Beschreibung	Webseite/Hersteller
Screen Calipers 3.1	Virtuelle Schieblehre zum Messen von Abständen und Elementen auf dem Bildschirm. **Lizenz:** Eingeschränkt funktionsfähige Demo-Version **System:** Windows, 2000, XP oder NT	Iconico http://www.iconico.com
CSS Editor 1.0.8	CSS-Editor **Lizenz:** Testversion **System:** Windows	Thomas Rudolph http://www.css-maker.de
FireFox 1.0.6	Mozilla-Browser **Lizenz:** Open Source **System:** Windows, Mac OS X, Linux	Mozilla http://www.mozilla.org
Mozilla 1.7.x	Mozilla Browser **Lizenz:** Open Source **System:** Windows, Mac OS X	Mozilla http://www.mozilla.org
Opera 8.x	Opera Browser **Lizenz:** Shareware **System:** Windows (deutsch), Mac OS X (englisch)	Opera Software ASA http://www.opera.com
iCab 3.0 Beta	iCab-Browser **Lizenz:** Shareware **System:** Mac OS X 10.2+ (Auf der Webseite sind auch Versionen für frühere Mac-OS-Versionen verfügbar)	Alexander Clauss http://www.icab.de
Buch-Beispiele	Alle Beispiele zum Buch, in Form von Testseiten für einzelne CSS-Eigenschaften. Alle Beispiele können Sie über die Startseite index.html aufrufen.	

B Glossar

Attribut

Ein Attribut wird im Anfangs-Tag- eines (X)HTML-Elements notiert und beschreibt das Element.

Barrierefreiheit

Als barrierefrei gilt eine Webseite dann, wenn sie so gestaltet ist, dass alle Ausgabeprogramme, die (X)HTML beherrschen, die Inhalte der Seite darstellen können. Barrierefrei bedeutet also keinesfalls, dass die Seite in allen Browsern gleich aussehen muss, sondern nur, dass sie auch behinderten Besuchern zugänglich ist. Diese Besucher nutzen oft besondere Programme oder Hardware, wie Screenreader oder Braillezeilen, die die Inhalte vorlesen (Screenreader) oder in Blindenschrift darstellen (Braillezeilen). Auf diesen Ausgabegeräten spielt die Formatierung dann keine Rolle. Für sie ist nur wichtig, dass der HTML-Code strukturiert und korrekt ist.

Blockelemente

Als Elemente auf Blockebene bzw. Blockelemente werden alle (X)HTML-Elemente bezeichnet, die wie Absätze und Überschriften als eine visuelle Einheit angezeigt werden. Weitere Elemente können zu Blockelementen gemacht werden, indem Sie die `display`-Eigenschaft auf `block`, `list-item`, `compact` oder `run-in` setzen. Elemente auf Blockebene erzeugen eine Blockbox, die als Hauptblockbox bezeichnet wird. Sie definiert weitere abgeleitete Boxen für den Inhalt.

Browserweiche

Als Browserweiche wird Code oder ein Skript bezeichnet, das in der Lage ist, abhängig vom Browser bestimmte Ausgaben bzw. Formatierungen zu erzielen. Browserweichen können per CSS, per JavaScript oder mit einer serverseitigen Skriptsprache erstellt werden.

CSS

CSS ist die Abkürzung für Cascading Style Sheets. Dabei handelt es sich um eine Sammlung von Eigenschaften, Werten und Syntaxregeln, die dazu dienen Webseiten zu formatieren.

Generische Schriftfamilien

Generische Schriftfamilien sind CSS-Schlüsselwörter, die einen Satz von ähnlichen Schriften definieren. In CSS sind fünf generische Schriftfamilien definiert: `serif`, `sans-serif`, `monospace`, `cursive` und `fantasy`.

Grundlinie

Die Grundlinie eines Textes ist die Linie, auf der alle Buchstaben angeordnet werden. Buchstaben wie »j«, »g« etc. ragen nach unten jedoch über die Grundlinie hinaus. Die untere Kante dieser Buchstaben stellt die untere Textkante dar, die obere Kante der Großbuchstaben und der hohen Kleinbuchstaben wie »l« und »k« stellen die obere Textkante dar.

HTML-Element

Ein HTML-Element ist ein Teil des Sprachumfangs von HTML, das aus einem Anfangs- und einem Endtag besteht, zwischen denen der Inhalt des Elements stehen kann. In Ausnahmefällen gibt es weder Inhalt noch Endtag und das Element besteht ausschließlich aus dem einleitenden Tag.

Hurenkinder

Von einem Hurenkind bzw. einer Witwe spricht man, wenn nach einem Seitenumbruch die letzte Zeile eines Absatzes allein auf einer Seite steht.

Kapitälchen

Kapitälchen sehen aus wie Großbuchstaben, werden jedoch als Kleinbuchstaben im Text in einer geringeren Größe oder zumindest einer geringeren Höhe angezeigt, sofern die Schriftart nicht explizit einen Zeichensatz für Kapitälchen zur Verfügung stellt. DIES IST EIN TEXT IN KAPITÄLCHEN.

Klangumgebung

Als Klangumgebung wird ein virtueller akustischer Raum bezeichnet, innerhalb dessen die Sprachausgabe einer Seite erfolgt.

Operatoren

Operatoren dienen dazu, aus den einfachen Selektoren komplexere zusammenzusetzen. Sie können damit ganz speziell bestimmte Kombinationen und Hierarchien von HTML-Elementen ansprechen.

Positionierung

Der Begriff »Positionierung« umfasst die CSS-Befehle (vornehmlich aus CSS 2.0), die zur Positionierung von Elementen auf der Seite verwendet werden. Darunter fallen auch CSS-Eigenschaften zur Bestimmung der Größe und der Stapelreihenfolge sowie zur Anordnung der Elemente.

Pseudo-Elemente

Pseudo-Elemente sind solche Elemente, die zwar nicht über den (X)HTML-Dokumentenbaum entstehen, aber zumindest davon abgeleitet werden können.

Pseudo-Klassen

Pseudo-Klassen entsprechen immer Elementen, die im Dokumentenbaum definiert sind. Allerdings werden diese klassifiziert und zwar nach bestimmten Eigenschaften. Dabei handelt es sich normalerweise um Eigenschaften, die sich nicht vom Dokumentenbaum ableiten lassen, wie z.B. ob ein Link bereits besucht wurde, gehovert wird oder den Fokus hat.

Rendering-Engine

Als Rendering-Engine wird der Programmteil eines Browsers bezeichnet, der für die Auswertung des (X)HTML- und CSS-Codes sorgt und diesen entsprechend den gültigen Standards in eine grafische Ausgabe umsetzt. Allein die Rendering-Engine des Browsers bestimmt also die Qualität der Ausgabe.

Scalable Vector Graphics (SVG)

SVG ist eine auf XML basierende Markup-Sprache zur Erzeugung von Vektorgrafiken. Auch darin erfolgt die Formatierung der Grafikelemente mit Hilfe von CSS, wenngleich die CSS-Befehle vom CSS 2.0-Standard etwas abweichen.

Schusterjungen

Von einem Schusterjungen spricht man, wenn nach einem Seitenumbruch die erste Zeile eines Absatzes allein auf der letzten Seite zurückbleibt.

Screenreader

Ein Screenreader ist ein Programm, das Blinden und Sehbehinderten den Umgang mit dem PC ermöglicht, indem es den Bildschirminhalt vorliest. Screenreader sind gleichzusetzen mit Browsern, die eine Sprachausgabe ermöglichen. Während ein Screenreader den gesamten Bildschirminhalt vorliest, liest ein Browser mit Sprachausgabe den Inhalt der Webseite vor. Das kann auch unsichtbare Teile umfassen und sollte im optimalen Fall auch zur Berücksichtigung des CSS-Codes führen.

Selektor

Ein Selektor definiert, für welche Elemente der Webseite ein Stil gilt. Als Selektor gilt alles das, was vor der ersten geschweiften Klammer steht. Der Inhalt des Stils einschließlich der umgebenden geschweiften Klammern ist der Deklarationsblock. Selektoren dürfen immer nur zusammen mit einem Deklarationsblock auftreten.

Small Screen Rendering (SSR)

SSR stellt eine Entwicklung von Opera dar, die es ermöglicht, normale Webseiten auch auf kleinen Displays, z.B. von Handys und PDAs, darzustellen.

Stil

Ein Stil bzw. CSS-Stil ist ein Selektor mit Deklarationsblock. Der Deklarationsblock umfasst die CSS-Eigenschaften und deren Werte und steht in geschweiften Klammern. Der Selektor steht vor dem Deklarationsblock und definiert, auf welche Elemente der Seite der Stil angewendet wird.

Stimmvolumen

Der Stimmumfang bzw. das Stimmvolumen bestimmt, wie gut die Stimme in einem größeren Raum trägt bzw. wie sanft eine Stimme ist.

Stylesheet

Als Stylesheet wird die Gesamtheit aller CSS-Stile bezeichnet, die für ein HTML-Dokument gelten. Das können Stile sein, die Sie direkt in der Seite definieren, sowie verknüpfte und importierte CSS-Dateien.

Überschreibung

Mit Überschreibung ist ein Verhalten gemeint, das es ermöglicht, einmal definierte Eigenschaften durch eine Neudefinition etwas später im Stylesheet zu überschreiben. Neben der Reihenfolge, in der Sie die Stile definieren, spielt aber auch die Spezifität des Selektors eine Rolle, die sich berechnen lässt.

Umschließender Block

Beim visuellen Formatierungsmodell erzeugt jedes Element kein, ein oder mehrere Blöcke, die wiederum Boxen gemäß dem Boxmodell enthalten. Der Block der obersten Ebene ist der umschließende Block, der das gesamte (X)HTML-Dokument umfasst.

Valider Code

Valide ist Code dann, wenn er dem entsprechenden W3C-Standard genügt. Valider HTML-Code muss also dem mit der DocType-Angabe definierten Standard entsprechen, d.h. valider CSS-Code der CSS 1.0- oder CSS 2.x-Spezifikation. Um zu prüfen, ob der Code valide ist, können Sie so genannte Validatoren einsetzen. Sie prüfen den Code gemäß der verwendeten Standards und markieren Fehler bzw. führen sogar Verbesserungen durch.

Vererbung

Vererbung meint, dass CSS-Formatierungen, die Sie für ein bestimmtes Element festlegen, auch auf untergeordnete und verwandte Elemente angewendet werden. Wenn Sie Vererbung geschickt nutzen, können Sie sich damit eine Menge Aufwand sparen und Ihre Stylesheets noch wartungsfreundlicher gestalten.

W3C-Standard

Ein W3C-Standard ist kein Standard wie beispielsweise eine DIN-Norm, sondern nur eine Empfehlung des W3C (W3-Consortium). Allerdings machen diese Empfehlungen durchaus Sinn, weil sie den Browser- und Softwareherstellern Richtlinien an die Hand geben, wie mit bestimmten Teilen einer Webseite umgegangen werden soll.

WYSIWYG

WYSIWYG ist die Abkürzung für »What You See Is What You Get«, was übersetzt werden kann mit: »Was du siehst, bekommst du«. Gemeint sind damit Editoren, bei denen Sie bereits zum Zeitpunkt des Entwurfs ein Ergebnis sehen, das weitgehend der Darstellung im Browser entspricht.

Befehlsindex

Stichwortverzeichnis

B

C

D

O

P

Q

R

S

T

U

V

W

X

Z

Alphabetische Übersicht der CSS-Eigenschaften

Markt+Technik

CSS-Eigenschaft	CSS-Version	Palm	NN	Internet Explorer			Opera		Mozilla		Safari/Konq	iCab
		2.x	4.x	5.x (Mac)	6 (Win)	7 (Win)	7	8	0.x+	1.0+	2.0+/ 3.x+	3.0
:atactive	1-3, TV, Mobile			◎	◎	◎	●	●	●	●	●	●
:after	2-3						⊙	⊙		●[1]	⊙	●
:before	2-3						⊙	⊙		●[1]	⊙	●
:first	2-3						●	●				
:first-child	2-3, TV			●			●	●		●	●	●
:first-letter	1-3, TV			●			●	●		●[1]	●	●
:first-line	1-3, TV			●			●	●		●[1]	●	●
:focus	2-3, TV, Mobile			◎			◎	◎	●	●	◎	◎
:hover	1-3			◎	◎	◎	●	●	●	●	●	●
:lang()	2-3			●			●	●		●[1]		●
:left	2-3						●	●				
:link	1-3, TV, Mobile	●	●	●	●	●	●	●	●	●	●	●
:not	3									●[1]	●	
:right	2-3						●	●				
:visited	1-3, TV, Mobile	●	●	●	●	●	●	●	●	●	●	●
azimuth	2-3											
background	1-3, TV, Mobile	●	●	●	●	●	●	●	●	●	●	●
background-attachment	1-3	●		●	●	●	●	●	●	●	●	●
background-color	1-3, TV, Mobile	●	○	●	●	●	●	●	●	●	●	●
background-image	1-3, TV, Mobile	○	○	●	●	●	●	●	●	●	●	◎
background-position	1-3, TV, Mobile	●		●	●	●	●	●	●	●	●	●
background-repeat	1-3, TV, Mobile	●	○	●	●	●	●	●	●	●	●	●
border	1-3, TV, Mobile	●	●	●	●	●	●	●	●	●	●	●
border-bottom	1-3, TV, Mobile	●		●	●	●	●	●	●	●	●	●
border-bottom-color	1-3, TV, Mobile	●	●	●	●	●	●	●	●	●	●	●
border-bottom-left-radius	3											
border-bottom-style	1-3, TV, Mobile	◎	○	⊙	⊙	⊙	●	●	●	●	●	●
border-bottom-width	1-3, TV, Mobile	⊙	○	●	●	●	●	●	●	●	●	●
border-break	3											
border-collapse	2-3	●			●	●	●	●		●[1]	●	●
border-color	1-3, TV, Mobile	●	●	●	●	●	●	●	●	●	●	●
border-left	1-3, TV, Mobile	●		●	●	●	●	●	●	●	●	●
border-left-color	1-3, TV, Mobile	●	●	●	●	●	●	●	●	●	●	●
border-left-style	1-3, TV, Mobile	◎	○	⊙	⊙	⊙	●	●	●	●	●	●
border-left-width	1-3, TV, Mobile	⊙	○	●	●	●	●	●	●	●	●	●
border-radius	3											
border-right	1-3, TV, Mobile	●		●	●	●	●	●	●	●	●	●
border-right-color	1-3, TV, Mobile	●	●	●	●	●	●	●	●	●	●	●
border-right-style	1-3, TV, Mobile	◎	○	⊙	⊙	⊙	●	●	●	●	●	●
border-right-width	1-3, TV, Mobile	⊙	○	●	●	●	●	●	●	●	●	●
border-spacing	2-3						◎	◎		◎[1]	◎	●
border-style	1-3, TV, Mobile	◎	○	⊙	⊙	⊙	●	●	●	●	●	●
border-style	1-3, TV, Mobile	◎	○	⊙	⊙	⊙	●	●	●	●	●	●
border-top	1-3, TV, Mobile	●		●	●	●	●	●	●	●	●	●
border-top-color	1-3, TV, Mobile	●	●	●	●	●	●	●	●	●	●	●
border-top-left-radius	3											

Alphabetische Übersicht der CSS-Eigenschaften

Markt+Technik

CSS-Eigenschaft	CSS-Version	Palm	NN	Internet Explorer			Opera		Mozilla		Safari/Konq	iCab
		2.x	4.x	5.x (Mac)	6 (Win)	7 (Win)	7	8	0.x+	1.0+	2.0+/ 3.x+	3.0
border-top-right-radius	3											
border-top-right-radius	3											
border-top-style	1-3, TV, Mobile	◎	○	⊙	⊙	⊙	●	●	●	●	●	●
border-top-width	1-3, TV, Mobile	⊙	○	●	●	●	●	●	●	●	●	●
border-width	1-3, TV, Mobile	⊙	○	●	●	●	●	●	●	●	●	●
bottom	2-3, TV			●	●	●	●	●	●	●	●	●
box-shadow	3											
caption-side	2-3,	◎		◎			◎	◎	◎	●[1]	◎	◎
clear	1-3, TV, Mobile	●	●	●	●		●	●	●	●	●	●
clip	2, 2.1, 3			●	●	●	●	●	●	●	●	●
color	1-3, TV, Mobile	◎	●	●	●	●	●	●	●	●	●	●
content	2-3						⊙	⊙	◎	◎	◎	●
counter-increment	2-3						●	●		◎[2]		●
counter-reset	2-3						●	●		◎[2]		●
cue	2, 3											
cue-after	2, 3											
cue-before	2, 3											
cursor	2-3			◎	◎		◎	◎	◎	◎	◎	◎
display	1-3, Mobile, TV	◎	◎	◎	◎	◎	◎	◎	◎	◎	◎	◎
elevation	2-3											
empty-cells	2-3			⊙			●	●	●	●	●	●
float	1-3, TV, Mobile	⊙		●	●	●	●	●	●	●	●	●
font-effect	3											
font-family	1-3, TV, Mobile		●	●	●	●	●	●	●	●	●	●
font-size	1-3, TV, Mobile	◎	●	●	●	●	●	●	●	●	●	●
font-stretch	1, 2, 3											●
font-style	1-3, TV, Mobile	●	◎	●	●	●	●	●	●	●	●	●
font-variant	1-3, TV, Mobile			●	●	●	●	●	●	●	●	●
font-weight	1-3, TV, Mobile	◎	◎	●	●	●	●	●	●	●	●	●
height	1-3, TV, Mobile			●	◎	◎	◎	◎	◎	◎	◎	●
left	2-3, TV	●	◎	○	●	●	●	●	●	●	●	●
letter-spacing	1-3			●	●	●	●	●	●	●	●	●
line-height	1-3, TV	●	●	●	●	●	●	●	●	●	●	●
list-style	1-3, TV, Mobile	●	●	●	●	●	●	●	●	●	●	●
list-style-image	1-3, TV, Mobile	●		●	●	●	●	●	●	●	●	●
list-style-position	1-3, TV, Mobile	◎	◎	◎	◎	◎	◎	◎	◎	◎	◎	●
list-style-type	1-3, TV, Mobile	◎	◎	◎	◎	◎	◎	◎	◎	◎	◎	◎
margin	1-3, TV, Mobile	⊙	●	●	●	●	●	●	●	●	●	●
margin	2-3						⊙	⊙				
marker-offset	2, 3											
marks	2, 3											
max-height	2-3						◎	◎	◎	◎	◎	◎
max-width	2-3						●	●	●	●	●	●
min-height	2-3						◎	◎	◎	◎	◎	●
min-width	2-3						●	●	●	●	●	●
opacity	3									●	●	
orphans	2-3											
outline	2-3, TV			◎			●	●		●[2]	◎	

Alphabetische Übersicht der CSS-Eigenschaften

Markt+Technik

CSS-Eigenschaft	CSS-Version	Palm	NN	Internet Explorer			Opera		Mozilla		Safari/Konq	iCab
		2.x	4.x	5.x (Mac)	6 (Win)	7 (Win)	7	8	0.x+	1.0+	2.0+/ 3.x+	3.0
outline-color	2-3, TV			◎			●	●		●[2]	◎	
outline-style	2-3, TV			◎			●	●		●[2]	◎	
outline-width	2-3, TV			◎			●	●		●[2]	◎	
overflow	2-3	◎		⊙	●	●	●	●	●	●	●	●
padding	1-3, TV, Mobile	⊙	●	○	●	●	●	●	●	●	●	●
page	2, 3											
page-break-after	2, 3			●	●	●	●	●		●[1]	●	
page-break-before	2, 3			●	●	●	●	●		●[1]	●	
page-break-inside	2, 3				●	●	●	●				
pause	2-3											
pause-after	2-3											
pause-before	2-3											
pitch	2-3											
pitch-range	2-3											
play-during	2-3											
position	2-3, TV	⊙		●	◎	⊙	●	●	●	●	●	●
quotes	2-3			⊙			◎	◎	◎	◎	⊙	◎
richness	2-3											
right	2-3, TV	●		○	●	●	●	●	●	●	●	●
size	2, 3			⊙			⊙	⊙				
speak	2-3											
speak-header	2, 3											
speak-numeral	2, 3											
speak-punctuation	2, 3											
speech-rate	2-3											
stress	2-3											
table-layout	2-3			●	●	●	●	●	●	●	●	●
text-align	2, 3, Mobile, TV		◎	◎	◎	◎	◎	◎	◎	◎	◎	◎
text-align	1-3, TV, Mobile	◎	◎	◎	●	●	●	●	●	●	●	●
text-align-last	3				◎	◎						
text-decoration	1-3, TV, Mobile	●	◎	◎	●	◎	●	●	●	●	●	●
text-indent	1-3, TV, Mobile	◎	◎	●	●	●	●	●	●	●	●	●
text-justify	3				●	●						
text-shadow	2, 3										◎	
text-transform	1-3, TV, Mobile		⊙	●	●	●	●	●	●	●	●	●
top	2-3, TV			●	●	●	●	●	●	●	●	●
vertical-align	2, 3, TV, Mobile	●	◎	●	●	●	●	●	●	●	●	●
visibility	2-3, TV, Mobile	◎	◎	◎	◎		◎	◎	●	●	◎	◎
voice-family	2-3											
volume	2-3											
white-space	1-3, TV, Mobile	⊙	◎	◎	◎	◎	◎	●	◎	◎	●	●
widows	2-3											
width	1-3, TV, Mobile	●	⊙	●	●	●	●	●	●	●	◎	●
word-spacing	1-3			●	●	●	●	●	●	●	●	●
z-index	2-3, TV	◎	●	●	●	●	●	●	●	●	●	●

1 ab Version 1.4+
2 ab Version 1.8+

Selektoren

	CSS-Version	Palm	NN	IE			Opera		Mozilla		Safari/Konq	iCab
Selektor		2.x	4.x	5.x (Mac)	6 (Win)	7 Win)	7	8	0.x+	1.0+	2.0+/3.x+	3.0
Element-Selektor	1-3, Mobile, TV	●	●	●	●	●	●	●				●
ID-Selektor	1-3, Mobile, TV	●	●	●	●	●	●	●				●
Klassen-Selektor	1-3, Mobile, TV	●	●	●	●	●	●	●				●
Nachfahren-Selektor	1-3, Mobile, TV	●	●	●	●	●	●	●				●
Indirekter Geschwister-Selektor	2-3, TV, Mobile											
Kind-Selektor	2-3, TV, Mobile	●		●			●	●	●	●	●	●
Geschwister-Selektor	2-3	●		●			●	●	●	●	●	●
Universal-Selektor	2-3, TV, Mobile	●	⊙	●	●	●	●	●	●	●	●	⊙
Attribut-Selektor `[Attribut]`	2-3						●	●	●	●	⊙	●
Attribut-Selektor `[Attribut=Wert]`	2-3						●	●	●	●	●	●
Attribut-Selektor `[Attribut~=Wert]`	2-3	●					●	●	●	●	●	●
Attribut-Selektor `[Attribut|=Wert]`	2-3						●	●	●	●	●	●
Attribut-Selektor `[Attribut^="Anfang"]`	3									●[1]		●
Attribut-Selektor `[Attribut$="Ende"]`	3									●[1]		●
Attribut-Selektor `[Attribut*="Ende"]`	3									●[1]		●
Attribut-Selektor/Pseudo-Klasse `:not[Attribut]`	3									●[1]		
Kombinierter Attribut-Selektor `*[attributselector] [attributselector]`	2-3						● 2	● 2	● 2	● 2	⊙ 2	● 2
Tabellen-Selektoren `table.row[zeile]` `table.column[spalte]`	3											

1 Unterstützung erst ab Mozilla 1.4+

2 Vorausgesetzt, es werden nur Attribut-Selektoren verwendet, die der Browser unterstützt

Legende

○ fehlerhaft bis unbrauchbar

⊙ teilweise fehlerhaft

⊙ teilweise Unterstützung

● weitgehend fehlerfreie und vollständige Unterstützung

Beilage aus:
CSS-Praxisbuch
Helma Spona
ISBN 3-8272-6892-3
€ 39,95 (D) / € 41,10 (A)