Sven Brencher
Medien Consulting & Services

HTML5 und CSS3

Sven Brencher
s.brencher@inlite.de
www.svenbrencher.de
01724242342
twitter.com/svenbrencher

HTML5 und CSS3 Seminar

1. Neue HTML5 Tags und Funktionen

1.1. Der neue Doctype und die veränderte Schreibweisen

1.1.1. Doctype

HTML5 soll wieder einfacher werden und das merkt man gleich am neuen Doctype. XML Namespaces und aufwendige Kennzeichnungen der HTML Versionen entfallen. Die Elemente <html>, <head> und <body> dürfen sogar ganz weggelassen werden. XML-Namespaces werden ebenfalls nicht mehr deklariert.

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset=utf8>
```

Vergleich zu XHTML:

```
<!DOCTYPE html PUBLIC
"-//W3C//DTD XHTML
1.0 Transitional//EN"
"http://www.w3.org/
TR/xhtml1/DTD/xhtml1-
transitional.dtd">
<html xmlns="http://www.
w3.org/1999/xhtml">
```

1.1.2. Verzicht auf XHTML Syntax

Bei Elementen, die nicht geschlossen werden braucht kein Schrägstrich mehr gesetzt zu werden. Manche Elemente, wie zum Beispiel <li> und <p> müssen gar nicht mehr geschlossen werden. Ebenso dürfen die Anführungszeichen bei Attributen weggelassen werden, wenn der Wert keine Leerzeichen beinhaltet.

```
XHTML: <img src="..." />  HTML5: <img src=...>
XHTML: <br />             HTML5: <br>
XHTML: <p>Hallo</p>       HTML5: <p>Hallo
XHTML: <p class="bold">   HTML5: <p class=bold>
```

Die technischen Spezifikationen des W3C zu HTML5 Working Draft für Web Autoren: http://www.w3.org/TR/html5/

Weitere gute Artikel und Dokumentationen findet man unter http://www.webplatform.org

Die Pläne des W3C für HTML5 bis 2014: http://dev.w3.org/html5/decision-policy/html5-2014-plan.html und HTML5.1 http://www.w3.org/html/wg/drafts/html/master/ (Seit Dezember 2012).

1.2. Verbesserte Semantik mit neuen Block Elementen

Die Struktur von HTML5 Webseiten lässt sich durch eine verbesserte Semantik deutlich besser kennzeichnen. So kann man folgenden neuen HTML Elementen bereits eine inhaltliche Bedeutung entnehmen:

```
<header><h1>Titel</h1></header>
<hgroup><h1>Titel</h1><h2>Untertitel</h2></hgroup>
<nav><ul><li>Home</li></ul></nav>
<figure>
  <img src="..."/>
  <figcaption>Bildunterschrift</figcaption>
</figure>
<footer>
  <time datetime="2011-10-28" pubdate>2011</time>
</footer>
<section>
  <article>Hier steht ein Artikel</article>
  <aside>Neben dem Artikel ist eine Sidebar</aside>
</section>
```

Diese Grafik zeigt schnell, welche Browser eine HTML5 oder CSS3 Funktion bereits unterstützen. Für aktuelle Browserversionen und unterstützte Funktionen im Detail helfen die folgende Webseiten:
http://caniuse.com
http://fmbip.com/litmus

1.3. Alternativen für ältere Browser

1.3.1. CSS Ergänzung für die neuen Block Elemente

Da ältere Browser HTML5 Tags nicht immer als Block-Elemente erkennen, sollte man dies mit folgendem Code im CSS Stylesheet ergänzen:

```
header, section, footer, aside, nav, article, figure,
figcaption, menu, details, hgroup {
  display: block;
}
```

Block Elemente werden im Browser untereinander dargestellt. Inline Elemente werden nebeneinander gerendert. Inline-block Elementen lässt sich trotzdem eine feste Breite zuweisen.

1.3.2. HTML5 Shiv und Lösungen für ältere Browser

Der Internet Explorer bis zur Version 8 unterstützt keine HTML5 Elemente und ist auf eine JavaScript Umwandlung in verständliche Elemente angewiesen. Dies erledigt folgender Eintrag im <head> Bereich der Seite und ist unbedingt zu empfehlen:

```
<!--[if lt IE 9]>
  <script src="http://html5shiv.googlecode.com/svn/
  trunk/html5.js"></script>
<![endif]-->
```

Statt aus dem Code Repository von Google sollte das HTML5Shiv besser als komprimierte Version vom eigenen Server verwendet werden. Die aktuelle Version kann unter https://github.com/aFarkas/html5shiv heruntergeladen werden.

Weitere HTML5 und CSS3 Features sowie einige JavaScript APIs stellen Frameworks wie Modernizr und CSS3PIE für ältere Browser zur Verfügung. Die aktuelle Version des Modernizr enthält auch schon das HTML5 Shiv. Modernizr wiederum ist Bestandteil des HTML5Boilerplates.

Modernizr
FRONT-END DEVELOPMENT DONE RIGHT

Das Modernizr Framework verwendet Javascript um HTML5, CSS3 und JavaScript APIs in älteren Browsern zu simulieren: http://www.modernizr.com

progressive internet explorer CSS3 PIE

CSS3PIE (progressive Internet Explorer) hilft dem IE 6-9 bei neueren CSS3 Eigenschaften auf die Sprünge: http://css3pie.com

Aktuelle Artikel zu HTML5, Frameworks uvm. findet man auch unter http://html5bookmarks.com/

1.4. Änderungen in der Header Struktur

1.4.1. Das <hgroup> Element

Überschriftenstrukturen (h1 bis h6) können durch eine <hgroup> zusammengefasst werden. Das <hgroup> Element darf nur die Überschriftenelemente <h1> bis <h6> enthalten. Der Sinn dahinter ist, dass alle weiteren Überschriften nach <h1> aus der Outlinestruktur verschwinden.

```
<hgroup>
  <h1>Titel</h1>
  <h2>Zwischentitel</h2>
</hgroup>
```

Anmerkung: im April 2013 hat das W3C überlegt, ob es das hgroup Element wieder aus der Spezifikation streicht. Also erstmal auf den Einsatz verzichten.

1.4.2. Das <header> Element

Das <header> Element kennzeichnet sowohl die Kopfzeile eines HTML Dokumentes wie auch Überschriften innerhalb von Artikeln. Innerhalb des <header> Elementes dürfen auch weitere Elemente wie Suchfelder, Inhaltsangaben, Listen, Navigationen, Absätze und Logos platziert werden:

```
<header>
  <hgroup>
    <h1>Meine Firma</h1>
    <h2>Unser Slogan</h2>
  </hgroup>
  <img src="logo.jpg">
  <p>Experten für HTML5</p>
  <nav>
    <ul>
      <li><a>HTML5</a></li>
      <li><a>CSS3</a></li>
    </ul>
  </nav>
</header>
```

Ein <header> Element kann auch innerhalb von <article> oder <section> verwendet werden:

```
<article>
  <header><h1>HTML5 ist großartig</h1></header>
  <p>...</p>
</article>
```

Innerhalb eines <header> fangen die Überschriftenstrukturen immer wieder mit <h1> an. Die Erstellung einer HTML5 Outline weicht insofern von älteren Versionen sehr stark ab.

1.4.3. Die Überschriften Struktur und der neue Outline Modus

Jedes <header> Element beginnt immer mit einer <h1> Überschrift. Ein <h1> bis <h6> innerhalb eines <articles>, einer <section>, eines <aside> und eines <nav> wird vom Browser automatisch als untergeordnete Überschrift erkannt, auch wenn beide mit dem <h1> Element ausgezeichnet sind. Insofern gibt es die Beschränkung auf sechs Überschriften <h1> bis <h6> nur noch innerhalb einer <hgroup>.

Dazu ein Beispiel:

```
<header><h1>Dinosaurier der Kreidezeit</h1></header>
<section>
  <h1>Warum sind alle Dinos tot?</h1>
  <article>
    <header><h1>Allgemeine Theorie</h1></header>
  </article>
</section>
```

Eine solche Struktur würde in einem Inhaltsverzeichnis folgendermaßen abgebildet werden:

1.) Dinosaurier der Kreidezeit (Seitentitel)
1.1.) Warum sind alle Dinos tot? (Abschnittstitel)
1.1.1.) Allgemeine Theorie (Artikeltitel)

Mit einer <section>, einem <article>, einem <nav> oder einem <aside> beginnt immer ein neues, untergeordnetes Element innerhalb einer Überschriftenstruktur.
Die Struktur eines HTML5 Dokumentes lässt sich mit einen HTML5 Outliner überprüfen: http://gsnedders.html5.org/outliner oder als Extension unter http://code.google.com/p/h5o

1.5. Abbildungen mit <figure> und <figcaption>

Gegenüber dem alt-Attribut des <img> Elementes kennzeichnet die <figcaption> eine sichtbare Bildunterschrift.

<figure> kennzeichnet visuelle Elemente oder Abbildungen, die zum Beispiel zu einem Artikel gehören. <figcaption> kennzeichnet die zugehörige Bildunterschrift.

```
<figure>
  <img src="apfel.jpg" alt="Ein Apfel">
  <figcaption>Vitamine für Webdesigner</figcaption>
</figure>
```

Während eine <figure> durchaus mehrere <img> oder <canvas> Elemente enthalten kann, sollte es nur eine <figcaption> geben. Diese kann allerdings vor oder nach den Bildern gesetzt werden. Die <figcaption> kann ihrerseits wieder Hyperlinks oder Verweise in Form von <a> Elementen enthalten.

Ein simples Grafikelement für das Design der Webseite sollte nicht mit dem <figure> Element versehen werden. Es ist eher für Abbildungen im Inhalt vorgesehen. Es kann sogar beliebige Elemente aufnehmen, wenn die Zusammenstellung eher einem Abbildungscharakter entspricht. Denkbar also auch für Codebeispiele, die im Inhalt auffallen sollen.

1.6. Die Block-Elemente für eine semantische Aufteilung der Seite

HTML 5 Elemente für die semantische Struktur der Webseitenbereiche.

1.6.1. Das Element

Das Element gibt es nach wie vor und im Prinzip spricht nichts dagegen ein komplettes HTML5 Dokument nur mit Elementen auszustatten. Generell gilt jedoch, dass man das Element erst dann einsetzt, wenn es kein semantisch besser passendes Element gibt.

Das Element sollte nach wie vor verwendet werden, wenn man lediglich ein Hilfsmittel für CSS Stile benötigt.

1.6.2. Das <section> Element

Ein <section> Element gruppiert einen Abschnitt bzw. eine inhaltliche Einheit auf einer Seite und fasst zum Beispiel mehrere Artikel zusammen. Das <section> Element sollte nicht verwendet werden, wenn <article>, <aside> oder <nav> dem Inhalt nach besser passen würden.

Generell sollte man einer <section> auch eine Überschrift geben können, die beschreibt um welche Art Inhalt es sich handelt. Mit jeder <section> beginnt eine neue Hierarchie-Ebene für Überschriften.

```
<section>
  <h1>Artikelübersicht</h1>
  <article><h1>Artikel 1</h1><p>Inhalt</p></article>
  <article><h1>Artikel 2</h1><p>Inhalt</p></article>
</section>
```

Outline für dieses Beispiel:
1. Artikelübersicht
    1. Artikel 1
    2. Artikel 2

1.6.3. Das <article> Element

Das <article> Element kennzeichnet einen inhaltlichen Beitrag, der auch vom Rest der Seite losgelöst bzw. unabhängig bestehen könnte. Neben dem Inhalt kann ein <article> Element auch einen eigenen <header> und <footer> haben. In diesen können zum Beispiel Überschriften, Autoren oder Veröffentlichungsdaten enthalten sein.

Tatsächlich könnten auch die einzelnen Kommentare zu einem Blogbeitrag für sich wieder einzelne <article> Elemente darstellen. Diese sollten dann aber wieder über eine eigene <section> innerhalb des übergeordneten <article> Elements zusammengefasst werden.

Der Inhalt eines <article> Elementes kann bei längeren Artikeln wieder in logische <section> Elemente aufgeteilt werden.

Innerhalb eines <article> Elements kann das <time> Element verwendet werden, um zum Beispiel ein Veröffentlichungsdatum zu kennzeichnen:

```
<article>
  <header>
    <h1>Die Affen</h1>
    <p>von Jim Panse</p>
    <p><time datetime="2012-01-23">
      Veröffentlicht am 23.01.2012
    </time></p>
  </header>
  <p>Wer hat die Kokusnus geklaut?</p>
</article>
```

1.6.4. Das <aside> Element

Das <aside> Element ist nicht die klassische Sidebar für eine Subnavigation! Vielmehr enthält es Inhalte, die sich auf einen Artikelinhalt oder die ganze Seite beziehen.

Das <aside> Element hat zwei verschiedene Bedeutungen:

- Innerhalb eines <article> Elements enthält es zusätzliche Informationen, die sich auf den Artikel beziehen. Zum Beispiel eine Stichwortliste oder Tagcloud, sowie ein Glossar oder eine Aufzählung. Grundsätzlich sollte der Artikel auch ohne die ergänzenden Informationen in dem <aside> Element einen Sinn ergeben.

- Als Container für sekundäre Inhalte findet das <aside> Element Anwendung, wenn es nicht innerhalb eines <article> Elements steht. So etwas kann auch eine Sidebar sein, die sich nicht direkt auf einen Artikel bezieht aber der Inhalt des <aside> Elements sollte sich dann schon auf den Inhalt der gesamten Seite beziehen. Dazu zählt zum Beispiel eine Blogroll, eine zusätzliche Navigation oder sogar eine Anzeige, wenn sie einen Bezug zur Seite hat.

1.6.5. Der <footer>

Der <footer> kann sowohl als Fußzeile der Seite angesehen werden, wie auch als Fußzeile eines Artikels. Das <footer> Element kann verschiedenste Kindelemente anzeigen.

Als Inhalte für eine Artikelfußzeile bietet sich z.B. eine Autorenzeile an. Als Dokumentfußzeile lassen sich Links zu weiterführenden Seiten, Copyright-Informationen oder Social Media Integrationen einbinden.

1.7. Neue Elemente für HTML5 Anwendungen

Für die Erstellung von HTML5 Anwendungen gibt im Gegensatz zu den Inhalts-strukturierenden Elementen zahlreiche neue Elemente die bestimmte Funktionen kennzeichnen. Dazu gehören Bezeichnungen für häufig verwendete Steuerelemente in Anwendungen.

Obwohl die Nachfrage nach richtigen HTML5 Anwendungen sehr groß ist, ist die Unterstützung in den aktuellen Browsern dafür noch sehr gering und Entwickler müssen sehr oft auf JavaScript Bibliotheken zurückgreifen.

1.7.1. Mehr Details zum aufklappen

Mit dem <details> Element lassen sich ohne JavaScript zusätzliche Details aufklappen, wenn man auf ein sichtbares <summary> Element klickt. <summary> muss das erste Kindelement des <details> Elements sein. Danach dürfen beliebige weitere folgen.

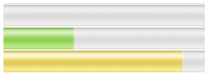
Aktuell wird dieses Element nur vom Chrome Browser und von Safari 6 unterstützt und daher steht es momentan auf der "Feature at Risk" Liste vom W3C für HTML5. Ähnliches Los erwartet auch die Anwendungs-Elemente <menu>, <command> und <output>

```
<details>
  <summary>Mehr zum Thema wird sichtbar...</summary>
  <p>...nachdem sie oben geklickt haben.<p>
  <p>Leider nur im Google Chrome Browser.</p>
</details>
```

1.7.2. Werte- und Fortschrittsbalken

Grafische Darstellung eines Wertes innerhalb eines festgelegten Bereichs. Zum Beispiel verfügbarer Speicherplatz oder ein Highscore. Progress wird ab IE 10 unterstützt; das Meter Element zumindest teilweise.

```
<meter min="0" max="100" low="40" high="90"
optimum="100" value="91">Dein Punktestand</meter>
```



Darstellung des Fortschritts einer Aufgabe.

```
<progress>lade Daten herunter...</progress>
<progress value="75" max="100">75% geladen</progress>
```

1.8. Änderungen bei der Bedeutung von verschiedenen Hervorhebungen zwischen HTML4 und HTML5

1.8.1. Markieren von Suchergebnissen mit <mark>

Hervorhebung einer Textstelle, die auf Interaktion mit dem User ausgelegt wird – zum Beispiel zur Hervorhebung eines Suchergebnisses.

```
<p>Hier ist das <mark>Ergebnis</mark> nach dem Sie
gesucht haben</p>
```

Hier ist das Ergebnis nach dem Sie gesucht haben

1.8.2. Betonungen und Hervorhebungen mit und

Hervorhebung wichtiger Textpassagen. Wird in der Regel fett gesetzt.

```
<p>Das ist sehr <strong>wichtig</strong>!</p>
```

Eine betonte Textpassage oder ein Akzent. Ein Nachdruck in der Rede. Wird in der Regel kursiv gesetzt.

```
<p>Dies ist eine <em>leichte</em> Betonung.</p>
```

Sowohl wie auch Elemente lassen sich verschachteln, um zum Beispiel sehr sehr wichtige Inhalte zu kennzeichnen.

```
<strong>Diese Information bitte <strong>genau</strong>
durchlesen!</strong>
```

Das <b> oder <i> Element sollte nur noch verwendet werden, wenn die anderen symantisch nicht passen würden. Das ist der Fall bei rein optischen Hervorhebungen ohne das der Inhalt eine wichtigere Bedeutung hat. Das könnte zum Beispiel bei Namen der Fall sein.

1.8.3. Weitere Kennzeichnungen mit <small>, <cite>, <adress>, <s> und <code>

Kennzeichnung für Kleingedrucktes; Fußnoten oder Disclaimer:

```
<small>Copyright &copy; 2012 Sven Brencher</small>
```

Titel eines Werks, eines Buches, eines Films oder eines Songs:

```
<p>2012 erscheint <cite>der kleine Hobbit</cite> im
Kino.</p>
```

Kontakt- oder Adressinformationen für das gesamte Dokument oder einen einzelnen Artikel :

```
<address>
  <a href="http://www.google.de">Google</a>
</address>
```

Wird <adress> im <body> verwendet, dann bezieht es sich auf Kontaktinformationen für das Dokument. Liegt es innerhalb eines <article> Elements, dann zeichnet es die Kontaktdaten des Autoren aus.

Um Inhalte zu kennzeichnen, die nicht mehr relevant sind, bietet sich das <s> Element an, um diese auch durchzustreichen.

```
<p>Statt <s>24,95 €</s> jetzt nur 19,95 €.</p>
```

Beim <code> Element kann im class-Attribut das Prefix language- die Art der Sprache definiert werden.

```
<code class="language-javascript">
  alert('Dies ist Javascript')
</code>
```

1.8.4. Inhaltliche Trennungen mit <hr>

Das <hr> Element zieht nicht nur eine Trennlinie, sondern bildet auch eine inhaltliche Trennung von zwei Absätzen. Es darf allerdings nicht zwei <section> oder <article> Elemente voneinander trennen.

```
<p>Der erste Standpunkt ist durchaus nachvollziehbar
<hr>
<p>Der andere Standpunkt hat ebenfalls gute Gründe
```

1.9. Neue Formularelemente, Eigenschaften und Typen

Zum Testen der verschiedenen Formularfunktion immer an das <form> Element denken! Opera ist der einzuge Browser, der alle HTML5 Form Funktionen unterstützt.

1.9.1. Neue Formularattribute: required, autocomplete, autofocus, placeholder und pattern

required

Kennzeichnen eines benötigten Feldes, dass der Browser ohne Javascript vor dem Versand auf Inhalt überprüft. Kann auf alle Feldtypen angewender werden. Required wird vom IE 10 und teilweise von Safari 6 verstanden.

```
<form>
  E-Mail:<input type="email" name="usrmail" required>
  <input type="submit" value="senden">
</form>
```

autocomplete

Autocomplete kann für die Feldtypen text, search, url, telephone, email, password, datepickers, range und color verwendet werden oder sogar direkt in das <form> Element geschrieben werden.

```
<form autocomplete="on">
  Vorname: <input type="text" name="fname"><br>
  Nachname: <input type="text" name="lname"><br>
  E-Mail: <input type="email" name="email"
     autocomplete="off"><br>
  <input value="senden" type="submit">
</form>
```

autofocus

Autofocus kann bei allen Formularfeldtypen verwendet werden. Es wird ab IE 10 unterstützt.

```
<input type="text" autofocus="autofocus">
```

pattern

Das Pattern Attribut validiert die Eingabe eines Textfeldes über einen regulären Ausdruck (GREP). Funktioniert mit den Feldtypen text, search, url, telephone, email und password. Pattern wird ab IE 10 unterstützt.

```
<form action="forms.html">
  <input type="text" name="Land"
     pattern="[A-Za-z]{2,3}">
  <input type="text" name="PLZ" pattern="[0-9]{5}">
  <input type="submit">
</form>
```

placeholder

Das Platzhalter Attribut definiert einen Text in einem text, search, url, telephone, email oder password Feld. Dabei wird der Text automatisch gelöscht, wenn das Feld den Focus erhält. Placeholder wird ab IE 10 unterstützt.

```
<input type="search" placeholder="Neuer Suchbegriff">
```

form

Das form-Attribut gibt an zu welchem Formular dieses Feld gehört.

```
<form id="user">
  Vorname:<input type="text" name="fname">
  <input type="submit">
</form>
Nachname:<input type="text" name="lname" form="user">
```

form overrides

Überschreibt einige der Eigenschaften, die im <form> Element definiert wurden und kann auf die Typen submit und image angewendet werden. Es wird ab IE 10 unterstützt.

- formaction
- formenctype
- formmethod
- formnovalidate
- formtarget

```
<form action="forms.html" method="get" id="user">
  <input type="submit" value="Senden">
  <input type="submit" formaction="formadmin.php"
     value="Als Administrator versenden">
</form>
```

Sonstige Attribute

- list: siehe Datalist auf Seite 14.
- height- und width Attribute für Formularfelder mit type="image" (alle Browser).
- min, max und step Attribute für Formularfelder mit type="number" (nur Firefox und Chrome).
- multiple="multiple" für die Mehrfachauswahl in den Feldtypen email und file (Alle Browser außer Internet Explorer).

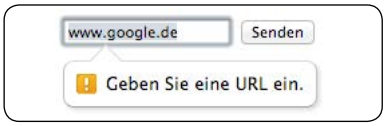
1.9.2. Neue Formularfeldtypen

Mit neuen <input> Typen können Browser Datumsfelder anzeigen oder nach bestimmten Eingaben validieren. Die Validierung findet nur bei Feldern mit name-Attribut statt, die innerhalb eines <form> Elementes liegen.

Beim URL Feld muss sogar das http:// Prefix eingegeben werden – ansonsten gibt es eine Fehlermeldung beim senden. Besonders in mobilen Browsern ist das spannend, um auch die Tastatur anzupassen oder zum Beispiel Telefonnummern-Felder mit dem Adressbuch abzugleichen.

Felder für E-Mail Adressen, URLs, Telefonnummern und Suchfunktion

```
<form action="forms.html">
  <input type="email" value="ich@meineseite.de">
  <input type="url">
  <input type="tel" placehowlder="1234 567890"
     pattern="^\(?\d{3}\)?[-\s]\d{3}[-\s]\d{4}.*?$">
  <input type="search" results="10"
     placeholder="Suchen...">
</form>
```



Mobile Browser stellen jeweils andere Tastenfunktionen für verschiedene Feldtypen zur Verfügung. Der Typ "tel" steht nur auf mobilen Browsern zur Verfügung. Ein Suchfeld wird von vielen Browsern mit abgerundeten Seiten angezeigt – grundsätzlich schreibt die Spezifikation vor, dass ein Suchfeld das Aussehen und Verhalten des Betriebssystems übernehmen soll.

Number und Range

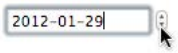
Bei diesen Feldern lassen sich nur Zahlen eingeben. Beim Number-Feld ergänzen die Browser eine Step-Funktion. Das Rangefeld wird oft als horizontaler Slider dargestellt.

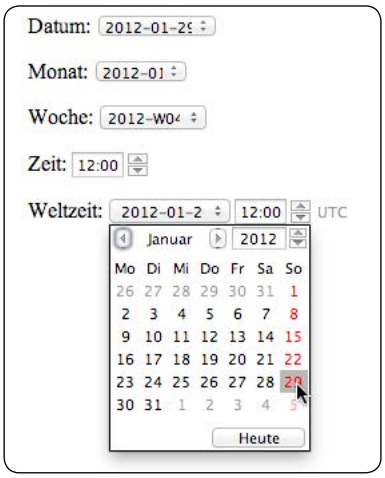
```
<input type="number" step="1" min="-5" max="10"
  value="3" width="60px">
<input type="range" min="0" max="50" value="20">
```

Datumsfelder

Das Datumsfeld kann ganz ohne Javascript in einigen Browsern ein Datumspicker darstellen. Die Art der Darstellung wird allerdings dem Browserhersteller überlassen. Ein richtiges Datumsauswahlfeld stellt nur Opera zur Verfügung.

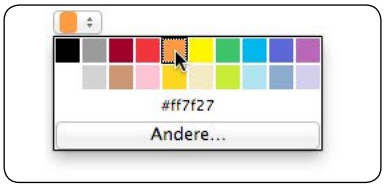
```
<input type="date" min="2010-01-01" max="2014-12-31"
  value="2012-01-29">
```





Weitere Datumstypen sind month, week, time, datetime (Weltzeit) und datetime-local (ohne Zeitzone). Time, datetime und datetime-local werden in Sekunden-Schritten angegeben. Wobei derzeit alle auf der Liste der "Features at Risk" beim W3C stehen.

```
<form action=forms.html>
  <p>Datum: <input type=date>
  <p>Monat: <input type=month>
  <p>Woche: <input type=week>
  <p>Zeit: <input type=time>
  <p>Weltzeit: <input type=datetime>
</form>
```



Farbpicker

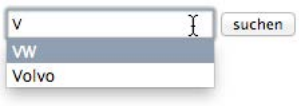
Zum auswählen von Farben. Die Vorgabe wäre das öffnen eine dem Betriebssystem ähnlichen Farbwählers. Dieser Input Type steht auf der Liste der "Features at Risk" beim W3C, da nur Chrome und Opera eine Unterstützung anbieten.

```
<form action="forms.html>
  <input type="color" value="#ff7f27">
</form>
```

1.9.3. Autovervollständigung mit der Datenliste

Eine Liste mit vordefinierten Werten, die als Vorschläge in ein Eingabetextfeld verwendet werden können. Wird ab IE 10 unterstützt.

```
<input list="autos" name="KFZ Suche">
<datalist id="autos">
  <option value="Audi">
  <option value="BMW">
  <option value="Porsche">
  <option value="Volvo">
  <option value="VW">
</datalist>
<input value="suchen" type="submit">
```



1.9.4. Neue Formular Elemente: keygen und output

Keygen erzeugt beim Senden eines Formulars einen privaten Schlüssel und sendet einen öffentlichen Schlüssen an den Server.

```
<keygen name="security">
```

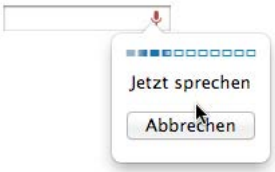
Das output Element wird verwendet, um Skriptergebnisse oder Berechnungen auszugeben. Derzeit steht das Output Element auf der Liste der "Features at Risk" beim W3C.

```
<form oninput="ergebnis.value=
  parseInt(border.value)*2+parseInt(breite.value)">
  <p>Breite <input type=number name=breite value=50>
  <p>Rahmen <input type=range name=border min=0 max=5
     onChange="rahmen.value=parseInt(border.value)">
     <output name=rahmen></output> px
  <p>Verdrängung <output name="ergebnis"></output> px
</form>
```

1.9.5. Das Spracheingabe Feld

Erlaubt die Eingabe über Sprache. Das ist zwar kein offizieller HTML5 Standard, aber trotzdem cool :-)

```
<input type="text" x-webkit-speech>
```



1.9.6. Formulare und JavaScript Überprüfung in HTML5

Mit JavaScript kann man die Validierung der Formularfelder überprüfen.

```
<form id="adresse" action="forms.html">
  <p><input id="plz" name="Postleitzahl" type="text"
     pattern="[0-9]{5}" required>
  <p><input type="button" value="Prüfen"
     onclick="pruefe()">
</form>
<script>
  var feld = document.getElementById('plz');
  var pruefe = function(){
     alert( feld.validity.valid );
     //alert( feld.validity.valueMissing );
  }
</script>
```

form.checkValidity() : Boolean
Führt eine Formularüberprüfung durch.
willValidate : Boolean
Gibt an, ob ein Feld überprüft wird.
validity.valid : Boolean
Gibt an, ob ein Feld gültig ist.
validity.valueMissing
Prüft ob ein required Feld Werte enthält.
validity.patternMismatch
Prüft ob ein Feld dem Pattern nach zu urteilen fehlerhafte Werte enthält.

Weitere Infos und Eigenschaften unter http://dev.w3.org/html5/spec/Overview.html#the-constraint-validation-api

1.10. Und darauf sollte man in Zukunft verzichten

Es gibt in HTML5 viele Elemente und Attribute, die nicht mehr unterstützt oder durch Alternativen ersetzt werden.

1.10.1. Nicht mehr unterstützte Elemente und alternative Vorschläge

- <applet> (stattdessen <embed> oder <object> verwenden)
- <acronym> (stattdessen <abbr> verwenden)
- <bgsound> (stattdessen <audio> verwenden)
- <dir> (stattdessen <ul> verwenden)
- <frame> (stattdessen <iframe> und CSS verwenden)
- <frameset> (stattdessen <iframe> und CSS verwenden)
- <isindex> (stattdessen <form> verwenden)
- <listing> (stattdessen <pre> oder <code> verwenden)
- <nextid> (GUIDs verwenden)
- <noembed> (stattdessen <object> oder <embed> verwenden)
- <plaintext> (text/plain MIME type verwenden)
- <rb> (stattdessen <ruby> verwenden)
- <strike> (stattdessen <del> oder <s> verwenden)
- <xmp> (stattdessen <code> mit "<" und "&" Escape Zeichen verwenden; also "<" oder "&")

1.10.2. Nicht mehr unterstützte Elemente ohne Alternativen

<basefont>, <big>, <blink>, , <font>, <marquee>, <multicol>, <nobr>, <spacer> und <tt>.

1.10.3. Nicht mehr unterstützte Attribute

Neben einigen Elementen verschwinden auch viele Attribute. Dazu ein paar Beispiele:

- charset, method, rev und urn innerhalb von <a> und <link>
- coords, shape und methods innerhalb von <a>
- name innerhalb von <a>, <embed>, <img> und <option>
- nohred innerhalb von <area>
- profile innerhalb von <head>
- version innerhalb von <httml>
- usemap innerhalb von <input>
- longdesc innerhalb von <iframe> und <img>
- lowsrc innerhalb von <img>
- target innerhalb von <link>
- scheme innerhalb von <meta>
- archive, classid, code, codebase, codetype, declare und standby innerhalb von <object>
- type und valuetype innerhalb von <param>
- language, event und for innerhalb von <script>

Die komplette Liste ist sehr lang und kann im W3C Working Draft eingesehen werden: http://www.w3.org/TR/html5-author/obsolete.html

1.11. Hyperlink Beziehungen, Microdata und ARIA

1.11.1. Link Beschreibungen mit dem rel-Attribut

Das rel- Attribut ist nicht neu, aber mit HTML5 gibt es viele neue Werte und Änderungen für die Beschreibungen von Hyperlinkbeziehungen.

```
<a rel="archives" href="http://myblog.com/
  archives">Archive</a>
```

Eine Übersicht aller bekannten Werte für das "rel" Attribut findet man unter http://microformats.org/wiki/existing-rel-values#POSH_usage

1.11.2. Microdata und Rich Snippets

Microformate sind ebenfalls nicht neu, aber sie gehören jetzt zum HTML5 Standard. Zum Beispiel vCard für Kontaktdaten:

```
<div class="vcard">
  <div class="adr">
     <span class="type">Arbeit</span>:
     <div class="street-address">Domsland 161</div>
     <span class="postal-code">24340</span>
     <span class="locality">Eckernförde</span>
     <div class="country-name">Deutschland</div>
  </div>
  <div class="tel">
     <span class="type">Arbeit</span> 043516666571
  </div>
  <div>Email:
     <span class="email">s.brencher@inlite.de</span>
  </div>
</div>
```

Rich Snippets lassen sich unter http://www.google.com/webmasters/tools/richsnippets testen.

Eine Liste der bekannten Microformat-Spezifikationen findet man unter http://microformats.org/wiki/Main_Page.

Den aktuellen Status zu HTML5 und Microformaten kann man unter http://microformats.org/wiki/html5 einsehen.

Bekannte und verbreitete Microformate umfassen:

- hCard
- hCalender
- hRecipe
- VoteLinks und hReview
- rel-license
- rel-tag
- XOXO

1.11.3. Accessible Rich Internet Applications (WAI-ARIA)

WAI-ARIA ermöglicht den barrierefreien Zugang für Webanwendungen.

```
<ul id="baum1" role="Baum" tabindex="0" aria-
labelledby="label_1">
  <li role="treeitem" tabindex="-1" aria-
expanded="true">Früchte</li>
  <li role="group">
     <ul>
        <li role="treeitem" tabindex="-1">Orangen</li>
        <li role="treeitem" tabindex="-1">Äpfel</li>
     </ul>
   </li>
</ul>
```

Mehr Infos zu WAI-ARIA unter http://www.w3.org/WAI/

1.12. Audio und Video

1.12.1. Audio

Mit dem neuen <audio> Element lassen sich sehr leicht Audiodateien auf einer Webseite einbinden. Folgende Formate werden von den Browsern dabei unterstützt:

					
mp3	✓		✓		✓
wav	✓	✓	✓	✓	
aac	✓		✓		✓
ogg/vorbis		✓	✓	✓	

Die Einbindung erfolgt über das <audio> Element und das controls Attribut sollte gesetzt werden, damit die Audiosteuerungselemente sichtbar sind.

```
<audio id="audio" src="sound.mp3" controls loop>
  <!-- Platz für alternative Inhalte oder Download -->
</audio>
```

Um mehrere Audio-Dateien für verschiedene Browser zu verwenden ist es möglich die source-Angabe als Element durchzuführen:

```
<audio controls>
  <source src="audio.ogg" type="audio/ogg">
  <source src="audio.mp3" type="audio/mpeg">
  Dein Browser spielt leider kein Audio ab!
</audio>
```

Via JavaScript kann man auf weitere Funktionen des <audio> Elementes zugreifen oder sogar ein neues Element erstellen:

```
<script>
  document.getElementById("audio").muted = false;
</script>
```

Erstellen und abspielen eines neuen Sounds via JavaScript:

```
<input type="button" onclick="abspielen()">
<script>
  var abspielen = function(){
     new Audio('audio.mp3').play();
  }
</script>
```

Weitere Attribute:
In Schleife abspielen: **loop**
Automatisch abspielen: **autoplay**

Audiodaten im Voraus laden:
preload="none" (deaktiviert)
preload="metadata" (nur Metadaten laden)
preload="auto" (der Browser entscheidet selbst)

Eine Zusammenstellung verschiedener Video und Audioplayer findet man unter http://www.praegnanz.de/html5video

1.12.2. Video

Mit dem <video> Element lassen sich Videoinhalte genauso einfach wie Videoinhalte hinzufügen.

					
h.264 (mp4)	✓	(✓?)	(✓?)		✓
WebM/VP8		✓	✓	✓	
ogg/theora		✓	✓	✓	

Google hat angekündigt in zukünftigen Versionen des Chrome Browsers den lizenzpflichtigen h.264 Codec nicht mehr zu unterstützen! Bisher aber davon noch nichts umgesetzt. Da auch Firefox in Zukunft wohl Unterstützung für H.264 bieten wird, ist es fraglich, ob Google diese Androhung umsetzen wird.

Die Einbindung eines einzelnen Videoclips geht genau wie beim <audio> Element. Zusätzlich kann das poster-Attribut verwendet werden, um ein Vorschaubild einzublenden, das bis zum Start des Videos sichtbar ist. Dieses sollte nach HTML5 Spezifikation ein Standbild aus dem Video sein!

```
<video src="movie.mp4" poster="image.jpg"
  autoplay controls>
  <!-- Alternative Inhalte z.B. Flash Fallback -->
</video>
```

Um mehrere Formate zu verwenden kann das <source> Element verwendet werden. Der Browser testet in der angegebenen Reihenfolge durch, bis er ein passendes Video findet.

```
<video id="player">
  <source src="movie.mp4" type="video/mp4" />
  <source src="movie.ogv" type="video/ogg" />
  <source src="movie.webm" type="video/webm" />
</video>
```

Um Videos in das Format WebM zu konvertieren findet man eine QuickTime Erweiterung unter http://code.google.com/p/webm/downloads/list
Für das OGG Format gibt es sogar einen online Dienst: http://video.online-convert.com/convert-to-ogg

Via Javascript lassen sich auch eigene Player-Controls definieren oder Funktionen des Videos aufrufen.

```
<input type=button value=play onclick="start()">
<input type=button value=mute onclick="mute()">
<script>
  var video = document.getElementById("player");
  var start = function(){
     if(video.paused){
        video.play();
     } else { video.pause(); }
  }
  var mute = function(){
     if(video.muted){
        video.muted = false;
     } else { video.muted = true; }
  }
</script>
```

Weitere praktische Eigenschaften des Video Objektes:
volume = 0.5 (zwischen 0 und 1)

Einfügen einer eigenen Progressbar mit dem <progress> Element (Dieses Skript bezieht sich auch auf das obere Beispiel).

```
<progress id=progbar min=0 value=0 style="width:50%">
</progress>
<script>
video.addEventListener('loadedmetadata', function(){
  var fortschritt=document.getElementById('progbar');
  fortschritt.setAttribute('max', video.duration);
  setInterval(function(){
     fortschritt.setAttribute('value',
       video.currentTime);
  }, 500);
}, false);
</script>
```

das loadedmetadata Ereignis wird ausgelöst, wenn alle Filminformationen zur Verfügung stehen und unter anderem auch die Länge bekannt ist.

currentTime kann auch das Video an einen bestimmten Zeitpunkt bewegen.

Webvideo kodieren

WebM und Ogg Clips lassen sich am besten über ein Firefox Plugin erstellen: http://firefogg.org/

Praktische HTML5 Videoplayer findet man unter:

- http://mediafront.org/osmplayer/
- http://sublimevideo.net/
- http://videojs.com/

1.13. Das Canvas Element

Das Canvas Element eignet sich zum zeichnen, teilweise sogar für 3D Elemente und kann via JavaScript angesteuert werden. Das zeichnen via JavaScript ist umständlich, aber es stehen zahlreiche Frameworks zur Verfügung um es zu erleichtern.

```
<canvas id="leinwand" width="500" height="300"
  style="border:1px solid black"></canvas>
<script>
var canvas = document.getElementById('leinwand');
if(canvas.getContext){
  var context = canvas.getContext('2d');
  context.fillStyle = 'red';
  context.rect(200,150,100,100);
  context.fill();
  context.closePath();

  context.strokeStyle = 'red';
  context.lineWidth = 6;
  context.lineCap = 'round';
  context.beginPath();
  context.moveTo(200,150);
  context.lineTo(250,75);
  context.lineTo(300,150);
  context.stroke();
  context.closePath();

  context.clearRect(220,180,30,75);
  context.clearRect(260,180,20,30);

  Math.toRad = function(x){
     return x * Math.PI / 180;
  }

context.beginPath();
  context.fillStyle = 'yellow';
  context.arc(350, 80, 30, Math.toRad(0),
     Math.toRad(360), true);
  context.strokeStyle = 'white';
  context.lineWidth = 0;
  context.stroke();
  context.fill();
  context.closePath();
}
</script>
```

Der 3D Context ist noch sehr experimentell. Die 3D Beschleunigung des Canvas ist über WebGL möglich. Demos unter: http://www.chromeexperiments.com/webgl

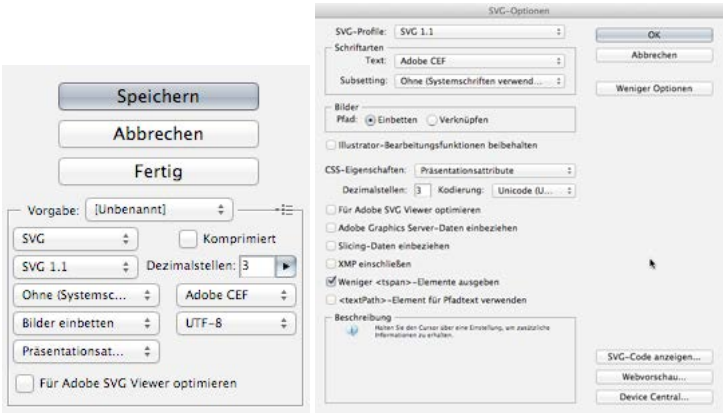
Ein praktisches Framework für Canvas Animationen stellt CreateJS dar: http://www.createjs.com

1.14. Inline SVG

SVG (Scalable Vector Graphics) sind Vektorgrafiken, die sich für die Interaktion via JavaScript eignen. Das <svg> Element lässt sich direkt innerhalb von anderen Elementen einsetzen.

```
<p>
  <svg>
    <circle id="myCircle" class="important"
      cx="50%" cy="50%" r="100"
      fill="url(#myGradient)"
      onmousedown="alert('hello');"/>
  </svg>
</p>
```

SVG von Hand zu entwickeln macht genauso wenig Spaß wie die Canvas Programmierung. Adobe Illustrator bietet gute SVG Werkzeuge:



Illustrator SVG Exportoptionen unter Datei > Für Web und Geräte speichern und Datei > Speichern > SVG. Letzteres bietet auch eine Code-Vorschau.

2. Neue CSS3 Funktionen und Module

Den aktuellen Stand aller CSS3 Module listet das W3C unter http://www.w3.org/TR/#tr_CSS auf.

CSS3 ist keine einzelne Spezifikation, sondern besteht aus vielen Modulen, die von unterschiedlichen Gruppen bearbeitet werden. Der Vorteil ist, dass einzelne Teile der gesamten Spezifikation bereits den Recommended Status erhalten können, während sich andere noch im Working Draft Zustand befinden. Die wichtigsten Module für Webdesigner sind folgende:

- Media Queries
  Zuweisung verschiedener CSS Stile nach Geräten und Eigenschaften
  http://www.w3.org/TR/css3-mediaqueries/
- Selectors Level 3
  Selektoren und Pseudo Klassen
  http://www.w3.org/TR/css3-selectors/
- CCS Color Module Level 3
  RGBA und HSLA Farben und Deckkraft
  http://www.w3.org/TR/css3-color/
- CSS Image Values and Replaced Content Module Level 3
  Hintergrundbilder und Verläufe in CSS3
  http://www.w3.org/TR/css3-images/
- CSS Fonts Module Level 3
  Typografie, Fontfamilien und @font-face
  http://www.w3.org/TR/css3-fonts/
- CSS Text Level 3
  Textschatten, Auszeichnungen und Ausrichtung
  http://www.w3.org/TR/css3-text/
- CSS Basic User Interface Module Level 3 - CSS3 UI
  Box-sizing, Content und Text-Overflow
  http://www.w3.org/TR/css3-ui/
- CSS Backgrounds & Borders Module Level 3
  Schatten, Hintergründe, abgerundete Ecken und Border Images
  http://www.w3.org/TR/css3-background/
- CSS Multi-column Layout Module
  Mehrspaltige Layouts erstellen
  http://www.w3.org/TR/css3-multicol/
- CSS Flexible Box Layout Module
  Neue Display Eigenschaften, Flexbox Model
  http://www.w3.org/TR/css3-flexbox/
- CSS Template Layout Module
  Verwendung eines Rasters und Slots für das Layout
  http://www.w3.org/TR/css3-layout/
- CSS 2D Transforms
  Transformationswerte und Funktionen
  http://www.w3.org/TR/css3-2d-transforms/
- CSS 3D Transform Module Level 3
  3D Transformationen und Verzerrungen
  http://www.w3.org/TR/css3-3d-transforms/
- CSS Transitions Module Level 3
  Übergänge zwischen CSS Eigenschaften
  http://www.w3.org/TR/css3-transitions
- CSS Animations Module Level 3
  Animationen und Keyframes
  http://www.w3.org/TR/css3-animations

2.1. Browser Spezifische Informationen

2.1.1. Vendor Prefixes

Vendor Prefixes müssen vor viele CSS Eigenschaften gesetzt werden, da nicht alle Browser die CSS3 Befehle gleich umsetzen. Bei einer entsprechenden CSS Eigenschaft müssen die Werte für jeden Browser einzeln angegeben werden. Als letztes sollte die korrekte Schreibweise der Eigenschaft ohne Prefix definiert werden.

- Mozilla Firefox: -moz
- Safari und Chrome: -webkit
- Opera: -o
- Internet Explorer 9: -ms

```
#div {
  -moz-column-count: 3;
  -webkit-column-count: 3;
  column-count: 3;
}
```

Die Prefixes sind nicht mehr bei allen Eigenschaften notwendig. Gute Übersicht unter http://peter.sh/experiments/vendor-prefixed-css-property-overview

2.1.2. Browser Support

Die meisten neuen CSS Eigenschaften werden von modernen Browsern unterstützt.

Um Basiseigenschaften für die Internet Explorer 6-8 verwenden zu können – dazu zählen border-radius, box-shadow, border-image, multiple-background, linear-gradient für backgrounds ist CSS3PIE (http://css3pie.com) eine gute Lösung.

```
#div{
  border-radius: 1em;
  behaviour: url(PIE.htc);
}
```

Gute Übersichten zur aktuellen CSS3 Unterstützung im eigenen Browser findet man unter http://www.findmebyip.com

Der gleiche Anbieter bietet unter http://fmbip.com/litmus eine gute Übersicht der CSS Fähigkeiten der einzelnen Browser.

Sehr aktuelle Informationen dazu gibt es auch unter http://caniuse.com

2.2. Media Queries Modul

Media Queries weisen Stylesheets oder CSS Eigenschaften nach Abhängigkeit von verschiedenen Geräten oder Browser-Eigenschaften zu. Damit lässt sich ein unterschiedliches Aussehen für Smartphones, Tablets und Desktop Browser generieren.

Einbindung einer CSS Datei für ein bestimmtes Ausgabemedium unter unter HTML4:

```
<link rel="stylesheet" type="text/css" media="screen"
href="sans-serif.css">
```

Oder die Abfrage im CSS Stylesheet selbst:

```
@media screen {
  * { font-family: sans-serif }
}
```

Die für HTML4 typischen Media Typen print, screen und handheld sind in HTML5 deutlich erweitert worden. Dementsprechend komplexer wird auch die Abfrage:

```
<link rel="stylesheet" type="text/css" media="all
and (min-width:500px) and (max-width:1024px)"
href="example.css">
```

oder wieder direkt im Stylesheet:

```
@media all and (min-width:300px)
and (max-width:1024px) {
  * { font-family: sans-serif }
}
```

Die wichtigsten Media Queries:

```
color: 0 | 1 (für Farb- oder SW-Bildschirm)
orientation: portrait oder landscape
min-width und max-width in px oder em
min-height und max-height in px oder em
min-device-width und max-device-width
min-device-height und max-device-height
aspect-ratio oder device-aspect-ration: 16/9
```

Die komplette Liste der Media Queries unter http://www.w3.org/TR/css3-mediaqueries

2.3. Neue Level 3 Selektoren

2.3.1. Neue Pseudo Selektoren

Eine gute Übersicht über neue Selektoren und Pseudo Klassen: http://reference.sitepoint.com/css/selectorref

:nth-child(N) und :nth-last-child(N)

Mit dem Pseudo Selektor :nth-child(n) können Elemente nach bestimmten Positionen oder nach gerader (even) bzw. ungerader (odd) Anzahl ausgewählt werden.

```
.row:nth-child(3) { ... }
.row:nth-child(even) { ... }
.row:nth-child(odd) { ... }
```

Neben einer einfachen Zahl kann n auch eine sogenannte Number Expression in der Form an+b verwendet werden. Dabei steht b für den Beginn der Auswahl und a gibt in welchen Schritten nach b Elemente ausgewählt werden sollen. Beispiele:

- 3n+1 würde das erste Element auswählen und danach jedes weitere dritte Element.
- 4n+6 würde das sechste Element auswählen und danach jedes weitere vierte Element.
- 3n bzw. 3n+3 oder 3n+0 sind gleich und würden jedes dritte Element auswählen.
- 1n+3 kann auch vereinfacht als n+3 geschrieben werden und wählt ab dem dritten Element jedes weitere aus.
- -n+3 würde alle Elemente bis zum dritten Element auswählen.
- 5n-2 würde ab dem -2 Element nach oben in 5er Schritten auswählen. Natürlich gibt es kein -2 Element, aber mit der :nth-last-child(n) Methode ergibt es dann wieder Sinn.

Mit der Pseudo Klasse :nth-last-child(n) wird ausgehend vom letzten Element nach vorne ausgewählt. :nth-last-child(1) wählt also das letzte Element aus. Auch hier können Number Expressions verwendet werden.

Um die letzten vier Listeneinträge auszuwählen schreibt man:

```
li:nth-last-child(-n+4) { ... }
```

-n dreht in diesem Fall die Reihenfolge von hinten nach vorne wieder um. D.h. ab dem viert-letzten Element wird wieder bis zum Ende ausgewählt.

:nth-of-type(N) und :nth-of-last-type(N)

Möchte man eine Anzahl oder Reihe bestimmter Elemente auswählen, dann eignen sich dazu diese Methoden. Der vorangestellte Selektor wird dabei gesucht. In Verbindung mit dem Child Selektor lassen sich so Kindelemente eines bestimmten Typs auswählen. Andere Elemente werden dabei übersprungen.

Mit dem nächsten Beispiel wird jeder dritte Absatz beginnend mit dem zweiten von oben ausgewählt:

```
p:nth-of-type(3n+2) { ... }
```

Um die Auswahl auf ein übergeordnetes Element zu beschränken bietet sicher der Child Selektor an:

```
#gallery>img:nth-of-type(2n) { ... }
```

Hiermit wird jedes zweite Bild innerhalb der ID gallery angesprochen.

Natürlich gibt es auch die Möglichkeit die Auswahlreihenfolge wieder von hinten nach vorne umzukehren:

```
article>p:nth-last-of-type(1) { ... }
```

Diese Auswahl bezieht sich auf den letzten Absatz innerhalb eines <article> Elementes, selbst wenn danach noch ein anderes Element folgt.

Zur Errinnerung an bekannte CSS2.1 Selektoren:

```
.content>p { ... }  /* Der Child Selector, der
  darauffolgende Kind-Elemente auswählt */
h2+p { ... } /* Der Adjacent Sibling Selector, der
  ein direkt auf das h2 folgendes, benachbartes
  p-Element auswählt */
```

Weitere praktische Selektoren

Folgende Selektoren können die Auswahl vereinfachen:

```
p:first-child { ... } /* Der erste Absatz */
p:last-child { ... } /* Der letzte Absatz */
article>p:first-of-type { ... } /* Erster Absatz
  innerhalb eines Artikels */
article>p:last-of-type { ... } /* Letzter Absatz
  innerhalb eines Artikels */
p:only-child { ...} /* Wählt den Absatz nur aus, wenn
  er das einzige Kindelement ist */
p:only-of-type { ... } /* Wählt den Absatz aus, wenn
  es der einzige Absatz eines Elements ist */
p:empty { ... } /* Wählt einen leeren Absatz aus */
input:not(type="submit") { ... } /* Wählt
  Eingabefelder aus, deren Typ nicht submit ist */
:not(.active) { ... } /* Wählt Elemente aus die nicht
  die active-Klasse zugewiesen haben */
```

Selektoren für Formularfelder

Der Zustand von Formularfeldern lässt sich ebenfalls mit entsprechenden CSS Stilen versehen.

```
input:disabled { ... } /* Ein deaktiviertes Feld */
input:enabled { ... } /* Ein aktiviertes Feld */
input:checked {... } /* Ein selektierter Radio Button
  oder eine Checkbox */
```

Auswahl nach URL

Mit dem :target Selektor lässt sich ein Element der Seite auswählen dessen ID einem Fragment der URL entspricht.

```
*:target { ... }
```

Wenn die URL http://www.svenbrencher.de/index.html#about entspricht, dann würde das Element mit dem Attribut id="about" ausgewählt werden.

2.3.2. General Sibling Selektor – Nachbarn auswählen

Mit dem Sibling Selektor lassen sich Nachbarelemente auswählen.

```
h2~p {  ... } /* Wählt alle p Elemente aus, die
  darauffolgende Nachbarn eines h2 Elementes sind */
```

Zur Erinnerung an CSS2.1:

```
section>p {} /* alle p-Kindelemente einer section */
h1+p {} /* direkt nachfolgendes Element */
```

2.3.3. Attribut Selektoren

Attributselektoren wählen Elemente aus, die bestimmte Attribute besitzen. Folgende Schreibweise wird dafür verwendet:

```
[ATTRIBUTNAME { = | ^= | $= | *= } ATTRIBUTWERT]
```

Folgende Bedeutungen haben die Vergleichsoperatoren:

- = : Der Attributwert muss identisch sein.
- ^= : Der Attributwert beginnt mit den verwendeten Zeichen.
- $= : Der Attributwert endet mit den verwendeten Zeichen.
- *= : Der Attributwert enthält die verwendeten Zeichen.

Beispiele:

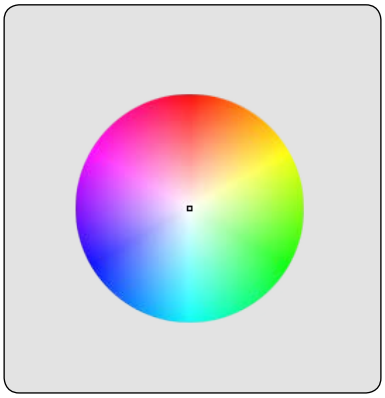
```
input[type="text"] { ... } /* der Typ ist text */
a[href^="http:"] { ... } /* URL beginnt mit http: */
img[src$=".png"] { ... } /* Dateiendung ist .png */
div[id*="content"] { ... } /* id enthält content */
```

2.4. Farbe, Deckkraft und Verläufe (Farb- und Image Module)

2.4.1. Die Farbmodelle HSL, HSLA, RGB und RGBA

HSL Farbmodell

Neue Farbdefinitionen erleichtern den Umgang mit verschiedenen Farbmodellen und ermöglichen sogar Transparenzen.



Das HSL Farbmodell steht für Hue (Farbton), Saturation (Sättigung) und Lightness (Helligkeit). Der Farbton wird in Grad auf einem Farbkreis angegeben, wobei 0° und 360° für rot stehen. Die Werte für Sättigung und Helligkeit werden zwischen 0% und 100% angegeben. Beim HSLA Farbmodell steht der letzte Wert für Alpha (Transparenz) und wird zwischen 0.0 (transparent) und 1.0 (deckend) angegeben.

```
div {
  color: hsl(40,30%,80%);
  background-color: hsla(210, 100%, 40%, 0.8);
}
```

Helligkeit 0% ergibt immer Schwarz und 100% immer Weiß, egal welche Werte für Farbton und Sättigung angegeben sind. Der Sättigungswert 0% ergibt in jedem Fall einen Graustufenwert.

RGB Farbmodell

Neben dem HSL Farbmodell gibt es auch das RGB Modell. Hier werden die Werte für Rot, Grün und Blau zwischen 0 und 255 angegen. Hierbei steht 0 für einen Schwarzwert im jeweiligen Kanal und 255 für einen Weißwert. Optional kann bei Verwendung von RGBA der Alphawert wieder zwischen 0 und 1 angegeben werden.

```
div {
  border: 6px solid rgba(204,30,30,0.2);
}
```

2.4.2. Deckkraft eines Elementes über die Opacity einstellen

Steuert die Deckkraft eines gesamten Elementes. Im Gegensatz zu RGBA und HSLA wird hier sowohl der Inhalt, wie auch das Element selbst transparent.

```
#gallery-thumbs img {
  opacity: 0.6;
}
#gallery-thumbs img:hover {
  opacity: 1;
}
```

2.4.3. Verläufe

Verläufe mit Hilfe von CSS3 statt mit Bildhintergründen sind zwar mittlerweile in allen modernen Browser unterstützt, allerdings müssen beim Safari Browser noch die Vendor-Prefixes vorangestellt werden.

Zur Erinnerung noch mal die Prefixes, die für alle Verläufe notwendig sind:
-moz für Firefox (Mozilla)
-webkit für Safari und Chrome
-o für Opera

Lineare Verläufe

Lineare Verläufe über die Funktion linear-gradient() lassen sich der background Eigenschaft zuweisen. Prinzipell reicht es, wenn man der Funktion zwei Farben übergibt:

```
div {
  background: -webkit-linear-gradient(red, blue);
  background: linear-gradient(red, blue);
}
```

Weitere Parameter können Anfang und Richtung des Verlaufs definieren. Für den Anfang stehen die Werte top, bottom, left und right zur Verfügung. Die Richtung wird in Grad mit dem Suffix deg (Degree) angegen.

```
div {
  background: -webkit-linear-gradient(left, #512, #f70);
}

div {
  background: -webkit-linear-gradient(45deg, #fff, #000);
}
```

Die Richtungsangaben in Grad:
0° links --> rechts (left)
45° links unten --> rechts oben
90° unten --> oben (bottom)
135° rechts unten --> links oben
180° rechts --> links (right)
225° rechts oben --> links unten
270° oben --> unten (top)
315° links oben --> rechts unten

Der Standardwert wenn nichts angegeben wird ist 270° bzw. top. Alternativ können auch negative Werte angegeben werden.

Statt die Syntax zu tippen, bieten sich der CSS Gradient Generator an: http://www.colorzilla.com/gradient-editor/

Prinzipiell können beliebig viele Farbwerte angegeben werden:

```
div {
  background: -webkit-linear-gradient(#512, #f70, #512);
}
```

Jede Farbe kann mit einem Stopwert in Prozent kombiniert werden. Sie wird dann bis zu dem angegeben Prozentsatz über die Fläche gefüllt.

```
div {
  background: -webkit-linear-gradient(top, #512 ,
    #712 15%, #912 25%, #712 35%, #712 75%, #512);
}
```

Radiale Farbverläufe

Radiale Verläufe werden im Prinzip wie lineare Verläufe angelegt. Haben aber noch deutlich mehr Optionen.

```
div {
  background: radial-gradient(red, blue, yellow);
}
```

Die Farben lassen sich auch über Prozentwerte verschieben; kombiniert mit einem Hintergrundbild und einem RGBA Farbwert lässt sich so z.B. ein Vignetierungseffekt abbilden:

```
div {
  width:400px; height: 240px;
  background: radial-gradient(rgba(0,0,0,0) 80%,
black), url(dream.jpg);
}
```

Als Formen lassen sich circle oder ellipse einstellen.

```
div {
  background: radial-gradient(left, ellipse,
     red, black);
}
```

Zusätzlich können vor den Farben die Positionen center, top, bottom, left und right angeben werden.

```
div {
  background: radial-gradient(bottom,
     yellow, white);
}
```

Das Zentrum des Verlaufs lässt sich auch in x- und y-Koordinaten definieren:

```
div {
  width:400px; height:240px;
  background: radial-gradient(150px 100px,
     ellipse, yellow 5%, rgba(0,0,0,0) 15%),
     radial-gradient(250px 100px,
     ellipse, yellow 5%, rgba(0,0,0,0) 15%), #000;
}
```

Und zuletzt lassen sich zu circle oder ellipse noch die Schlüsselwörter closest-side, closest-corner, farthest-side und farthest-corner angeben, wenn die Position absolut definiert ist. Diese bestimmen bis wohin der Verlauf maximal berechnet wird. Wer möchte kann diese Werte auch als Breite und Höhe in Pixel angeben.

```
background: radial-gradient(100px 100px,
  50px 120px, yellow, red);
background: radial-gradient(left,
  ellipse closest-corner, red, green);
```

2.4.4. Verläufe wiederholen – CSS Pattern

Eine Sammlung richtig schöner CSS3 Patterns findet man unter http://lea.verou.me/css3patterns/

Lineare und radiale Verläufe können wiederholt werden. Dafür gibt es die Funktionen repeating-linear-gradient() und repeating-radial-gradient(). Natürlich noch mit Browser-Prefixes.

Wichtig dabei ist auch, dass die einzelnen Farben Stopwerte erhalten, denn sonst dehnt sich ein einzelnes Verlaufsmuster auf die ganze Fläche aus.

```
div {
  background: repeating-linear-gradient(
     #ccc 10%, #eee 20%, #ccc 30%);
}
```

Legt man die Stopwerte übereinander kann man sogar Streifenmuster erstellen. Da hat zumindest der Browser ein bisschen was zum berechnen:

```
div {
  background: repeating-linear-gradient(left,
     rgba(100,0,0,.2) 2%, rgba(100,0,0,.2) 4%,
     rgba(0,0,0,0) 4%, rgba(0,0,0,0) 6%),
     -webkit-repeating-linear-gradient(top,
     #ccc 5%, #ccc 10%, #cc1 10%, #cc1 15%);
}
```

2.5. Typografie und Webfonts (Font und Text Module)

Das W3C listet im CSS Text Level 3 Working Draft alle Empfehlungen für Text auf: http://www.w3.org/TR/css3-text/

2.5.1. Eigene Webfonts mit @font-face

Über die @font-face Methode lassen sich individuelle Schriften auf der Webseite einbinden. Bei unterschiedlichen Schriftschnitt, sollten die Eigenschaften font-weight oder font-style definiert werden, damit jeweils richtige kursive oder fette Schnitte verwendet werden.

```
<style>
  @font-face {
     font-family: "Tekton Pro";
     src: url(fonts/TektonPro-Obl.otf)
        format("opentype")    /* Optional */
        local("Tekton Pro")   /* Optional */
        font-style:italic;    /* Optional */
  }
  header {
     font-family: "Tekton Pro";
     font-style:italic;
  }
</style>
```

Die optionale Eigenschaft local() greift auf eine lokal installierte Schrift zu, wenn diese vorhanden ist.

Fontsquirrel bietet einen guten Service für die Konvertierung von Schriften an: http://www.fontsquirrel.com/fontface/generator

Leider unterstützen nur moderne Browser die Schriftformate OpenType und TrueType. Für ältere Browser müssen verschiedene Schriftdateien angegeben werden. Es gibt eine sogenannte Bulletproof @font-face Syntax. Der Nachteil dieser Syntax ist, dass es keine Garantie gibt, das zukünftige Browsergenerationen noch damit umgehen können und leider braucht man auch mehrere verschiedene Formate der Schrift, darunter SVG, EOT und WOFF.

Es bietet sich daher an auf die Services von Webfont Anbietern, wie zum Beispiel Google Webfonts, zurückzugreifen: http://www.google.com/webfonts.

```
<head>
  <link href="http://fonts.googleapis.com/
     css?family=Contrail+One" rel="stylesheet"
     type="text/css">
  <style>
     body {
        font-family: 'Contrail One', cursive;
     }
  </style>
</head>
```

Webfontservices einbinden

Google bietet dabei sogar einen neuen Service: Dabei muss nicht die gesamte Font-Datei geladen werden, sondern nur die benötigten Zeichen. Dazu muss nur die URL um die text Eigenschaft verändert werden.
Die benötigten Zeichen unbedingt auch in Groß- und Kleinschreibung angeben:

```
http://fonts.googleapis.com/
  css?family=Contrail+One&text=Halo
```

Weitere Services stellen u.a. folgende Anbieter zur Verfügung:

- Fontshop: http://www.fontshop.com/webfonts
- Fontsquirrel: http://www.fontsquirrel.com
- Adobe Typekit: https://typekit.com

2.5.2. Text Schatten

Fügt einem Text einen Schatten hinzu. Browser-Prefixes werden nicht benötigt. Die Werte für die text-shadow Eigenschaft werden in folgender Reihenfolge angegeben: x-Versatz, y-Versatz, Weichzeichnerstärke und Farbe. Text-shadow wird ab IE 10 unterstützt.

```
.shadow {
  text-shadow: 2px 2px 2px #000;
}
```

Es ist auch möglich mehrere Schatten nacheinander zu definieren. So lässt sich zum Beispiel eine Kontur erstellen. Dabei werden die einzelnen Schatten durch Komma getrennt nacheinander angegeben.

```
.kontur {
  color: #FFF;
  text-shadow: 0 -1px #999, 1px 0 #999,
     0 1px #999, -1px 0 #999;
}
```

Oder ein eingelassener Text.

```
header {
  font-family:Verdana, Geneva, sans-serif;
  font-size:60px;
  font-weight:bold;
  background-color:#555;
  color:#666;
  text-shadow: 0px 2px 3px #888, 0px -2px 3px #222;
}
```

Die Alternative für den Internet Explorer 9 und ältere:

```
.ie-shadow { filter: glow(strength=5 color=black)
    blur(strenght=2 direction=45);
}
```

2.6. Basic UI Interface Module 3

2.6.1. Alternatives Box Modell

Mit der box-sizing Eigenschaft gibt es eine alternative Möglichkeit die Verdrängung eines Elements bestehend aus Breite/Höhe, Border, Margin und Padding zu berechnen. Zwar unterstützen alle Browser aktuell diese Funktion aber das W3C hat den box-sizing Wert padding-box auf die Liste der "Features at risk" gesetzt. Das heißt es könnte aus der Spezifikation wieder verschwinden. Aktuell benötigt der Firefox Browser noch Vendor-Prefixes.

Die box-sizing Eigenschaft kennt vier Werte:

- padding-box
- content-box (der Standardwert)
- border-box
- inherit

Der Wert content-box entspricht der Spezifikation einer Box aus CSS2.1. Border, Margin und Padding werden außerhalb der Breite und Höhe Werte gezeichnet und erhöhen die Verdrängung einer Box.

Mit padding-box wird der Innenabstand einer Box von der Höhe und Breite der Box abgezogen, sodass nur noch Margin und Border auf die Verdrängung einer Box aufgerechnet werden.

Über border-box wird der Rahmen (border) und der Innenabstand (padding) einer Box nach innen gezeichnet und nicht mehr der Höhe und Breite hinzugerechnet. Das ist sehr praktisch, wenn man prozentuale Angaben für Breite und Höhe gemacht hat, aber feste Werte für Innenabstand und Rahmen wählt.

```
section {
  width:400px;
}
article {
  float: left;
  width: 50%;
  padding: 10px;
  border: 5px solid #DDD;
  box-sizing: border-box;
}

<section>
  <article>Erste Artikelspalte</article>
  <article>Zweite Artikelspalte</article>
</section>
```

2.6.2. Konturen statt Rahmen

Im Gegensatz zur Border Eigenschaft nehmen Konturen keinen Platz bei der Verdrängung anderer Elemente ein. Sie eignen sich gut für Schaltflächen und Formularfelder. Die Outline Eigenschaft unterstützt außerdem einen Offset-Wert um die Kontur vom Element zu entfernen.

```
input {
  border: 1px solid #999;
  outline: 4px solid red;
  outline-offset: 2px;
}
```

2.6.3. Text Overflow

Die Eigenschaft text-overflow erzeugt drei Punkte mit denen dargestellt werden kann, dass eine Box noch mehr Text enthält, als eigentlich hineinpasst. Damit sich die Box nicht selbst erweitert werden die Eigenschaften overflow auf hidden und white-space auf nowrap gestellt.

Es gibt vier Werte für die text-overflow Eigenschaft: ellipsis, clip, string und ellipsis-word. Die letzte wird nur vom Internet Explorer unterstützt.

- ellipsis: am Ende des Textes werden die Punkte dargestellt. Firefox schneidet wie in der Spezifikation einfach das Wort ab.
- ellipsis-word: die Punkte sollen so gesetzt werden, dass sie direkt nach einem Wort kommen und das Wort nicht mehr angeschnitten wird. Bis auf den Internet Explorer wird das leider nicht von den Browsern unterstützt. Safari, Chrome und Opera zeigen dieses Verhalten aber auch mit dem einfachen ellipsis Wert.
- clip: das letzte Wort bzw. der Buchstabe wird einfach angeschnitten.

```
div {
  white-space:nowrap; overflow:hidden;
  text-overflow:ellipsis;
  width: 200px; padding: 2px;
  border:#333333 solid 1px;
}
div:hover {
  white-space:inherit;
  overflow:visible;
}
```

2.6.4. Resize – Objekte skalieren

Die Resize Eigenschaft kontrolliert, ob der Anwender ein Objekt skalieren darf. Als Werte stehen both, horizontal und vertical zur Verfügung.

```
div {
  resize: both;
}
```

2.7. Hintergründe, abgerundete Ecken und Bildecken

2.7.1. Box-Schatten

Im Gegensatz zum Textschatten lässt sich der Boxschatten auf Block-Elemente anwenden. Der Boxschatten wird mit den Werten Farbe, x-Abstand, y-Abstand und Weichzeichnerstärke angegeben. Für ältere Safari und Android Browser sollte das -webkit-Prefix verwendet werden.

```
div {
  -webkit-box-shadow: #000 5px 5px 10px;
   box-shadow: #000 5px 5px 10px;
}
```

Der Schatten kann über eine weitere Größenangabe am Ende zusätzlich skaliert werden. Mit negativen Werten lässt er sich auch verkleinern.

```
div {
  box-shadow: #000 0px 20px 20px -26px;
}
```

Mit der Eigenschaft inset kann ein Schatten nach innen versetzt werden.

```
div {
  padding:20px;
  box-shadow: #333 5px 5px 15px -5px inset;
}
```

2.7.2. Abgerundete Ecken

Elementen lässt sich mit der Eigenschaft border-radius eine abgerundete Ecke hinzufügen. Für ältere Safari und Android Browser sollte das Prefix -webkit verwendet werden.

```
div {
  -webkit-border-radius: 20px;
  border-radius: 20px;
}
```

Die Ecken lassen sich auch einzeln in der Reihfolge oben-links, oben-rechts, unten-rechts, unten-links definieren. Werden nur zwei Werte angegeben, dann gelten diese für oben-links und unten-rechts sowie für oben-rechts und unten-links.

```
div {
  border-radius: 0px 15px 15px 0px;
}
```

Statt der border-radius Eigenschaft kann man auch die Detaileigenschaften border-top-left-radius, border-top-right-radius, border-bottom-left-radius und border-bottom-right-radius verwenden. Das besonder hierbei: Es lassen sich damit sogar unterschiedliche Werte für die horizontale und vertikale Rundung definieren.

```
div {
  width: 200px; height:30px;
  padding-left:25px;
  padding-top:20px;
  border-top-right-radius: 25px 50px;
  border-top-left-radius: 25px 45px;
  background: -moz-linear-gradient(rgb(255,204,153),
     rgb(255,204,51));
  box-shadow: #CCC 0px -5px 10px -5px;
}
```

Die Werte lassen sich auch mit einem Slash getrennt in der border-radius Eigenschaft defineren. Erst werden dabei alle horizontalen Rundungen genannt und nach dem Slash alle vertikalen Rundungen.

```
div {
  border-radius: 10px 150px 150px 10px /
     25px 25px 25px 25px;
}
```

Die Werte lassen sich sogar in Prozent angeben und beziehen sich dann auf die Höhe und Breite des Elements.

```
div {
  width:150px; height:200px;
  border-radius: 50%;
}
```

2.7.3. Multiple Backgrounds

Für die Verwendung von mehreren Hintergründen für ein Element ist nicht mal mehr eine neue Syntax notwendig. Vielmehr lassen sich einfach mit einem Komma getrennt nacheinander verschiedene Bilder, Positionen und Wiederholungseigenschaften angeben. Das erste definierte Bild liegt immer auf der obersten Ebene und alle weiteren liegen jeweils darunter. Sinnvoll kann das sein, wenn die oberen Bilder PNG Dateien mit Transparenzen sind oder wenn mehrere kleinere Bilder zu einem "Teppich" platziert werden sollen.

```
div {
  width: 500px; height: 500px;
  background-image: url(sign.png), url(map.png);
  background-position: 250px 250 px, left top;
  background-repeat: no-repeat, no-repeat;
}
```

Die Schreibweise lässt sich auch abkürzen.

```
div {
  background: url(sign.png) center bottom no-repeat,
     url(muster.png) top left repeat;
}
```

Registerkarten mit Multiple Backgrounds

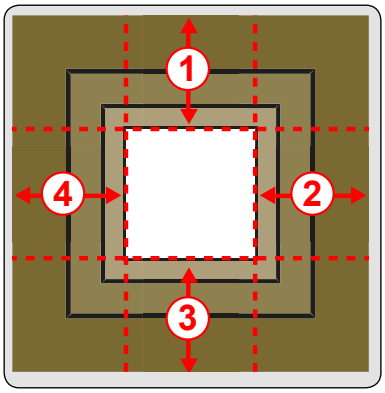
Multiple Backgrounds erleichtern auch die Erstellung von "Sliding Doors" – Registerkarten oder Tabs, die sich je nach Inhalt verbreitern und jeweils mit Bildern gestaltete Kanten verwenden.

```
<head>
<style>
  .tab ul li {
     float: left;
     display: block;
     height:25px;
     padding: 5px 20px 5px 20px;
     background:url(left.jpg) no-repeat top left,
        url(right.jpg) no-repeat top right,
        url(bg.jpg) repeat-x top left;

  }
</style>
</head>
<body>
<nav class="tab">
  <ul>
     <li><a href="#">Startseite</a>
     <li><a href="#">Produkte</a>
     <li><a href="#">Services</a>
     <li><a href="#">Kontakt</a>
  </ul>
</nav>
</body>
```

2.7.4. Bilder für die Rahmen – 9-Slice-Skalierung

Mit der border-image Eigenschaft kann man dem Rahmen einer Box eine Grafik für die Ecken und Kanten geben. Die Kanten können dabei verzerrt oder wiederholt werden, wenn sich die Abmessungen der Box verändern. Im Prinzip ähnelt es dem 9-Slice-Skalierungsprinzip.



Die Grafik sollte so gestaltet werden, dass die Ecken eine feste Größe haben und die Kanten skaliert oder wiederholt werden können. Der Wert nach der URL bestimmt den Abstand zur Kante der Grafik. Zuletzt kann über stretch oder repeat festgelegt werden, ob die Kanten skaliert oder wiederholt werden sollen.

```
.rahmen {
  border-width: 27px;
  -moz-border-image: url(border.jpg) 27 stretch;
}
```

Die Werte für die Verzerrung/Wiederholung können auch getrennt für die horizontalen und vertikalen Kanten, sowie für alle vier Kantenim Uhrzeigersinn angegeben werden.

2.8. Spaltenlayouts mit CSS Multi Columns

Mehrspaltige Texte werden ab dem Internet Explorer 10 unterstützt und Firefox, Chrome sowie Safari benötigen unbedingt noch Vendor-Prefixes. Im Grunde aber auch kein großes Problem, denn im Zweifelsfall zeigen Browser eben nur eine einzige Spalte an.

2.8.1. Textspalten mit fester Breite oder fester Anzahl

Nicht alle älteren Firefox Versionen kommen mit der verkürzten columns Schreibweise zurecht, daher sollte man eher auf column-width und column-count zurückgreifen.

```
.festeBreite {
  -webkit-column-widht: 300px;
}
.festeAnzahle {
  -webkit-column-count: 3;
}
.spalten {
  -webkit-columns: 3 300px;
}
```

Wird nur die Breite definiert, dann entscheidet der Browser, wie viele Spalten in das Element passen. Steht die Anzahl der Spalten fest, dann haben diese eine flexible Breite.

In der Regel ergibt es nicht viel Sinn mit columns sowohl die Breite, wie auch die Anzahl der Spalten zu definieren, denn die Browser reagieren jetzt unterschiedlich, um den verfügbaren Platz aufzufüllen. Opera ignoriert die Anzahl der Spalten; die anderen Browser ignorieren die Breite der Spalten.

2.8.2. Den Spaltenabstand definieren

Mit CSS lässt sich auch der Abstand der Spalten und eine Trennlinie formatieren.

```
article {
  width: 28em;
  -moz-column-width: 12em;
  -moz-column-gap: 1em;
  -moz-column-rule: 1px solid #DDD;
}
```

Die Breite der Trennlinie hat keinen Einfluss auf die Breite des Spaltenstegs und dieser wird tatsächlich nur zwischen den Spalten eingerechnet.

2.8.3. Spalten ausgleichen und überspannen

Die Eigenschaften column-fill und column-span erlauben es sehr flexible Artikel-Layouts zu definieren.

Der Eigenschaft column-span kann man eine Zahl für die Anzahl der überspannten Spalten geben oder das Schlüsselwort all. In diesem Fall läuft zum Beispiel eine Überschrift über alle Spalten. Um die Eigenschaft zurückzusetzen verwendet man das Schlüsselwort none.

Für column-fill stehen die Werte auto oder balance zur Verfügung. Der Wert balance ergibt nur Sinn, wenn die Höhe der Spalten festgelegt ist. Ansonsten wird der Inhalt automatisch ausgeglichen. Der Wert auto füllt die Textspalten nacheinander, sodass in der letzten eventuell mehr Weißraum bleibt während balance dafür sorgt, dass die Spalten gleichmäßig gefüllt werden.

```
article {
  height: 200px;
  -webkit-column-count:4;
  -webkit-column-rule: solid 1px black;
  column-fill: balance; /* nur Opera */
}
article h1 {
  -webkit-column-span:all;
}
```

2.9. Das Flexbox Box Modell

2.9.1. Flexibles Box Modell versus traditionelles Box Modell

Das flexible Box Modell – auch Flexbox genannt – vereinfacht das Design einer Webseite mit mehreren Spalten erheblich. Beim traditionellen Box Modell wird oft mit floats gearbeitet und die Verdrängung einer Spalte ergibt sich aus dessen Breite plus dem Rahmen und den Abständen. Das führt oft zu komplizierten Berechnungen – insbesondere bei Liquid-Layouts. Das ganze ist noch sehr experimentell und muss auf jeden Fall mit Vendor-Prefixes versehen werden.

Eine Unterstützung der Flexbox ist ab dem Internet Explorer 10 vorgesehen, allerdings wird das Prefix -ms benötigt. Bis auf Opera 12.1. benötigen auch alle anderen Browser Vendor Prefixes.

Leider wurde die Syntax mehrmals überarbeitet, sodass man einige Werte doppelt anwenden muss, damit alle Webkit und Mozilla Browserversionen damit umgehen können. Flexiejs stellt unter http://flexiejs.com/ ein gutes Polyfill bis Internet Explorer 6 bereit.

Bis vor kurzem war als Displayeigenschaft für das Flexbox Modell der Wert "box" vorgesehen. Laut Spezifikation wurde dieser in "flexbox" oder "inline-flexbox" geändert.
Im Firefox wird ein Flexbox Element derzeit automatisch als inline Element angesehen und füllt nicht mehr 100% der Breite aus.

Der Flexbox Container

Als Flexbox bezeichnet man einen Container, der seinerseits mehrere Spalten oder Zeilen enthalten soll. Der Container wird mit dem flex bzw. inline-flex-Wert für die display Eigenschaft versehen. Der Wert flex erzeugt einen Block-Level Container; inline-flex einen Inline Container.

```
section {
  display: -moz-flex;
  display: -webkit-flex;
  display: -webkit-flex;
  display: flex
}
```

Die Kindelemente einer Flexbox

Die einzelnen Kindelemente bzw. Container erhalten die flex-Eigenschaft für die Aufteilung des zur Verfügung stehenden Platzes. Die flex-Werte aller Kindelemente werden zusammengerechnet und dann relativ für jedes Element aufgeteilt.

```
article.a {
  -moz-flex: 2; /* Firefox Syntax */
  -webkit-flex: 2; /* Safari & Chrome Syntax */
  flex: 2;
  border: 5px solid green;
}
article.b {
  -moz-flex: 3;
  -webkit-flex: 3;
  flex: 3;
  border: 5px solid red;
}
```

Bei diesem Beispiel wird der Container horizontal in fünf gleich große Teile aufgeteilt. Der Artikel mit der Klasse "a" nimmt ⅖ des Platzes und der Artikel mit der Klasse "b" nimmt ⅗ des Platzes ein. Der Vorteil dieses Prinzips ist, dass es sich mit beliebig vielen Spalten durchführen lässt.

```
<section>
  <article class="a">Erster Artikel</article>
  <article class="b">Zweiter Artikel</article>
</section>
```

Kindelemente mit einer festen Breite werden vorher von dem zur Verfügung stehenden Platz abgezogen. So teilen sich die flex-Elemente nur den übrigen Platz auf.

Ausrichtung der Kindelemente

Dem Flexbox Container lassen sich weitere Eigenschaften zuweisen, die bestimmen, wie die Kindelemente angeordnet werden.

- **Horizontale oder vertikale Anordnung der Boxen:** flex-direction kann die Werte row (von links nach rechts), row-reverse, column (von oben nach unten) oder column-reverse erhalten.
- **Ein- oder mehrere Zeilen/Spalten:** mit der flex-wrap Eigenschaft und den Werten nowrap, wrap und wrap-reverse lassen sich Umbrüche bei den Kindelementen erzeugen.
- **Individuelle Reihenfolge:** den Kindelementen kann über die order-Eigenschaft eine individuelle Reihenfolge gegeben werden. Der jeweils niedrigste Wert steht dabei zuerst im Flex-Container.
- **Aufteilung in Multiline Flexcontainern:** align-content bestimmt die vertikale Ausrichtung der Kindelemente im Flexbox Container und kann die Werte flex-start (oben), flex-end (unten), center (mittig), space-between bzw. space-around (gleichmäßige Verteilung der Abstände) und stretch (die Boxen nehmen die gesamte Höhe ein) erhalten. Der Standardwert ist stretch.
- **Ausrichtung:** align-items bestimmt die horizontale Ausrichtung der Kindelemente und kann die Werte flex-start (linksbündig), flex-end (rechtsbündig), center (zentriert), baseline und stretch (ausgeglichen) enthalten. Das ist natürlich nur interessant, wenn die Kindelemente als inline definiert sind oder eine feste Breite haben. Tatsächlich kann die Ausrichtung mit der Eigenschaft align-self für jedes Kindelement eines Flex-Containers überschrieben werden.

```
section {
  display: -webkit-flex;
  -webkit-flex-direction: column;
  -webkit-align-content: flex-end;
}
```

Der Flexplorer unter http://bennettfeely.com/flexplorer/ ist eine gute Möglichkeit Flexbox Layouts anzulegen.

2.10. Template Layout Modul

Das Template Layout Modul ist eine weitere sehr praktische Layouttechnik. Direkt unterstützt wird diese Vorgehensweise zwar in keinem Browser, aber es gibt ein sehr gutes jQuery Polyfill.

Bei dieser Layoutvariante wird ein Raster von Slots über die display Eigenschaft eines Elternelements definiert. Kindelemente können diese Slots dann über die position Eigenschaft ausfüllen.

Ein Aufteilungsbeispiel für eine Kopfzeile, einen dreispaltigen Inhaltsbereich und eine Fußzeile sieht folgendermaßen aus:

```
body {
  display:   "aaa"
             "bcd"
             "fff";
}
```

Die Seite wird dadurch horizontal in drei gleich große Bereiche aufgeteilt. Möchte man eine andere Aufteilung wäre auch folgendes Muster denkbar:

```
body {
  display:   "aaaaaa"
             "bbcccd"
             "ffffff";
}
```

Die Zuweisung von Kindelementen in diese Bereiche sieht dann so aus:

```
#header   { position: a; }
#navi     { position: b; }
#content  { position: c; }
#sidebar  { position: d; }
#footer   { position: f; }
```

Es ist auch möglich Abstände zu definieren. Statt einem Buchstaben setzt man in der Display Eigenschaft einfach einen "." und schon hat man einen Bereich, der nicht gefüllt werden kann.

Das Template Layout Modul wird momentan noch nicht unterstützt, aber mit folgendem jQuery Polyfill sollte der Einsatz kein Problem mehr darstellen:

Das jQuery Script findet man unter http://code.google.com/p/css-template-layout/

```
<head>
  <script src="https://ajax.googleapis.com/ajax/libs/
jquery/1.7.1/jquery.min.js"></script>
  <script src="jquery.tpl_layout1.1.6.min.js">
  <script>
     $(document).ready(function(){ /
        $.setTemplateLayout();
     });
  </script>
</head>
```

2.11. Transformationen in 2D und 3D

Mit den 2D und 3D Transformationen lassen sich Elemente verzerren, drehen oder verschieben. In manchen Browsern passiert das sogar über die GPU beschleunigt und ist somit wesentlich effizienter als über JavaScript DOM Elemente zu verzerren.

2.11.1. 2D Transformationen

Die 2D Transformationseigenschaft ist in den aktuellen Browsern gut implementiert – wenn auch noch über Vendor-Prefixes (auch im Internet Explorer). Folgende Transformationsfunktionen sind für die Eigenschaft möglich:

- matrix(a,b,c,d,e,f)
- translate(n) oder translate(x, y) bzw. translateX(n) und translateY(n)
- scale(n) oder scale(x,y) bzw. scaleX(n) und scaleY(n)
- rotate(angle)
- skewX(angle) und skewY(angle)

Mit folgendem Code verschiebt man ein Element um 100px nach unten und 200px nach rechts:

```
div {
  transform: translate(100px, 200px);
}
```

In der transform Eigenschaft können auch mehrere Transformationen nacheinander durchgeführt werden.

```
div {
  transform: rotate(45deg) scale(1.25, 1.25);
}
```

Den Ursprung einer Transformation definieren

Normalerweise gehen alle Transformationen von der Mitte eines Elements aus. Über transform-origin wird zunächst angegeben auf welchen Punkt sich die Werte beziehen und danach jeweils um wie weit dieser Punkt verschoben wird.

Der transform-origin Eigenschaft können jeweils Bezugspunkte auf der x-Achse und der y-Achse angegeben werden. Zur Verfügung stehen Schlüsselworte wie left, center, right für den ersten Wert (x-Achse) und top, center, bottom für den zweiten Wert (y-Achse). Alternativ können beide Werte auch in Einheiten oder Prozentual angegeben werden.

```
img {
  -moz-transform-origin: right bottom;
  -moz-transform: rotate(15deg);
}
```

Werte in Prozent oder Pixel beziehen sich auf die obere linke Ecke eines Elements.

Offset eines Ursprungs definieren

Für transform-origin lassen sich laut Spezifikation sogar vier Werte angeben. Dazu wird nach jedem Schlüsselwort noch eine negative oder positive Verschiebung in x- oder y-Richtung angegeben.

```
img {
  -webkit-transform-origin: right -50px bottom -50px;
  -webkit-transform: rotate(25deg) scale(1.2, 1.2);
}
```

Die Möglichkeit einen Offset des Ursprungs zu definieren wird momentan noch von keinem Browser unterstützt.

2.11.2. 3D Transformationen

3D Transformationen sind den 2D Transformationen sehr ähnlich. Die transform-origin Eigenschaft bekommt aber noch einen dritten Wert für die Z-Tiefe der in absoluten Einheiten angegeben werden kann.

Zusätzlich gibt es noch die Funktionen translate3d(x,y,z), scale3d(x,y,z) und rotate3d(x-Richtung, y-Richtung, z-Richtung, Winkel).

Um zu bestimmen, wie die Kindelemente eines 3D Containers dargestellt werden gibt es die Eigenschaft transform-style, die auf "flat" oder "preserve-3d" gesetzt werden kann. Die Eigenschaft perspective bestimmt den Grad der perspektivischen Verzerrung wobei der Wert none keine Verzerrung aufweist und höhere Werte jeweils eine weitwinkligere Ansicht ergeben.

Der Punkt von dem ein 3D Element betrachtet wird, wird über die Eigenschaft perspective-origin in x- und y-Richtung angegeben. Zuletzt kann die Eigenschaft backface-visibility auf "visible" oder "hidden" eingestellt werden und bestimmt dann, was auf der Rückseite zu sehen ist.

2.12. CSS Übergänge mit Transitions

CSS Übergänge erlauben eine "Überblendung" von verschiedenen CSS Eigenschaften nach einem definierten Ereignis. Das kann gerade in Verbindung mit Transformationen sehr spannend sein. Für alle Transitions müssen Vendor-Prefixes verwendet werden.

Transitions benötigen eines der folgenden Ereignisse bzw. Pseudoklassen, um ausgelöst zu werden:

- :link und :visited
- :hover
- :active (Click-Ereignis)
- :focus (ausgewählte Elemente)

Mit der transition-property Eigenschaft wird festgelegt welche CSS Eigenschaften animiert werden sollen. Über die transition-duration wird die Dauer für alle oder für jede Eigenschaft einzeln angegeben. Nur Eigenschaften, die numerische Werte enthalten dürfen animiert werden. Eigenschaften, wie display funktionieren nicht.

Eine Liste aller erlaubten Eigenschaften findet man unter http://www.w3.org/TR/css3-transitions/#animatable-properties-

```
div {
  width: 100px;
  height: 100px;
  white-space: nowrap;
  overflow: hidden
  background: #CCC;
  -moz-transition-property: width, height, background;
  -moz-transition-duration: 1s, 1s, 300ms;
}
div:hover {
  width: 400px;
  height: 400px;
  background: lime;
}
```

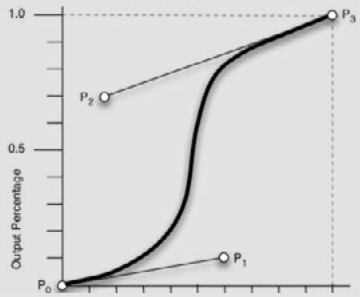
Die transition-property kann auch den Wert all enthalten. In diesem Fall werden alle Eigenschaften überblendet.

Mehr Dynamik

Abbrems- und Beschleunigungsverhalten lassen sich über die transition-timing-function Eigenschaft definieren. Folgende Werte sind möglich:

- **ease** entspricht cubic-bezier(0.25, 0.1, 0.25, 1.0)
- **linear** entspricht cubic-bezier(0.0, 0.0, 1.0, 1.0)
- **ease-in** entspricht cubic-bezier(0.42, 0, 1.0, 1.0)
- **ease-out** entspricht cubic-bezier(0, 0, 0.58, 1.0)
- **ease-in-out** entspricht cubic-bezier(0.42, 0, 0.58, 1.0)
- **cubic-bezier(P1-x, P1-y, P2-x, P2-y)**

Die cubic-bezier Funktion enthält vier Werte jeweils zwischen 0.0 und 1.0.



Einen guten Konfigurator findet man unter http://cubic-bezier.com

Verzögerungen

Die Animationen können auch mit einer Verzögerung einsetzen. Dies ermöglicht die Eigenschaft transition-delay, der man für jede Animationseigenschaft einen Wert in Sekunden (s) oder Millisekunden (ms) geben kann.

```
img {
  -moz-transition-property: -moz-transform;
  -moz-transition-duration: 1s;
  -moz-transition-delay: 200ms;
  -moz-transition-timing-function: ease;
}
img:active {
  -moz-transform: scale(1.2,1.2) rotate(15deg);
}
```

CSS Transitions und JavaScript Ereignisse

CSS Transitions können auch ausgelöst werden, wenn man via JavaScript die Klassen austauscht.

```
<head>
  <script src="https://ajax.googleapis.com/ajax/libs/
jquery/1.7.1/jquery.min.js"></script>
  <script>
     $(document).ready(function() {
          $("img").click(function(){
             $(this).toggleClass("clicked");
          });
       });
  </script>
  <style>
     img {
        -moz-transform-origin:left top;
        -moz-transition-property: -moz-transform;
        -moz-transition-duration: 1s;
        -moz-transition-delay: 200ms;
        -moz-transition-timing-function:ease;
     }
     img.clicked {
        -moz-transform: scale(1.2,1.2) rotate(15deg)
          translate(100px, -25px);
     }
  </style>
</head>
<body>
  <img src="test.jpg" width="580" height="360">
</body>
```

2.13. Animationen

Animationen können im Gegensatz zu Transition mehrere Zustände nacheinander verändern. Dies erfolgt über sogenannte Keyframes. Diese Keyframes werden in einer eigenen Regel definiert. Jede @keyframes Regel erhält einen eigenen Namen.

```
@keyframes blend {
  from { background:purple; }
  to { background:red; }
}
```

Dem gewünschten Element wird nun über die animation-name Eigenschaft der Name der Animation mitgeteilt. Folgende Eigenschaften definieren weitere Animationsparameter:

- **animation-duration** definiert die Dauer in Sekunden (s) oder Millisekunden (ms).
- **animation-iteration-count** gibt die Anzahl der Wiederholungen als Ganzzahl oder für unendliche Wiederholungen das Schlüsselwort infinite an.
- **animation-timing-function** wendet verschiedene Abbrems -und Beschleunigungsfunktionen an; darunter linear, ease, ease-in, ease-out, ease-in-out oder cubic-bezier(x1, y1, x2, y2).
- **animation-direction** gibt an ob die Animation immer vorwärts (normal) abgespielt wird oder in der Schleife abwechselnd vorwärts und rückwärts abgespielt wird (alternate).
- **animation-delay** gibt einen Verzögerungswert zum Start der Animation in Sekunden oder Millisekunden an.

```
div {
  animation-name: blend;
  animation-duration: 2s;
  animation-iteration-count:infinite;
  animation-timinig-function: linear;
  animation-direction: alternate;
}
```

Keyframes

Statt den Werten from und to für Anfang und Ende der Animation können mehrere Prozentwerte angegeben werden. Gleichzeitig können natürlich auch mehrere Eigenschaften definiert werden.

```
@keyframes demo {
  0%   { left: 100px, top: 50px }
  20%  { left: 200px, top: 100px }
  100% { left: 50px, top: 20px }
}
```

Von der Theorie zur Praxis mit Vendor-Prefixes

Jedem Keyframe kann zusätlich auch eine eigene animation-timing-function gegeben werden.

```
@-moz-keyframes drehen {
   from {
      -moz-transform: rotate(0deg);
      -moz-animation-timing-function: ease-out;
  }
   30% {
      -moz-transform: rotate(45deg);
      -moz-animation-timing-function: ease;
  }
  to {
     -moz-transform: rotate(0deg);
     -moz-animation-timing-function: ease-in;
  }
}
img {
  -moz-animation-name: drehen;
  -moz-animation-duration: 2s;
  -moz-animation-iteration-count:infinite;
}
```

3. Neue Javascript Funktionen in HTML5

Den Status der verschiedenen JavaScript APIs findet man unter http://www.w3.org/standards/techs/js#w3c_all

Genau wie bei CSS3 sind die neuen APIs in einzelne Module aufgeteilt. Zu den wichtigsten neuen Modulen gehören folgende:

- Web Storage
  Daten direkt im Browser speichern
  http://www.w3.org/TR/webstorage/

- Indexed Database API
  Speichern umfangreicherer Daten im Browser
  http://www.w3.org/TR/IndexedDB/

- Geolocation API
  Ermöglicht das Abrufen von Geodaten aus dem Gerät oder Browser
  http://www.w3.org/TR/geolocation-API/

- Drag & Drop
  http://www.w3.org/TR/html5/dnd.html

- File API – Directories and System / Writers
  Dateien und Verzeichnisse lesen
  http://www.w3.org/TR/FileAPI/
  http://www.w3.org/TR/file-writer-api/
  http://www.w3.org/TR/file-system-api/

- Web Notifications
  Erlaubt die Darstellung einfacher Benachrichtigungen
  http://www.w3.org/TR/notifications/

- Web Workers
  Hintergrundaufgaben für Webanwendungen im Browser ausführen
  http://www.w3.org/TR/workers/

- WebSocket API
  Zweiwege-Kommunikation mit einem Remote Host via Sockets
  http://www.w3.org/TR/workers/

- DeviceOrientation Event
  Ermöglicht die Erfassung physikalischer Richtungs und Beschleunigungsinformationen eines Gerätes
  http://www.w3.org/TR/orientation-event/

3.1. Web Storage

Mit Web Storage kann man Daten im Browser speichern, die zu einem späteren Zeitpunkt innerhalb einer Session oder auch nach beenden des Browsers wieder abgerufen werden können.

3.1.1. localStorage Objekte

Der localStorage bleibt bestehen und wird nicht gelöscht, wenn der Anwender den Browser bzw. das Tab schließt. Ein solches localStorage Objekt ist sehr einfach erstellt und wieder ausgelesen:

```
<script>
  localStorage.nachname="Brencher";
  document.write(localStorage.nachname);
</script>
```

Eine Funktion, die speichert, wie oft der Besucher bereits die Seite aufgerufen hat würde folgendermaßen aussehen.

```
<script>
  if (localStorage.besuchszaehler)
  {
     localStorage.besuchszaehler=Number(
        localStorage.besuchszaehler) +1;
     document.write("Vielen Dank f&uuml;r Ihren " +
        localStorage.besuchszaehler+".ten Besuch.");
  }
  else
  {
     localStorage.besuchszaehler=1;
     document.write("Herzlich willkommen!");
  }
</script>
```

3.1.2. sessionStorage Objekte

Die Handhabung der sessionStorage Objekte ist identisch mit dem localStorage Objekt. Allerdings ist dieses Objekt gelöscht, wenn der Anwender den Browser oder das Tab schließt.

```
<script>
  sessionStorage.nachname="Brencher";
  document.write(sessionStorage.nachname);
</script>
```

3.1.3. Web Storage verwalten

Mit den Methoden getItem('key') setItem('key', 'value'), removeItem('key') und clear() können Web Storage Objekte verwaltet werden. Generell wird Web Storage einer Domäne zugeordnet. Verschiedene Webdokumente können innerhalb der Domäne auf die gleichen Objekte zugreifen.

Mit dieser Schaltfläche lässt sich das Web Storage Objekt löschen:

```
<input id="clearBtn" type="button" value="clear"
  onclick="localStorage.clear();
  window.location.reload();">
```

Über die Eigenschaften length lässt sich die Anzahl der Elemente auslesen. Die Methode key(i:int) gibt einen String zurück, der den Namen des Elementes enthält (nicht den Wert).

```
sessionStorage.setItem('email', 'info@inlite.de');
sessionStorage.length; // gibt in diesem Fall 1 zurück
sessionStorage.key(0); // gibt 'email' zurück
sessionStorage.getItem('email'); // info@inlite.de
sessionStorage.removeItem('email'); // Element löschen
sessionStorage.clear(); // löscht alle Einträge
```

3.1.4. Web Storage Ereignisse

Die Dokumente müssen zum Testen der Ereignisse auf einem Webserver liegen. Lokal wird das Ereignis nicht ausgegeben!

Bei einer Änderung des localStorage Objektes über eine der oben genannten Methoden werden in anderen Tabs oder Browserfenstern Ereignisse ausgegeben. Darüber lassen sich die Werte synchronisieren oder der alte und neue Wert vergleichen.

```
<button onclick="localStorage.setItem('Zugriff',
  Number(localStorage.Zugriff)+1)">Change</button>
<button onclick="localStorage.clear()">Clear</button>
<script>
  localStorage.setItem('Zugriff', '1');
  var storageHandler = function (storageEvent) {
     alert('Die Eigenschaft '
        + storageEvent.key
        + ' wurde aktualisiert. Statt '
        + storageEvent.oldValue
        + ' ist der neue Wert jetzt: '
        + storageEvent.newValue);
  };
  window.addEventListener('storage', storageHandler);
</script>
```

LocalStorage und SessionStorage Keys und Values können im Chrome Browser über Darstellung > Entwickler > Entwickler-Tools > Resources eingesehen werden.

3.2. Geolocation

Mit der Geolocation API lassen sich die Ortsangaben zu einem Gerät darstellen. In Verbindung mit einer Google Map lassen sich so Ortsbestimmungen auf einer Karte darstellen. Diese Funktion ist besonders in mobilen Browsern sehr interessant.

```
<!DOCTYPE html>
<html>
<head>
  <meta name="viewport" content="initial-scale=1.0,
     user-scalable=no">
  <style>
     html, body, #map_canvas {
        height: 100%; margin:0; padding:0;
     }
  </style>
  <script src="http://maps.googleapis.com/maps/api/
     js?key=YOUR_API_KEY&sensor=true"></script>
  <script>
  var geoCall = function(position) {
     var latLong = new google.maps.LatLng(
        position.coords.latitude,
        position.coords.longitude);
     var optionen = {
        center: latLong,
        zoom: 12,
        mapTypeId: google.maps.MapTypeId.ROADMAP
     };
     var karte = new google.maps.Map(
        document.getElementById("kartenCanvas"),
        optionen
     );
     var marker = new google.maps.Marker({
        position: latLong,
        map: karte,
        title:"Hier bin ich"
     });
  }
  </script>
</head>
<body>
  <div id="kartenCanvas" style="width:100%;
     height:100%"></div>
  <script>
  if (navigator.geolocation) {
     var nav = navigator.geolocation.
        watchPosition(geoCall);
  }
  </script>
</body>
</html>
```

Den Google API Key gibt es unter https://code.google.com/apis/console/b/0/?pli=1
Mehr Infos dazu unter http://code.google.com/apis/maps/documentation/javascript/tutorial.html

Die coords Eigenschaft enthält folgende Werte:
Breitengrad: coords.latitude
Längengrad: coords.longitude
Höhe: coords.altitude
Genauigkeit der Höhenmessung: coords.altitudeAccuracy
Richtung: coords.heading
Geschwindigkeit: coords.speed

Mit **Date(position.timestamp)** lässt sich der Zeitpunkt der Abfrage bestimmen.

geolocation.getCurrentPosition() ruft die Daten nur ein einziges mal ab.

geolocation.clearWatch(nav) stoppt die Überwachung wieder.

Zusätzlich können weitere Callback Handler aufgerufen werden:
watchPosition(positionHandler, errorHandler, optionHandler)

3.3. Drag & Drop

Bilder und Texte im Browser per Drag und Drop bewegen oder Inhalte vom Desktop auf die Webseite ziehen. Der umgekehrte Weg um Inhalte vom Browser auf den Desktop zu ziehen wird momentan nur von Google Chrome unterstützt.

Erlauben, dass ein Element gezogen werden darf:

```
<img src="icon.png" draggable="true">
```

3.3.1. Drag und Drop im Browser

Bei einem Drag und Drop treten folgende Ereignisse auf, die einzeln abgerufen werden können:

- dragstart
- drag
- dragenter
- dragleave
- dragover
- drop
- dragend

Für die Drag und Drop Aktion wird ein dataTransfer Objekt erstellt. Dieses beinhaltet das zu ziehende Element und die Daten. Es wird beim dragstart Ereignis erstellt und wird beim drop Ereignis wieder ausgelesen.

Während des Ziehens, kann statt der abgedimmten Browserversion mit der Methode setDragImage() auch ein Icon dargestellt werden. Die Methode setDate(format, data) legt fest von welchem Typ das gezogene Element ist und welche Daten es enthält.

Die dropEffect Eigenschaft gibt an ob der Browser eine passende Visualisierung für eine Bewegung (move), einen Kopiervorgang (copy) oder eine Verknüpfung (link) darstellt. Die Eigenschaft effectAllowed definiert den Typ des Drag und Drop Vorganges. Zur Verfügung stehen copy, copyLink, copyMove, link, linkMove, all und uninitialized.

```
<script>
var dragIcon = document.createElement('img');
dragIcon.src = 'dragIcon.png';
document.addEventListener('dragstart', function(e) {
  e.dataTransfer.setData('text/plain', 'Hallo');
  e.dataTransfer.setDragImage(dragIcon, -10, -10);
  e.dataTransfer.dropEffect='copy';
  e.dataTransfer.effectAllowed = 'copy';
}, false);
</script>
```

3.3.2. Das Drop Ereignis

Wenn das Objekt wieder losgelassen wird, dann wird ein dragend Ereignis losgelassen. Soll ein anderes Element das Objekt entgegennehmen, dann muss dieses Element auf das drop Ereignis warten. Damit ein drop Ereignis ausgelöst werden kann, müssen die dragenter und dragover Ergebnisse gecancelt werden. Ansonsten verhindern diese das drop Ereignis.

Das folgende Beispiel könnte eine Warenkorbunktion sein, bei der man Produktbilder in einen Warenkorb ziehen kann.

```
<img src="icon.png" draggable="true"
  ondragstart="dragStartHandler(event)">
<div id="dropTarget"
  ondragenter="cancel(event)"
  ondragover="cancel(event)"
  ondrop="dropHandler(event)"></div>

<script>
function cancel(event) {
  event.preventDefault();
}
function dragStartHandler(event) {
  var data = event.dataTransfer.setData('text/plain',
     event.target.src);
  event.dataTransfer.dropEffect='copy';
  event.dataTransfer.effectAllowed = 'copy';
}
function dropHandler(event) {
  var data = event.dataTransfer.getData('text/plain');
  var image = document.createElement('img');
  image.setAttribute('src', data)
  event.target.appendChild(image)
}
</script>
```