Inhalt

[1 Bedeutung 3](#_Toc354385953)

[2 Neue Tags zur besseren Strukturierung 3](#_Toc354385954)

[3 Formulare 8](#_Toc354385955)

[3.1 Neue Eingabetypen 9](#_Toc354385956)

[3.2 Neue Eingabeattribute 10](#_Toc354385957)

[3.3 Browserbasierte Validierung 12](#_Toc354385958)

[3.3.1 Das Attribut "required" 12](#_Toc354385959)

[3.3.2 Das Attribut "pattern" 12](#_Toc354385960)

[3.3.3 Das Attribut "formnovalidate" 12](#_Toc354385961)

[3.3.4 Einschränkungsvalidierungs-API 13](#_Toc354385962)

[3.4 Gemeinsame Verwendung 13](#_Toc354385963)

[4 Professionelles Layout 16](#_Toc354385964)

[4.1 Optimierte Darstellung von Tabelllen 16](#_Toc354385965)

[4.1.1 CSS3 und „nth-child“ 16](#_Toc354385966)

[4.1.2 CSS3 und „nth-of-type“ 17](#_Toc354385967)

[4.2 So werden Texte einfach mehrspaltig 19](#_Toc354385968)

[4.2.1 Spaltenbreite mit CSS3 bestimmen 19](#_Toc354385969)

[4.2.2 Spaltenanzahl mit CSS3 bestimmen 19](#_Toc354385970)

[4.2.3 Abstand und Trennlinie festlegen 19](#_Toc354385971)

[4.2.4 Umbrüche festlegen 20](#_Toc354385972)

[4.2.5 Unterstützung der CSS3-Eigenschaften durch aktuelle Browser 21](#_Toc354385973)

[4.3 CSS: DIV zentrieren 22](#_Toc354385974)

[4.3.1 Schritt 1 23](#_Toc354385975)

[4.3.2 Schritt 2 23](#_Toc354385976)

[4.3.3 Schritt 3 24](#_Toc354385977)

[4.4 CSS3-Buttons und Boxen ohne Grafiken erstellen 24](#_Toc354385978)

[4.4.1 Ein schlichter Button ohne viel Schnick-Schnack 24](#_Toc354385979)

[4.4.2 Schritt 4: Hover-Style 26](#_Toc354385980)

[4.5 Animationen mit Transitions 30](#_Toc354385981)

[4.6 Transform 33](#_Toc354385982)

[5 Zeichnen mit Canvas 41](#_Toc354385983)

[5.1 Wie bindet man das <canvas>-Element ein? 41](#_Toc354385984)

[5.2 Was kann man genau alles mit <canvas> anstellen? 42](#_Toc354385985)

[5.2.1 Canvas-Beispiel 1: Ein Bild drehen 42](#_Toc354385986)

[5.2.2 Beispiel 2: Eine kleine Animation 43](#_Toc354385987)

[5.2.3 Beispiel 3: Paint 44](#_Toc354385988)

[5.3 Statistiken grafisch darstellen 45](#_Toc354385989)

[6 Video und Audio einbetten 46](#_Toc354385990)

[6.1 Attribute des HTML-TAGs <video> 47](#_Toc354385991)

[6.2 Audio einbetten 48](#_Toc354385992)

[7 Sonstige Neuerungen in HTML5 49](#_Toc354385993)

[7.1 iFrames 49](#_Toc354385994)

[7.2 Sonstige Erweiterungen HTML5 50](#_Toc354385995)

[7.2.1 WebStorage und Datenbanken 50](#_Toc354385996)

[7.2.2 WebSocket 51](#_Toc354385997)

[8 Einführung in Jquery 52](#_Toc354385998)

[8.1 Das Framework 52](#_Toc354385999)

[8.2 Selektoren und Funktionen 53](#_Toc354386000)

[8.3 Reagieren auf Events 56](#_Toc354386001)

[8.4 DOM-Manipulation 56](#_Toc354386002)

[8.5 Animationen 57](#_Toc354386003)

[8.5.1 Praktisches Beispiel: Slideshow 59](#_Toc354386004)

[8.5.2 Praktisches Beispiel „tooltipp“ 60](#_Toc354386005)

[8.6 Professionelle Jquery Anwendungen 60](#_Toc354386006)

[8.6.1 Puzzle 60](#_Toc354386007)

[8.6.2 Drag & drop 63](#_Toc354386008)

[9 Werkzeuge 63](#_Toc354386009)

[10 Quellen & Links: 64](#_Toc354386010)

# Bedeutung

HTML5 & CSS3 entwickeln sich zum Standard im Web und **sollen** auf allen Browsern inkl. der mobilen Geräte laufen. Besonders letztere unterliegen einer rasanten Entwicklung und weisen enorme Steigerungsraten auf.

Leider musste ich im letzten Jahr feststellen, dass es mit der Normierung und Allgemeingültigkeit noch nicht so weit her ist.

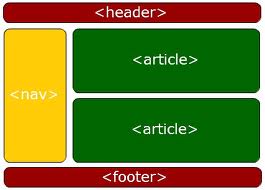
Die bedeutendsten Verbesserungen sind:

* Neue Tags zur besseren Strukturierung
* Multimedia mit weniger Plugins
* Bessere Anwendungen ohne Drittanbieter
* Clientseitige Speicherung
* Bessere Benutzeroberflächen und Formulare
* Neue CSS Selektoren
* Visuelle Effekte

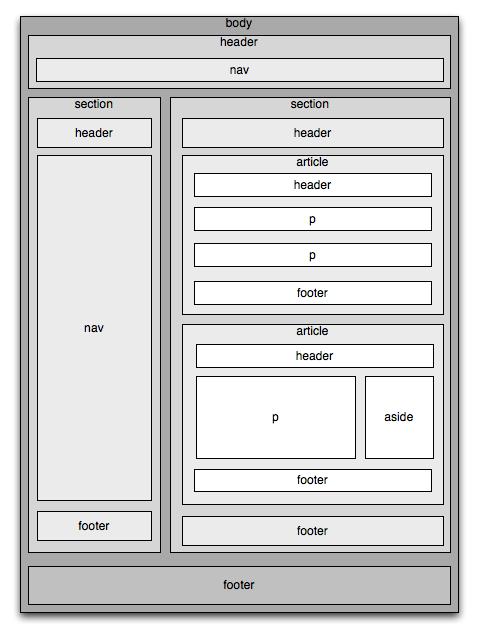
# Neue Tags zur besseren Strukturierung

Jede Seite beginnt mit <!DOCTYPE html>

Grober Überblick



Detaillierte Struktur



Erste Aufgabenstellung:



Die Texte finden Sie in den Beilagen.

<!DOCTYPE html >

<head>

<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />

<title>AwesomeCo </title>

</head>

<style>

body{

width:960px;

margin:15px auto;

font-family: Arial, "MS Trebuchet", sans-serif;

}

p{

margin:0 0 20px 0;

}

#page\_header nav ul, #page\_footer nav ul{

list-style: none;

padding: 0;

}

#page\_header nav ul li, #page\_footer nav ul li{

margin: 0 20px 0 0;

display:inline;

}

#posts{

float: left;

width: 74%;

}

aside{

float: right;

width: 35%;

margin-left: 5%;

font-size: 20px;

}

#sidebar{

float: left;

width: 25%;

}

#page\_footer{

clear: both;

width: 100%;

text-align: center;

}

</style>

<body>

<header id="page\_header">

<h1>AwesomeCo Blog!</h1>

<nav>

<ul>

<li> <a href "\*"> Latest Posts </a></li>

<li> <a href "\*"> Archives </a></li>

<li> <a href "\*"> Contributors </a></li>

<li> <a href "\*"> Contact Us </a></li>

</ul>

</nav>

</header>

<section id = "posts">

<article>

<header>

<h2>How Many Should We Put You Down For?</h2>

<p>Posted by Brian on

<time datetime="2010-10-01T14:39">October 1st, 2010 at 2:39PM</time>

</p>

</header>

<p>

The first big rule …

</p>

<aside>

<p>

"Never give someone a chance …."

</p>

</aside>

<p>

One way you can keep ….

</p>

<footer>

<p><a href="comments"><i>25 Comments</i></a> ...</p>

</footer>

</article>

</section>

<section id="sidebar">

<nav>

<h3>Archives</h3>

<ul>

<li><a href="2010/10">October 2010</a></li>

<li><a href="2010/09">September 2010</a></li>

<li><a href="2010/08">August 2010</a></li>

<li><a href="2010/07">July 2010</a></li>

<li><a href="2010/06">June 2010</a></li>

<li><a href="2010/05">May 2010</a></li>

<li><a href="2010/04">April 2010</a></li>

<li><a href="2010/03">March 2010</a></li>

<li><a href="2010/02">February 2010</a></li>

<li><a href="2010/01">January 2010</a></li>

</ul>

</nav>

</section>

<footer id="page\_footer">

<p>&copy; 2010 AwesomeCo.</p>

<nav>

<ul>

<li><a href="http://awesomeco.com/">Home</a></li>

<li><a href="about">About</a></li>

<li><a href="terms.html">Terms of Service</a></li>

<li><a href="privacy.html">Privacy</a></li>

</ul>

</nav>

</footer>

</body>

</html>

# Formulare

Nicht viele lassen sich für Formulare begeistern, HTML5 bringt jedoch einige wesentliche Verbesserungen mit sich, sowohl für den Entwickler des Formulars als auch für die Nutzer, die es ausfüllen. Neue Elemente, Attribute, Eingabetypen, die browserbasierte Validierung, CSS3-Stiltechniken sowie das FormData-Objekt vereinfachen die Erstellung von Formularen und sorgen für den nötigen Spaß.

HTML5 bietet fünf neue Eingabe- und Formularelemente.

| **Element** | **Verwendungszweck** | **Hinweise** |
| --- | --- | --- |
| [progress](http://dev.w3.org/html5/spec/Overview.html#the-progress-element) | Gibt an, inwieweit eine Aufgabe abgeschlossen ist | Das Element progress könnte zum Beispiel den Fortschritt einer Datei anzeigen, die gerade hochgeladen wird. |
| [meter](http://dev.w3.org/html5/spec/Overview.html#the-meter-element) | Stellt einen skalaren Messwert innerhalb eines bekannten Bereichs dar | Das Element meter könnte zum Beispiel für eine Temperatur- oder Gewichtsangabe verwendet werden. |
| [datalist](http://dev.w3.org/html5/spec/Overview.html#the-datalist-element) | Stellt eine Auswahl von option-Elementen dar, die zusammen mit dem neuen Attribut list zur Erstellung von Dropdown-Menüs verwendet werden kann | Wenn das Eingabefeld mit der zugehörigen Datenliste fokussiert wird, erscheint ein Dropdown-Menü, das Werte aus der datalist enthält. |
| [keygen](http://dev.w3.org/html5/spec/Overview.html#the-keygen-element) | Ein Steuerungselement zur Generierung eines Schlüsselpaars | Beim Senden des Formulars wird der private Schlüssel im lokalen Schlüsselspeicher abgelegt, während der öffentliche Schlüssel an den Server gesendet wird. |
| [output](http://dev.w3.org/html5/spec/Overview.html#the-output-element) | Zeigt die Ergebnisse einer Berechnung an | Zum Beispiel könnte mithilfe des output-Elements die Summe der Werte zweier Eingabeelemente angezeigt werden. |

## Neue Eingabetypen

HTML5 bietet dreizehn neue Eingabetypen. Wenn diese in einem Browser angezeigt werden, der keine Unterstützung bietet, wird auf die Texteingabe zurückgegriffen.

| **Eingabetyp** | **Verwendungszweck** | **Hinweise** |
| --- | --- | --- |
| tel | Zur Eingabe einer Telefonnummer | tel setzt keine bestimmte Syntax voraus, wenn also ein bestimmtes Format eingehalten werden soll, können Sie zusätzlich pattern oder setCustomValidity() verwenden. |
| search | Fordert Nutzer zur Eingabe von Text auf, nach dem gesucht werden soll | Der Unterschied zwischen search und text ist in erster Linie stilistischer Natur. Durch Verwendung des Eingabetyps search wird der Stil des Eingabefelds möglicherweise an die Suchfelder dieser Plattform angeglichen. |
| url | Zur Eingabe einer URL | url ist zur Eingabe einer einzelnen, [absoluten URL](http://www.w3.org/TR/html5/urls.html#absolute-url) bestimmt, die einen sehr großen Wertebereich abdeckt. |
| email | Zur Eingabe einer E-Mail-Adresse oder einer Liste von E-Mail-Adressen | Bei Verwendung des Attributs multiple können mehrere, durch Komma getrennte E-Mail-Adressen eingegeben werden. |
| datetime | Zur Eingabe von Datum und Uhrzeit auf der Grundlage der Zeitzone UTC |  |
| date | Zur Eingabe eines Datums ohne Zeitzone |  |
| month | Zur Eingabe eines Datums mit Jahr und Monat, jedoch ohne Zeitzone |  |
| week | Zur Eingabe eines Datums bestehend aus einer Jahres- und einer Wochenzahl, jedoch ohne Zeitzone | Ein Beispiel für dieses Format ist 2011-W05 für die fünfte Woche des Jahres 2011. |
| time | Zur Eingabe einer Uhrzeit mit Stunde, Minute, Sekunden und Bruchsekunden, jedoch ohne Zeitzone |  |
| datetime-local | Zur Eingabe von Datum und Uhrzeit ohne Zeitzone |  |
| number | Für numerische Eingaben | Gültige Werte sind [Gleitkommazahlen](http://www.w3.org/TR/html5/common-microsyntaxes.html#valid-floating-point-number). |
| range | Für numerische Eingaben, bei denen, im Gegensatz zu number, der tatsächliche Wert nicht von Bedeutung ist | In den meisten Browsern, die diesen Eingabetyp unterstützen, erfolgt die Implementierung in Form eines Schiebereglers. |
| color | Zur Auswahl einer Farbe über ein entsprechendes Steuerungselement | Der Wert muss eine [gültige einfache Farbe in Kleinbuchstaben](http://www.w3.org/TR/html5/common-microsyntaxes.html#valid-lowercase-simple-color) sein, zum Beispiel #ffffff. |

## Neue Eingabeattribute

HTML5 bietet zudem mehrere neue Attribute für die Eingabe- und Formularelemente.

| **Attribute** | **Verwendungszweck** | **Hinweise** |
| --- | --- | --- |
| autofocus | Richtet den Fokus der Eingabe auf das Element, sobald die Seite geladen wurde | autofocus kann auf Eingabe, Auswahl, Textbereich oder Schaltfläche angewendet werden. |
| placeholder | Gibt dem Nutzer einen Hinweis darauf, welche Art von Daten eingegeben werden sollen | Der Wert für "placeholder" wird als heller Text dargestellt, bis das Element fokussiert wird und der Nutzer Daten eingibt. Mögliche Anwendungsgebiete: Eingabe und Textbereich. |
| form | Gibt ein oder mehrere Formulare an, auf das bzw. die sich das Eingabeelement bezieht | Bei Verwendung des Attributs form können die Eingabeelemente überall auf der Seite platziert werden, nicht nur innerhalb des Formularelements. Darüber hinaus kann ein einzelnes Eingabeelement mit mehreren Formularen verknüpft werden. |
| required | Ein boolesches Attribut, das angibt, dass das Element erforderlich ist | Das Attribut required ist hilfreich, wenn Sie eine browserbasierte Validierung ohne Verwendung von benutzerdefiniertem JavaScript durchführen möchten. |
| autocomplete | Legt fest, dass ein Feld nicht auf der Grundlage der vorherigen Eingaben des Nutzers vom Browser automatisch vervollständigt oder vorausgefüllt werden soll | Das Attribut autocomplete eignet sich für Felder, in die zum Beispiel eine Kreditkartennummer oder eine Einmalpasswort eingegeben werden und die nicht automatisch vervollständigt werden sollen. Standardmäßig ist für das Attribut autocomplete der Wert on festgelegt, Sie müssen also off auswählen, um es zu deaktivieren. |
| pattern | Gleicht den Wert eines Elements mit einem regulären Ausdruck ab | Bei Verwendung von pattern sollten Sie auch einen Wert für title eingeben, damit der Nutzer weiß, welches Muster erwartet wird. |
| dirname | Sendet die Ausrichtung des Steuerungselements zusammen mit dem Formular | Wenn der Nutzer zum Beispiel Textdaten von rechts nach links eingegeben hat und das Eingabeelement das Attribut dirname enthält, wird zusammen mit dem Eingabewert ein entsprechender Hinweis gesendet. |
| novalidate | Verhindert eine Validierung der Formulareingaben, sofern dies für ein Formularelement angegeben ist |  |
| formaction | Überschreibt das für das Formularelement angegebene Attribut "action" | Dieses Attribut wird für die Elemente input und button unterstützt. |
| formenctype | Überschreibt das für das Formularelement angegebene Attribut "enctype" | Dieses Attribut wird für die Elemente input und button unterstützt. |
|  |  |  |
|  |  |  |
|  |  |  |

## Browserbasierte Validierung

Seien wir ehrlich. Das Validieren von Formulardaten ist eine ziemlich langweilige Aufgabe, dennoch muss es getan werden. Für eine clientseitige Formularvalidierung schreiben Sie heute wahrscheinlich einen benutzerdefinierten JavaScript-Code oder verwenden eine Bibliothek, um vor dem Absenden des Formulars zu überprüfen, ob die eingegebenen Werte gültig sind oder alle erforderlichen Felder ausgefüllt wurden.

Neue Eingabeattribute wie required und pattern in Kombination mit CSS-Pseudoklassen-Selektoren vereinfachen das Schreiben von Prüfmechanismen und das Anzeigen von Feedback für den Nutzer. Auch andere erweiterte Validierungstechniken ermöglichen es Ihnen, unter Verwendung von JavaScript benutzerdefinierte Validierungsregeln und -nachrichten festzulegen oder herauszufinden, ob und warum ein Element ungültig ist.

### Das Attribut "required"

Bei Verwendung des Attributs required muss das entsprechende Feld einen Wert enthalten, wenn das Formular gesendet wird. Im Folgenden finden Sie ein Beispiel eines Eingabefelds für eine erforderliche E-Mail-Adresse. Damit wird sichergestellt, dass das Feld einen Wert aufweist und dass es sich bei dem Wert gemäß [dieser](http://dev.w3.org/html5/spec/Overview.html#valid-e-mail-address) Definition um eine gültige E-Mail-Adresse handelt.

<input type="email" id="email\_addr" name="email\_addr" required />

### Das Attribut "pattern"

Das Attribut pattern legt einen regulären Ausdruck fest, der zur Validierung eines Eingabefelds verwendet wird. Dieses Beispiel zeigt ein erforderliches Texteingabefeld für eine Teilenummer. Nehmen wir für dieses Beispiel an, dass eine Teilenummer aus drei Großbuchstaben gefolgt von vier Zahlen besteht. Durch die Verwendung von required und pattern wird sichergestellt, dass das Feld einen Wert enthält und dieser das richtige Format einer Teilenummer aufweist. Bewegt der Nutzer seine Maus über das Feld, erscheint die Nachricht im Attribut "title".

<input type="text" id="part" name="part" required pattern="[A-Z]{3}[0-9]{4}"

title="Part numbers consist of 3 uppercase letters followed by 4 digits."/>

Um noch einmal auf das vorherige Beispiel zurückzukommen, können Sie das Eingabefeld auch rot umranden, wenn eine ungültige Teilenummer eingegeben wird. Fügen Sie hierzu dieses CSS hinzu, um das Eingabefeld im Falle eines ungültigen Werts rot zu umranden.

:invalid {

border: 2px solid #ff0000;

}

### Das Attribut "formnovalidate"

Das Attribut formnovalidate kann für die Elemente input oder button verwendet werden. In diesem Fall wird die Validierung beim Senden des Formulars deaktiviert. Im Folgenden finden Sie ein Beispiel, bei dem für das Senden eines Formulars über die Schaltfläche "Senden" eine gültige Eingabe erforderlich ist, beim Senden über die Schaltfläche "Speichern" hingegen nicht.

<input type="text" id="part" name="part" required pattern="[A-Z]{3}[0-9]{4}"

title="Part numbers consist of 3 uppercase letters followed by 4 digits."/>

<input type="submit" formnovalidate value="Save">

<input type="submit" value="Submit">

### Einschränkungsvalidierungs-API

Mit dem [Einschränkungsvalidierungs-API](http://dev.w3.org/html5/spec/association-of-controls-and-forms.html#the-constraint-validation-api) erhalten Sie leistungsstarke Tools für die benutzerdefinierte Validierung. Mithilfe des APIs können Sie zum Beispiel einen benutzerdefinierten Fehler festlegen, überprüfen, ob ein Element gültig ist, oder herausfinden, warum ein Element ungültig ist. Im Folgenden finden Sie ein Beispiel, bei dem ein benutzerdefinierter Fehler angezeigt wird, falls die Werte zweier Felder nicht übereinstimmen.

<label>Email:</label>

<input type="email" id="email\_addr" name="email\_addr">

<label>Repeat Email Address:</label>

<input type="email" id="email\_addr\_repeat" name="email\_addr\_repeat" oninput="check(this)">

<script>

function check(input) {

if (input.value != document.getElementById('email\_addr').value) {

input.setCustomValidity('The two email addresses must match.');

} else {

// input is valid -- reset the error message

input.setCustomValidity('');

}

}

</script>

## Gemeinsame Verwendung

Das folgende Beispiel zeigt ein Anfrageformular für eine Reservierung, welches verschiedene Eingabetypen, die Formularvalidierung sowie CSS-Selektoren und -Stile vereint.

HTML und JavaScript für das Formular sehen wir folgt aus:

<form oninput="total.value = (nights.valueAsNumber \* 99) +

((guests.valueAsNumber - 1) \* 10)">

<label>Full name:</label>

<input type="text" id="full\_name" name="full\_name" placeholder="Jane Doe" required>

<label>Email address:</label>

<input type="email" id="email\_addr" name="email\_addr" required>

<label>Repeat email address:</label>

<input type="email" id="email\_addr\_repeat" name="email\_addr\_repeat" required

oninput="check(this)">

<label>Arrival date:</label>

<input type="date" id="arrival\_dt" name="arrival\_dt" required>

<label>Number of nights (rooms are $99.00 per night):</label>

<input type="number" id="nights" name="nights" value="1" min="1" max="30" required>

<label>Number of guests (each additional guest adds $10.00 per night):</label>

<input type="number" id="guests" name="guests" value="1" min="1" max="4" required>

<label>Estimated total:</label>

$<output id="total" name="total">99</output>.00

<br><br>

<label>Promo code:</label>

<input type="text" id="promo" name="promo" pattern="[A-Za-z0-9]{6}"

title="Promo codes consist of 6 alphanumeric characters.">

<input type="submit" value="Request Reservation" />

</form>

<script>

function check(input) {

if (input.value != document.getElementById('email\_addr').value) {

input.setCustomValidity('The two email addresses must match.');

} else {

// input is valid -- reset the error message

input.setCustomValidity('');

}

}

</script>

Dies ist das CSS in Verbindung mit dem Formularcode:

:invalid {

border-color: #e88;

-webkit-box-shadow: 0 0 5px rgba(255, 0, 0, .8);

-moz-box-shadow: 0 0 5px rbba(255, 0, 0, .8);

-o-box-shadow: 0 0 5px rbba(255, 0, 0, .8);

-ms-box-shadow: 0 0 5px rbba(255, 0, 0, .8);

box-shadow:0 0 5px rgba(255, 0, 0, .8);

}

:required {

border-color: #88a;

-webkit-box-shadow: 0 0 5px rgba(0, 0, 255, .5);

-moz-box-shadow: 0 0 5px rgba(0, 0, 255, .5);

-o-box-shadow: 0 0 5px rgba(0, 0, 255, .5);

-ms-box-shadow: 0 0 5px rgba(0, 0, 255, .5);

box-shadow: 0 0 5px rgba(0, 0, 255, .5);

}

form {

width:300px;

margin: 20px auto;

}

input {

font-family: "Helvetica Neue", Helvetica, Arial, sans-serif;

border:1px solid #ccc;

font-size:20px;

width:300px;

min-height:30px;

display:block;

margin-bottom:15px;

margin-top:5px;

outline: none;

-webkit-border-radius:5px;

-moz-border-radius:5px;

-o-border-radius:5px;

-ms-border-radius:5px;

border-radius:5px;

}

input[type=submit] {

background:none;

padding:10px;

}

<http://www.wufoo.com/html5/>

# Professionelles Layout

Um beispielsweise Tabellenzeilen besser lesbar zu machen, musste früher mit javascript mühselig eine gestreifte Darstellung programmiert werden.

## Optimierte Darstellung von Tabelllen

Tabellenspalten und -zeilen abwechselnd zu gestalten hieß bislang immer, eigene Klassen für die jeweiligen Spalten oder Zeilen zu vergeben. Das ist insbesondere dann aufwändig, wenn eine Spalte oder Zeile nachträglich eingefügt wird. Seit CSS3 gibt es mit mit „nth-child“ und „nth-of-type“ zwei Pseudoklassen, mit denen ein solches Vorhaben einfach umzusetzen ist.

Das übernehmen nun neue CSS3 Selektoren.

### CSS3 und „nth-child“

Bei der CSS3-Pseudoklasse „nth-child“ lassen sich Styleangaben nur auf bestimmte Kind-Elemente anwenden. Nehmen wir als Beispiel eine Tabelle mit n Spalten und Zeilen:

<table>

<tr>

<td>Zeile1, Spalte 1</td>

<td>Zeile1, Spalte 2</td>

<td>Zeile1, Spalte 3 </td>

</tr>

<tr>

<td>Zeile2, Spalte 1</td>

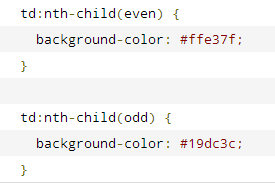
<td>Zeile2, Spalte 2</td>>

<td>Zeile2, Spalte 3</td>

</tr>

</table>

Mit der Pseudoklasse „nth-child“ können wir die Spalten und Zeilen unterschiedlich aussehen lassen. Die einfachste Möglichkeit ist, gerade und ungerade Spalten unterschiedlich aussehen zu lassen – mit den Schlüsselwörtern „even“ und „odd“:



Ausgangspunkt für die CSS3-Pseudoklasse „nth-child“ ist das jeweilige Elternelement – in diesem Fall das TR-Element. Allen geraden Kindelemente werden die Styles von „nth-child(even)“ zugewiesen und allen ungeraden Kind-Elementen die Styles von „nth-child(odd)“.

Neben der Verwendung von Schlüsselwörtern können auch Zahlen eingesetzt werden, um ein ganz bestimmtes Kind-Element anzusprechen:

td:nth-child(2) {

background-color: #ffe37f;

}

Bei der Angabe „nth-child(2)“ wirken sich die Styleangaben ausschließlich auf das zweite Kind-Element aus. Alternativ zu „nth-child“ existiert noch die Pseudoklasse „nth-last-child“. Sie macht dasselbe wie „nth-child“ mit dem Unterschied, dass hier vom letzten anstatt vom ersten Element gezählt wird. „nth-last-child(2)“ wirkt sich also auf das zweitletzte Element aus.

Als dritte Möglichkeit neben der Verwendung von Schlüsselwörtern und Zahlen für „nth-child“ lassen sich auch komplexere Abfolgen in der Form von „an+b“ verwenden. „a“ steht hierbei für eine Zahl, die den Intervall für die Wiederholung angibt. „b“ steht für das Element, auf das erstmalig die Styleangaben angewendet werden sollen:

td:nth-child(2n) {

background-color: #ffe37f;

}

Die Angabe „2n“ ist identisch mit „even“, da hier jedes zweite Element beziehungsweise alle geraden Elemente angesprochen werden. „2n+1“ ist identisch mit „odd“, da hier zwar auch jedes zweite Element angesprochen wird, die Zählung aber beim ersten Element beginnt. Mit „3n“ wird jedes dritte Element angesprochen. „3n+2“ spricht ebenfalls jedes dritte Element an; die Zählung beginnt allerdings beim zweiten Element.

### CSS3 und „nth-of-type“

Neben „nth-child“ gibt es noch die Pseudoklasse „nth-of-type“, die vom Grundprinzip identisch ist mit „nth-child“. Während „nth-child“ bei der Zählung alle Elemente berücksichtigt, zählt „nth-of-type“ nur Elemente des gleichen Typs mit:

<h1>Überschrift 1</h1>

<p>Absatz 1</p>

<p>Absatz 2</p>

<p>Absatz 3</p>

<h1>Überschrift 2</h1>

<p>Absatz 4</p>

<p>Absatz 5</p>

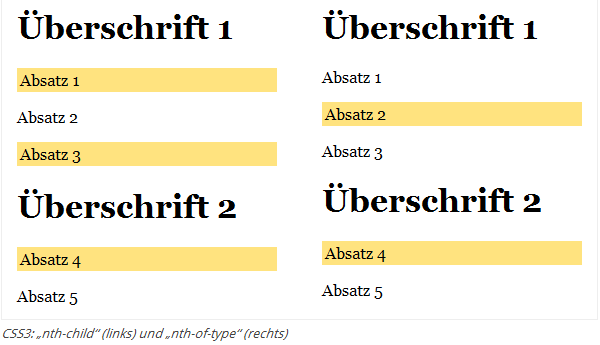
Wird „nth-child“ auf das P-Element angewendet, werden bei der Zählung auch die H1-Elemente berücksichtigt. Im Ergebnis heißt das, dass bei „p:nth-child(even)“ die Styleeigenschaften auf Absatz 1, 3 und 4 angewendet werden. Bei „p:nth-of-type(even)“ werden bei der Zählung nur P-Elemente berücksichtigt. Die Styleangaben wirken sich nur auf Absatz 2 und 4 aus:

p:nth-of-type(even) {

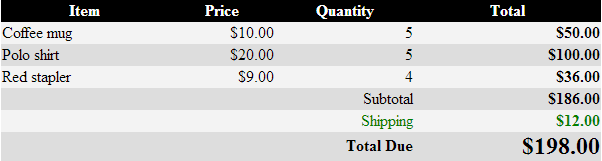
background-color: #ffe37f;

}

Auch für „nth-of-type“ gibt es die alternative Pseudoklasse „nth-last-of-type“, die vom letzten Kindelement beginnend zählt.



Übung: gewünschtes Aussehen:



Ausgangsdaten: Tabellen

## So werden Texte einfach mehrspaltig

Mehrspaltige Texte in HTML und CSS zu realisieren, ist bislang nur mit manuellem Aufwand und durch Aufteilung des Textes auf mehrere Bereiche möglich. Abhilfe schafft CSS3.

### Spaltenbreite mit CSS3 bestimmen

Bei der ersten Möglichkeit wird die Spaltenbreite festgelegt und die Anzahl der Spalten wird automatisch ermittelt – je nach Breite des gesamten Textblocks:

div {  
 width: 500px;  
 column-width: 200px;}

Die Breite des Textblocks entspricht in dem Beispiel 500 Pixel und die Spaltenbreite 200 Pixel. Das heißt, dass zwei Spalten Platz haben. Da die Gesamtbreite von 500 Pixeln eingehalten wird, werden die Spalten so angepasst, dass die Gesamtbreite immer ausgfüllt wird.

Das heißt, in dem Beispiel wären die Spalten etwas weniger als 250 Pixel breit, da noch einige Pixel Abstand zwischen den Spalten dazukommen. Ist die Gesamtbreite zu schmal für mehrere Spalten, wird der Text nur einspaltig dargestellt.

### Spaltenanzahl mit CSS3 bestimmen

Bei der zweiten Variante wird die Anzahl der Spalten vorgegeben und die Breite wird automatisch ermittelt:

div {  
 width: 500px;  
 column-count: 3;}

In dem Beispiel werden drei Spalten vorgegeben. Bei einer Gesamtbreite von 500 Pixeln, wäre jede Spalte etwas weniger als 166 Pixel breit.

Hat man eine feste Gesamtbreite für einen Text, bietet sich diese Variante an. Hat man eine flexible Gesamtbreite, bietet es sich an, die Spaltenbreite wie im ersten Ansatz festzulegen. So kann sichergestellt werden, dass die Spaltenbreite immer in etwa gleich breit ist – unabhängig davon, ob die Gesamtbreite sehr groß ausfällt (viele Spalten) oder sehr klein (nur eine Spalte).

### Abstand und Trennlinie festlegen

Normalerweise wird der Abstand zwischen den Spalten mit 1 em definiert. Allerdings lässt sich dieser Wert manuell einstellen:

div { width: 500px; column-count: 3; column-gap: 10px;}

Ebenso lässt sich eine Linie definieren, die zwischen den Spalten gezogen wird. Es lassen sich Linienbreite, -farbe und -stil festlegen – so wie man es von der CSS-Eigenschaft „border“ kennt:

div { width: 500px;  
 column-count: 3;  
 column-rule-color: red;  
  column-rule-width: 1px;  
  column-rule-style: dotted;}

### Umbrüche festlegen

Für die Feinjustierung von Spalten gibt es noch Möglichkeiten, Umbrüche festzulegen. Es kann zum Beispiel definiert werden, dass Überschriften immer in einer neuen Spalte stehen sollen:

h1 { column-break-before: column;}

Als Gegenstück lässt sich auch festlegen, dass Umbrüche erst nach einem bestimmten Elemenet erfolgen sollen. So lässt sich zum Beispiel definieren, dass es nach Bildern immer in einer neuen Spalte weitergehen soll:

img { column-break-after: column;}

Zu guter letzt gibt es noch die Möglichkeit, bestimmte Elemente immer über die gesamte Textbreite darstellen zu lassen, für Überschriften zum Beispiel manchmal sinnvoll:

h1 { column-span: all;}

Auch für Bilder bietet sich diese Möglichkeit an, da Bilder eine feste Breite haben und sich unterschiedlichen Spaltenbreiten nicht anpassen.



### Unterstützung der CSS3-Eigenschaften durch aktuelle Browser

Viele Möglichkeiten von [CSS3](http://t3n.de/news/css3-generatoren-vergleich-332000/) stehen in aktuellen Browsern bereits zur Verfügung. Auch die Mehrspaltigkeit wird von aktuellen Versionen von Firefox, Safari und Chrome bereits unterstützt. Allerdings ist derzeit noch der Einsatz des jeweiligen Vender-Präfixes notwendig.

Das heißt, für den Firefox muss „-moz“ und für Safari „-webkit“ vor jede Eigenschaft vorangestellt werden:

div {  
 -moz-column-width: 200px;  
 -webkit-column-width: 200px; }

Die verschiedennen Umbruchmöglichkeiten werden derzeit nur von Chrome und Safari unterstützt.

Übung Awosome newsletter



## CSS: DIV zentrieren

Im folgendem Abschnitt möchte ich zeigen, wie man mit CSS einen DIV-Layer permanent horizontal sowie auch vertikal in der Bildschirmmitte zentriert ausrichtet, um so z.B. eine Webseite im Browserfenster zentrieren zu können.

Hierzu benötigt man zuerst einmal den **XTHML-Code** des DIV-Layers mit der ID "page", in welchem ich Beispielhalber einmal einen kleinen Text hinterlegt habe:

<div id="page">  
Lorem ipsum dolor sit amet, consectetuer adipiscing elit.  
</div>

Nun erstellen wir im Stylesheet einen neuen CSS-Stil für den DIV-Layer "page" und definieren die Grundinformationen wie Hintergrundfarbe, Schrift und Rahmen:

#page {  
background: #CCC;  
border: 1px solid #666;  
font-family: Verdana, Arial, Helvetica, sans-serif;  
font-size: 1.2em;  
color: #333;  
}

|  |
| --- |
| Schritt 1 Zuerst definieren wir die Positionierung, Höhe und Breite unseres DIV-Layers:  #page { background: #CCC; border: 1px solid #666; font-family: Verdana, Arial, Helvetica, sans-serif; font-size: 1.2em; color: #333; position: absolute; width: 200px; height: 150px; }  Da sich der DIV-Layer nicht an anderen DIV´s ausrichten soll, definieren wir die Postition mit "postition: absolute;". Für Breite und Höhe nehme ich einmal 200px und 150px. |
| Schritt 2 Der DIV-Layer pappt nun am oberen linken Rand, da noch keinerlei Ausrichtungsangaben gemacht wurden. Damit der DIV-Layer nun permanent in der Mitte des Browserfensters positioniert ist, geben wir den Abstand zum oberen und zum linken Bildschirmrand mit "top: 50%;" und "left: 50%;" an:  #page { background: #CCC; border: 1px solid #666; font-family: Verdana, Arial, Helvetica, sans-serif; font-size: 1.2em; color: #333; position: absolute; width: 200px; height: 150px; top: 50%; left: 50%; } |
| Nun rutscht der DIV-Layer jedoch nicht in die Mitte sondern von der Mitte aus nach unten rechts weg, daher müssen wir nun den Mittelpunkt des DIV-Layers definieren. |
| Schritt 3 Unser DIV-Layer hat eine Breite von 200px und eine Höhe von 150px und von diesen beiden Werten nehmen wir nun je die Hälfte als Wert für den negativen Einzug von Oben und nach Links**:**  #page { background: #CCC; border: 1px solid #666; font-family: Verdana, Arial, Helvetica, sans-serif; font-size: 1.2em; color: #333; position: absolute; width: 200px; height: 150px; top: 50%; left: 50%; margin-top: -75px; margin-left: -100px; }  Nun ist der DIV-Layer über CSS ordnungsgemäß definiert und ausgerichtet. |

Wichtiger Link: [http://www.html-seminar.de](http://www.html-seminar.de/)

## CSS3-Buttons und Boxen ohne Grafiken erstellen

CSS3 macht in vielen Bereichen Bilder einfach unnötig. Für viele einfache Dinge muss man Photoshop gar nicht mehr öffnen. In diesem Abschnitt will ich zeigen, wie auch komplexe Buttons ohne Photoshop oder Grafiken nur mit CSS erstellt werden können.

### Ein schlichter Button ohne viel Schnick-Schnack

Als Basis für den Button dient folgender HTML-Code:

<span class="button"> Absenden</span>

Wir werden die Klasse „button“ benutzen, um den Button zu stylen. Eine Anmerkung noch: Natürlich kann man auch Links in Buttons umwandeln, indem man ihnen die Klasse „button“ zuweist. Dann sollte man aber auch darauf achten, die Schriftfarbe für die Pseudoklassen hover, active, link und visited zu definieren.

Bisher sieht man noch nichts von dem Button außer den puren Text. Die Puristen unter euch können also hier aufhören, euer Button ist fertig. ;-) Für alle anderen werden wir jetzt anfangen, dem Button ein neues Gewand zu geben.

#### Schritt 1: Schriftart

Als erstes definieren wir hier eine Schriftart, in diesem Fall Tahoma. Außerdem geben wir dem Button ein Padding von 10 Pixeln. Wichtig ist auch, dass wir mit der Eigenschaft „cursor“ definieren, welcher Mauszeiger beim „Hovern“ über dem Button gezeigt werden soll. Der „Pointer“ zeigt dem Nutzer, dass hier geklickt werden soll:

.button{font-family: Tahoma;  
 cursor: pointer;  
 padding: 10px;}

#### Schritt 2: Rahmen

Jetzt definieren wir einen Rahmen für den Button. Neu in CSS3: die Eigenschaft [boder-radius](http://www.css4you.de/moz-border-radius.html). Mit ihr kann man die Ecken abrufen. Mithilfe der Präfixe „-moz-“ und „-webkit-“ definieren wir die Eigenschaft auch in Firefox und Webkit-basierten Browsern. Leider wird diese Eigenschaft noch nicht von allen Browsern unterstützt.

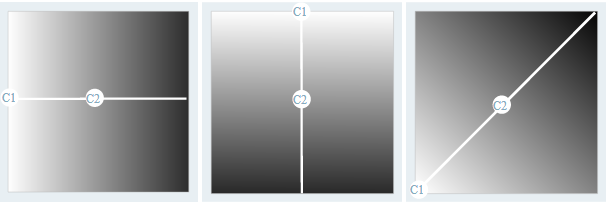
border: 1px solid #cacaca;-moz-border-radius: 10px; -webkit-border-radius: 10px; border-radius: 10px;

Hier haben wir also schon die erste Grafik gespart, da wir für runde Ecken keine Grafiken mehr brauchen.

#### Schritt 3: Hintergrund

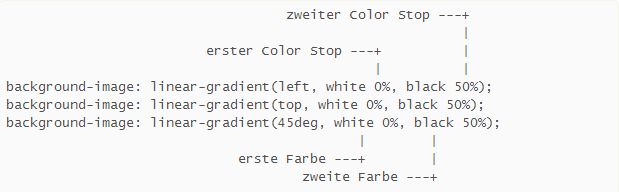
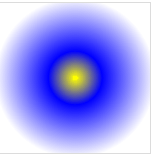
Als Nächstes wollen wir dem Button einen Hintergrund geben. Bisher musste man für Farbverläufe immer eine Grafik anlegen, aber mit CSS3 können wir hier noch eine Grafik sparen.

Ein Gradient oder Verlauf ist eine gedachte Linie, auf der Farbunterbrechungen (Color Stops) liegen. Im einfachsten Fall gibt es zwei Farbunterbrechungen in einem Verlauf. An jedem Color Stop beginnt eine Farbe, die bis zum nächsten Color Stop in die andere Farbe verläuft.



An welcher Stelle der Verlauf beginnt, wird durch die %-Angabe jedes Color Stops festgelegt. Für einen sauberen Verlauf müssen die Color Stops also von links nach rechts immer höhere Werte aufweisen.

Zwischen linear-gradient und der öffnenden Klammer darf kein Leerzeichen oder Zeilenumbruch stehen, sonst interpretiert der Browser den CSS-Stil nicht.



Sowohl Webkit-basierte Browser als auch Firefox unterstützen auch einen radialen (kreisförmigen) Verlauf – Internet Explorer bis einschl. Version 9 nicht.

background: radial-gradient(center,

circle farthest-corner,

yellow 0%,

blue 25%,

white 75%);

Quelle: <http://www.mediaevent.de/css/gradient.html>

Ich generiere meine Farbverläufe mit dem [CSS3 Gradient Generator von Damian Galarza](http://gradients.glrzad.com/). Im Grunde ist die Bedienung ganz einfach. Es müssen nur die Farben, die Position der Farben und die Richtung des Verlaufes festgelegt werden.

background: #dedede;  
 background-image: -webkit-gradient(  
 linear,  
 left bottom,  
 left top,  
 color-stop(0.09, rgb(199,199,199)),  
 color-stop(0.89, rgb(222,222,222)) );  
 background-image: -moz-linear-gradient(  
 center bottom,  
 rgb(199,199,199) 9%,  
 rgb(222,222,222) 89% );

Hier ist also der Code, den ich generiert habe. Noch ein kleiner Tipp: Wenn ihr noch keine Idee habt, welche Farben ihr benutzen sollt, klickt einfach im Gradient Generator auf „Generate Random Gradient“. Manchmal liefert der Generator einem damit echt interessante Ideen.

Ich habe zusätzlich vor der Definition der Verläufe eine normale Hintergrundfarbe definiert. Falls ein Browser keine Verläufe unterstützt, wird diese Farbe verwendet.

### Schritt 4: Hover-Style

Aber etwas fehlt noch: der Hover-Style. Der Nutzer soll doch wissen, dass er sich gerade über einem Button befindet.

.button:hover

{

border: 1px solid #afafaf;

}

Beim Hover soll also einfach nur der Rahmen etwas dunkler werden.

Wir haben jetzt also einen Button mit CSS3 erstellt. Aber natürlich will ich noch mehr neue Eigenschaften vorstellen, deshalb folgen jetzt noch einige Beispielbuttons.

#### Tricksen mit Schlagschatten

.button{  
 cursor: pointer;  
 font-family: Tahoma;  
 color: #fafafa;  
 padding: 12px 15px;  
 -moz-border-radius: 15px;  
 -webkit-border-radius: 15px; border-radius: 15px;  
 background: #96642b;  
 background-image: -webkit-gradient(  
 linear,  
 left bottom,  
 left top,  
 color-stop(0.23, rgb(115,71,17)),  
 color-stop(0.39, rgb(150,100,43)) );  
 background-image: -moz-linear-gradient(  
 center bottom,  
 rgb(115,71,17) 23%,  
 rgb(150,100,43) 39% );}

Auch Schatten musste man bisher mit [Photoshop](http://t3n.de/tag/photoshop) erstellen, aber [CSS3](http://t3n.de/tag/css3) bietet auch hierfür eine Eigenschaft. Mit [box-shadow](http://www.braekling.de/web-design/3274-css3-box-shadow-schatten.html) können leicht ein oder mehrere Schlagschatten definiert werden.

box-shadow: inset 3px 3px 6px rgba(255, 255, 255, 0.7), inset -3px -3px 6px rgba(0, 0, 0, 0.5);

Hier definiere ich also gleich zwei Schlagschatten. Mit dem Parameter „inset“ wirken die Schatten nicht nach außen, sondern nach innen, ähnlich wie „Inner Shadows“ in Photoshop. Außerdem verwende ich für die Farbwerte die in CSS3 neue Form rgba(). Dafür gibt man einfach nacheinander die Zahlenwerte für Rot, Grün, Blau und Alphatransparenz an. Wer auf die Transparenz verzichten kann, kann auch rgb() benutzen und den Alphawert weglassen.

Weiterhin ist zu beachten, dass wieder mit -moz- und -webkit- gearbeitet werden muss.

Generator box shadow: <http://www.webmaster-elite.de/online-tools/boxshadow-generator.html>

#### Schatten für die Schrift

Mit CSS kann man selbst Text einen oder mehrere Schatten verleihen. Das ist zwar nicht neu in CSS3, aber trotzdem toll. Die Syntax ist genau wie bei box-shadow, nur dass die Eigenschaft text-shadow heißt und hier keine Präfixe verwendet werden müssen.

.button{  
 cursor: pointer;  
 font-family: "ArmWrestlerBold";  
 color: #4a301e;  
 font-size: 1.1em;  
 letter-spacing: 1px;  
 text-shadow: 1px 1px 1px rgba(255, 255, 255, 0.5), -1px -1px 1px rgba(0, 0, 0, 1);  
 padding: 1em;  
 -moz-border-radius: 15px; -webkit-border-radius: 15px; border-radius: 15px;  
 background: #96642b;  
 background-image: -webkit-gradient(  
 linear,  
 left bottom,  
 left top,  
 color-stop(0.23, rgb(115,71,17)),  
 color-stop(0.39, rgb(150,100,43)) );  
 background-image: -moz-linear-gradient(  
 center bottom,  
 rgb(115,71,17) 23%,  
 rgb(150,100,43) 39% );  
 -moz-box-shadow: inset 3px 3px 6px rgba(255, 255, 255, 0.7), inset -3px -3px 6px rgba(0, 0, 0, 0.5);  
 -webkit-box-shadow: inset 3px 3px 6px rgba(255, 255, 255, 0.7), inset -3px -3px 6px rgba(0, 0, 0, 0.5);  
 box-shadow: inset 3px 3px 6px rgba(255, 255, 255, 0.7), inset -3px -3px 6px rgba(0, 0, 0, 0.5);}

Wie man sehen kann, habe ich auch hier wieder zwei Schatten definiert, sodass der Text wie eingestanzt wirkt.



#### Weitere Experimente

Ich habe noch ein wenig mit CSS rumgespielt, die Ergebnisse will ich euch natürlich nicht vorenthalten. Ihr findet hier die erstellten Buttons und den CSS-Code:

css3buttons6

.button{  
 cursor: pointer;  
 font-family: "ArmWrestlerBold";  
 font-size: 1.5em;  
 color: #196a96;  
 letter-spacing: 1px;  
 text-shadow: -1px -1px 1px rgba(0, 0, 0, 1), 1px 1px 1px rgba(255, 255, 255, 1);  
 padding: 10px 15px;  
 -moz-border-radius: 8px; -webkit-border-radius: 8px; border-radius: 8px;  
 background: rgb(14,147,208);  
 background-image: -webkit-gradient(  
 linear,  
 left bottom,  
 left top,  
 color-stop(0.17, rgb(14,147,208)),  
 color-stop(0.59, rgb(41,177,250)) );  
 background-image: -moz-linear-gradient(  
 center bottom,  
 rgb(14,147,208) 17%,  
 rgb(41,177,250) 59% );  
 -moz-box-shadow: 1px 1px 0px rgba(0, 0, 0, 1), -1px -1px 0px rgba(255, 255, 255, 1);  
 -webkit-box-shadow: 1px 1px 0px rgba(0, 0, 0, 1), -1px -1px 0px rgba(255, 255, 255, 1);  
 box-shadow: 1px 1px 0px rgba(0, 0, 0, 1), -1px -1px 0px rgba(255, 255, 255, 1);}

css3buttons7

.button{  
 cursor: pointer;  
 font-family: Verdana;  
 font-weight: bold;  
 color: #212121;  
 text-shadow: 1px 1px 1px rgba(255, 255, 255, 1);  
 padding: 10px 15px;  
 border: 2px solid #ffffff;  
 -moz-border-radius: 8px; -webkit-border-radius: 8px; border-radius: 8px;  
 background: rgb(224,224,224);  
 background-image: -webkit-gradient(  
 linear,  
 left bottom,  
 left top,  
 color-stop(0.49, rgb(224,224,224)),  
 color-stop(0.5, rgb(237,237,237)) );  
 background-image: -moz-linear-gradient(  
 center bottom,  
 rgb(224,224,224) 49%,  
 rgb(237,237,237) 50% );  
 -moz-box-shadow: 0px 0px 3px rgba(0, 0, 0, 1);  
 -webkit-box-shadow: 0px 0px 3px rgba(0, 0, 0, 1);  
 box-shadow: 0px 0px 3px rgba(0, 0, 0, 1);}

Super Linke für CSS3 Generator: [http://www.css3-generator.de](http://www.css3-generator.de/) und [http://css3generator.de](http://css3generator.de/)

## Animationen mit Transitions

Eine Transition ist als die fließende Veränderung einer CSS-Eigenschaft auf einer Zeitachse zu verstehen. Wo sich bislang bei einem Mouseover eine Farbe wie auf Knopfdruck änderte, kann sich diese über eine Transition langsam in die andere Farbe verwandeln.

Das Transitions-Modul des CSS-3-Standards ist mit fünf Eigenschaften überschaubar. Im Gegensatz zu Animationen und Transformationen ist der Einsatz einfach, die einzelnen Eigenschaften intuitiv und sicher der leichteste Einstieg in den Bereich der animierten Inhalte.

Pflichtparameter sind transition-property und transition-duration. über transition-property geben Webworker die CSS-Eigenschaft an, die sie mit einer Transition versehen wollen. übertransition-duration definiert er die Zeit, die eine fließende Veränderung dauern soll. Eine Minimalangabe sieht beispielsweise so aus (weiter unten findet ihr die Schreibweisen für die einzelnen Browser. Alle Eigenschaften sind erst mal in der Standardschreibweise des W3C gezeigt):

#div1 {

position: absolute;

width:200px;

height:100px;

left:50%;

top:50%;

margin-top:-50px;

margin-left:-100px;

background-color: red;

transition: background-color 1.5s;

}

#div1:hover {

background-color: blue;

}

Für transition-duration als auch für transition-delay werden Zeitwerte erwartet, entweder in Sekunden oder Millisekunden mit den Abkürzungen für die Einheiten **s** oder **ms.**

Die wahrscheinlich umfangreichste Eigenschaft ist transition-timing-function. Darüber wird das Geschwindigkeitsverhalten bestimmt, mit der sich die Transition verhält. Für die »Zeitfunktion« stehen folgende Werte zur Verfügung:

**Schlüsselworte (Keywords)**

* **ease** (Standard): Die Transition beginnt verlangsamt, nimmt schnell Geschwindigkeit auf und endet langsam. Zwei Drittel der Transition ist nach der halben Zeit dargestellt.
* ease-in: Die Transition beginnt langsam, endet schnell.
* ease-out: Die Transition startet schnell, endet langsam.
* ease-in-out: Die Transition beginnt und endet langsam, in der Mitte ist die Geschwindigkeit am höchsten.
* linear: Gleichbleibende Geschwindigkeit, keine Verlangsamung am Anfang oder Ende der Transition.

**steps(nr, [start|end]) benötigt immer zwei Parameter:**

* Als erster wird die Anzahl der Schritte definiert, in denen die Transition angezeigt werden soll.
* Der zweite Parameter hat die möglichen Werte start oderend. Damit wird bestimmt, ob ein Schritt vor oder nach einem Zeitintervall dargestellt werden soll. Was kompliziert klingt, ist anhand eines Beispiels leichter zu erklären: Ihr habt einen Lichtschalter, den ihr in fünf Sekunden anschalten sollt. Allerdings nur am Anfang oder Ende der fünf Sekunden. step (1, start) bedeutet, ihr schaltest den Schalter sofort an und es ist sofort hell. Oder ihr wartet fünf Sekunden und schaltest das Licht dann an, das wäre step(1,end). Beides Mal ist es nach fünf Sekunden hell, nur das »ab wann« wird durch den zweiten Parameter in steps() bestimmt.
* step-start ist ein Schlüsselwort und entspricht steps (1,start)
* step-end ist ebenfalls ein »Keyword« und entspricht steps (1,end). Beide Schlüsselworte sind also fest definiert auf einen Schritt in der Transition.

Damit lassen sich zahlreiche Animationen gestalten. Wie z.B.:

<style>

#div1 {

position: absolute;

width:200px;

height:100px;

left:50%;

top:50%;

text-align:center;

padding-top:40px;

color:#FFFFFF;

font-size:24px;

margin-top:-50px;

margin-left:-100px;

background-color: red;

transition-property: width, background-color;

transition-duration: 5s, 5s;

transition-timing-function: steps(10,start), steps(5,end);

}

#div1:hover {

width: 600px;

background-color: blue;

}

</style>

## Transform

**Jedes Element kann mit einem gewünschten Winkel rotiert angezeigt werden. Dadurch werden spannende Effekte möglich. Zumal in der realen Welt auch die wenigsten Dinge gerade sind. Rotiert werden können Schrift, Bilder und jedes beliebige Element über diese CSS-Anweisung.**

In CSS3 gibt es eine große Gruppe an Befehlen um Elemente zu transformieren, sprich im Aussehen zu verändern. Die englische Übersetzung von transform ist: „verwandeln, überführen, umwandeln“.

Die erste Anweisung, die wir uns ansehen möchten ist zum Rotieren von einem Element.

Die CSS3-Anweisung hat folgendes Aussehen:

transform: rotate(30deg);

Nach dem allgemeinen CSS3-Befehl transform wird die Art der „Umwandlung“ angegeben: hier also rotieren und zwar um 30 Grad.

Dabei stehen positive Gradzahlen für Drehung im Uhrzeigersinn.

Negative Gradangaben stehen für eine Drehung gegen den Uhrzeigersinn.

Falls gerade der Gedanke kommt, warum überhaupt negative Angaben, denn wenn man lange genug im Uhrzeigersinnetwas dreht, kann man dasselbe Aussehen bekommen, als würde man es gegen den Uhrzeigersinn drehen. Also ob ich etwas um 300 Grad drehe oder um -60 Grad, sollte ja nichts am Endergebnis ändern. Allerdings haben wir später auch noch in CSS3 Befehle, um die Drehung sichtbar zu machen. Daher ist es doch wichtig.

Auch hier haben viele Browserhersteller vorgegriffen und das rotieren von Elemente sehr früh eingeführt. Daher gibt es wieder die typischen Anweisungen, damit es auch mit alten Versionen der Browser funktioniert.

-moz-transform: rotate(30deg);

-ms-transform: rotate(30deg);

-o-transform: rotate(30deg);

-webkit-transform: rotate(30deg);

Wichtig ist, dass nach den speziellen für alte Browserversionen gedachtem CSS-Befehle der allgemein als letzte kommt. Dadurch führt dann der Browser immer den zuletzt erkannten Befehl aus und der allgemein definierte ist besser als irgendwas, was auch leicht in der Umsetzung abweichen kann. Daher bitte immer folgenden Aufbau: vom Speziellen zum Allgemeinen:

-moz-transform: rotate(30deg);

-ms-transform: rotate(30deg);

-o-transform: rotate(30deg);

-webkit-transform: rotate(30deg);

**Beliebtester Fehler bei transform: rotate**

Was schnell ins Auge gehen kann und man dann sich unter Umständen wundert, warum die Rotation nicht ausgeführt wird, ist ein versehentliches Leerzeichen zwischen rotate und der Klammer.

FALSCH: rotate (30deg);  
RICHTIG: rotate(30deg);

Aufgabe zu transform: rotate

Um diesen CSS3-Befehl rotate zu testen, bitte folgende Aufgabe nachbauen. Wir haben hier 4 Boxen, beschriftet mit 1 bis 4 und unterschiedlichen Farben, die rotieren. Es soll folgendes Aussehen entstehen:

**Aufgabe: über CSS3-Befehl zum Rotieren folgendes Aussehen von 4 Zahlen erstellen**

Zum Testen des eigenen Ergebnis immer einen aktuellen Browser verwenden (empfehlenswert ist der Google Browser Chrome).

Quelle::  
<http://www.html-seminar.de/beispiel-css3-transform-rotate.htm>

**Lösung:**

Starten wir mit einer quasi leeren HTML5-Datei

<!DOCTYPE html>

<html>

<head>

<title>CSS3 transform: rotate</title>

<meta charset="UTF-8" />

<link href="css3-transform-rotate.css"

type="text/css" rel="stylesheet" />

</head>

<body>

</body>

</html>

Wir benötigen nun 4 Bereiche für unseren Inhalt 1,2,3 und 4. Also erstellen wir 4 DIV-Bereiche, die wir über die ID ansprechen können mit dem gewünschten Inhalt.

<div id="box1">1</div>

<div id="box2">2</div>

<div id="box3">3</div>

<div id="box4">4</div>

Nun erstellen wir die CSS-Datei mit dem in der HTML-Datei gewählten Dateinamen „css3-transform-rotate.css“. Hier werden wir als erstes der box1 die Hintergrundfarbe orange mitgeben. Dadurch haben wir eine gute Kontrolle, ob die Verknüpfung zwischen der HTML-Datei und der CSS-Datei funktioniert (oder sich beim Dateinamen ein Tippfehler eingeschlichen hat).

#box1 {

background-color: orange;

}

Als Ergebnis haben wir nun die Zahl 1 orange hinterlegt, was sich über die komplette Breite erstreckt.

Um Tipparbeit (und Tippfehler) zu sparen, können wir uns nun überlegen, was alle 4 Boxen gemeinsam haben. Dazu gehört:

* die Breite der Box
* die Höhe der Box
* der schwarze Rahmen
* der Abstand von oben

Jetzt ist natürlich taktisch geschickt, eine CSS-Klasse mit diesen Eigenschaften zu erstellen. Klasse deshalb, da ja öfters vorkommt. Im Beispiel wird diese einfach .box genannt:

.box {

height: 140px;

width: 100px;

border: 1px solid black;

top: 100px;

}

Damit das wirkt, müssen wir unsere 4 DIVs in der HTML-Datei ergänzen um die Klassenangabe. Hier nun der vollständige Code für die HTML-Datei

<!DOCTYPE html>

<html>

<head>

<title>CSS3 transform: rotate</title>

<meta charset="UTF-8" />

<link href="css3-transform-rotate.css"

type="text/css" rel="stylesheet" />

</head>

<body>

<div id="box1" class="box">1</div>

<div id="box2" class="box">2</div>

<div id="box3" class="box">3</div>

<div id="box4" class="box">4</div>

</body>

</html>

Wenn wir uns nun das bisherige Ergebnis ansehen, sehen wir, dass die 4 Boxen untereinander abgebildet werden, der Abstand von oben über top: aber ignoriert wird.

**bisheriges Ergebnis: 4 Boxen untereinander**

Damit die Angaben wie top in CSS genutzt werden können, benötigen wir position: absolute

.box {

position: absolute;

top: 50px;

height: 140px;

width: 100px;

border: 1px solid black;

}

Das erzeugt uns jetzt erstmals ein sehr lustiges Aussehen!

**alle Boxen liegen übereinander und die Zahlen überdecken sich**

Durch die absolute Positionierung liegen die Boxen nun übereinander und die Zahlen überdecken sich. Daher müssen die einzelnen Boxen über die entsprechenden ID-Selektoren platziert werden und bekommen ihre Farben zugewiesen.

.box {

position: absolute;

top: 50px;

height: 140px;

width: 100px;

border: 1px solid black;

}

#box1 {

background-color: orange;

left: 50px;

}

#box2 {

background-color: #e82d0c;

left: 200px;

}

#box3 {

background-color: #cc0aff;

left: 350px;

}

#box4 {

background-color: #c2e80c;

left: 500px;

}

Dadurch haben wir folgendes Aussehen erreicht.

**farbige Boxen sauber ausgerichtet**

Nun wird das Aussehen der Schrift geändert und diese groß und mittig ausgerichtet. Praktischerweise können wir die Schrift über unsere allgemeine Klasse box gestalten.

Was machen wir mit der Schrift:

* Schriftfarbe wird weiß   
  color: white;
* zentrieren diese horizontal   
  text-align: center;
* zentrieren diese vertikal, da die Box eine Höhe von 140 Pixel hat, ergibt sich   
  line-height: 140px;
* Geben der Schrift noch die Schriftfamilie, die Stärke und die Größe mit

Als kompletter CSS-Code sieht das wie folgt aus:

.box {

position: absolute;

top: 50px;

height: 140px;

width: 100px;

border: 1px solid black;

color: white;

text-align: center;

line-height: 140px;

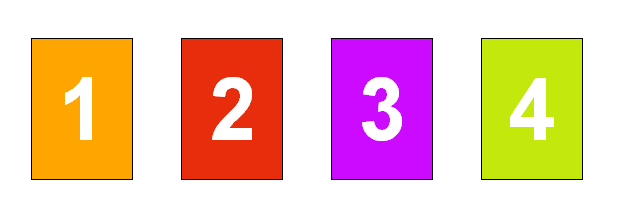
font-family: sans-serif;

font-size: 65pt;

font-weight: bold;

}

Und dadurch haben wir sauber die Zahlen in den Boxen:

**Boxen mit Schrift in groß und zentriert**

**Rotieren der Boxen**

Was jetzt noch fehlt ist das rotieren der Boxen. Hier kommt jetzt der CSS3-Befehl transform: rotate zum Einsatz.

Da die Drehung für jede Box unterschiedlich ist, muss dies für jede ID gemacht werden. Bitte auch sofort für die alten Browserversionen die Angaben machen!

#box1 {

background-color: orange;

left: 50px;

-moz-transform: rotate(15deg);

-ms-transform: rotate(15deg);

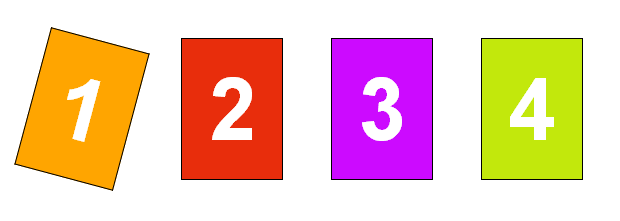
-o-transform: rotate(15deg);

-webkit-transform: rotate(15deg);

transform: rotate(15deg);

}

Somit wird nun die erste Box um 15 Grad im Uhrzeigersinn gedreht.

**Drehung um 15 Grad der ersten Box**

Und nun für die weiteren 3 Boxen die Drehungen. Box 2 um 70 Grad, Box 3 um 150 Grad und Box 4 um 180 Grad.

Hier der komplette CSS-Code für dieses Beispiel:

.box {

position: absolute;

top: 50px;

height: 140px;

width: 100px;

border: 1px solid black;

color: white;

text-align: center;

line-height: 140px;

font-family: sans-serif;

font-size: 65pt;

font-weight: bold;

}

#box1 {

background-color: orange;

left: 50px;

-moz-transform: rotate(15deg);

-ms-transform: rotate(15deg);

-o-transform: rotate(15deg);

-webkit-transform: rotate(15deg);

transform: rotate(15deg);

}

#box2 {

background-color: #e82d0c;

left: 200px;

-moz-transform: rotate(70deg);

-ms-transform: rotate(70deg);

-o-transform: rotate(70deg);

-webkit-transform: rotate(70deg);

transform: rotate(70deg);

}

#box3 {

background-color: #cc0aff;

left: 350px;

-moz-transform: rotate(150deg);

-ms-transform: rotate(150deg);

-o-transform: rotate(150deg);

-webkit-transform: rotate(150deg);

transform: rotate(150deg);

}

#box4 {

background-color: #c2e80c;

left: 500px;

-moz-transform: rotate(180deg);

-ms-transform: rotate(180deg);

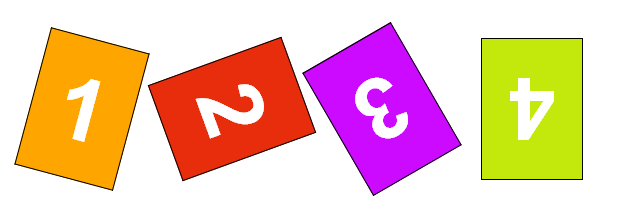
-o-transform: rotate(180deg);

-webkit-transform: rotate(180deg);

transform: rotate(180deg);

}

Und das sollte bei aktuellen Browsern angezeigt werden.

**fertige Lösung zu transform: rotate**

Über den Befehl transform: rotate sind schöne Effekt für das Design bei Websites möglich. Einfach mal die Steuerung bei <http://www.umzug-mexiko.info/> ansehen. Hier werden die einzelnen Bereiche der Steuerung leicht gedreht um aus dem geraden, statischen herauszukommen.

Weitere Möglichkeiten kann man am besten auf <http://www.css3-generator.de/transform.html> erforschen.



# Zeichnen mit Canvas

Das englische Wort Canvas bedeutet auf Deutsch „Leinwand“ und ist ziemlich genau das: Eine Leinwand, auf die man mit Javascript dynamisch Bitmap-Grafiken malen kann. Erfunden wurde das Element vom Webkit-Team und später sind auch Mozilla und Opera auf den <canvas>-Zug aufgesprungen. Damit ist das das Element heutzutage in alle anständigen Browser implementiert und Teil des werdenden HTML5-Standards.

## Wie bindet man das <canvas>-Element ein?

Sowohl ein öffnender als auch ein schließender Tag werden für <canvas> gebraucht – zwischen den Tags kann man Fallback-Content notieren, also Inhalt, der ausgegeben wird wenn das<canvas>-Element selbst nicht angezeigt werden kann. Breite und Höhe sollten als Attribute angegeben werden – wenn man die Maße per CSS festlegt, wird der <canvas>-Inhalt ausgehend von der Ausgangsgröße entsprechend verzerrt.

**<canvas** width="100" height="100"**>**

Fallback-Inhalt

**</canvas>**

Eine ID zu vergeben ist meist auch sinnvoll – man will ja das Element irgendwann mit Javascript beackern können.

Ein <canvas>-Element stellt einen oder mehrere sogenannte *rendering contexts* zur Verfügung, quasi Ebenen auf denen man zeichnen kann. Der im Moment einzige Context ist der 2D-Context – an einem 3D-Context wird aber gearbeitet. Für den Moment vergessen wir also einfach all das und beschränken uns auf den 2D-Context. Um mit <canvas> erste Schritte zu machen, brauchen wir nur zwei Dinge: ein <canvas>-Element in einem HTML-Dokument und eine Javascript-Funktion, die auf dem <canvas>-Element herummalt. Das Canvas-Element könnte so aussehen:

**<canvas** id="testcanvas1" width="300" height="200"**>**

Dein Browser kann diese Grafik nicht darstellen.

**</canvas>**

Es enthält Maße, Fallback-Content und eine ID zur Steuerung. Wenn wir jetzt dieses Rechteck mit einer Farbe füllen wollten, bräuchten wir nur 8 Zeilen Javascript:

**function** drawCanvas(){

**var** canvas = document.getElementById('testcanvas1');

**if**(canvas.getContext){

**var** context = canvas.getContext('2d');

context.fillStyle = "rgb(255, 0, 255)";

context.fillRect(0, 0, canvas.width, canvas.height);

}

}

Zur Erläuterung:

* var canvas = document... holt das <canvas>-Element anhand seiner ID
* var context = canvas.getContext('2d') holt den 2D-Context in die handlichecontext-Variable. Das brauchen wir, weil wir letzlich nicht auf dem <canvas>-Element selbst, sondern auf einem Context des Elements zeichnen.
* context.fillStyle = "rgb(255, 0, 255)" setzt die Füllfarbe für alles, was wir im Context hiernach zeichnen werden auf ein schönes, kräftiges Pink
* context.fillRect(0, 0, canvas.width, canvas.height) zeichnet ein gefülltes Rechteck, beginnend bei den Koordinaten 0,0 und endend bei der unteren rechten Ecke des<canvas>-Elements, definiert durch die gesamte Höhe und Breite
* Jetzt braucht man nur noch die Funktion drawCanvas() irgendwie auszuführen (z.B. mittels<body onload="drawCanvas()">) und schon hat man [ein fertiges, funktionierendes Canvas-Beispiel](http://www.peterkroener.de/test/canvas/).

## Was kann man genau alles mit <canvas> anstellen?

Natürlich kann man mit <canvas> einiges mehr anstellen als nur pinke Flächen malen. Die komplette API ist in den [offiziellen HTML5-Spezifikationen](http://www.whatwg.org/specs/web-apps/current-work/multipage/the-canvas-element.html) sehr erschöpfend behandelt. Für eine etwas bekömmlichere Darstellung, die auf zwei A4-Seiten passt und obendrein noch bebildert ist, sei [dieser Cheat Sheet](http://blog.nihilogic.dk/2009/02/html5-canvas-cheat-sheet.html) aufs allerwärmste empfohlen.

Kurz gesagt kann man Pfade zeichnen, vorhandene Bilder manipulieren, Text scheiben und mehr. Die eingangs erwähnten Beispiele wie Bespin und MochaUI machen deutlich, dass man mit Canvas sehr viel anstellen kann. Man muss es eben nur wollen. Drei weitere (einfache) Beispiele gefällig?

### Canvas-Beispiel 1: Ein Bild drehen

Man kann in ein <canvas>-Element Bilder oder auch den Inhalt anderer <canvas>-Elemente zeichnen. [Das könnte zum Beispiel so aussehen.](http://www.peterkroener.de/test/canvas/bild.html) Im Prinzip ist das das gleiche Spiel wie beim Einführungs-Beispiel:

**function drawCanvas(){**

**var canvas = document.getElementById('testcanvas1');**

**var img = new** Image**();**

**img.onload = function(){**

**if(canvas.getContext){**

**var context = canvas.getContext('2d');**

**context.translate(200, 0);**

**context.rotate(90 \*** Math**.PI/180);**

**context.drawImage(img, 0, 0, 200, 200);**

**}**

**}**

**img.src = 'test.png';**

**}**

* var img = new Image() erstellt ein neues img-Element, das als Referenz für unser canvas-Element dient.
* context.translate(200, 0) positioniert den Punkt, um den der Context rotieren wird
* context.rotate(90 \* Math.PI/180) macht eine 90°-Drehung
* context.drawImage(img, 0, 0) zeichnet das Bild img ausgehend von den Koordinaten 0,0.
* Weil in diesem Beispiel der gesamte Context und nicht das Bild rotiert wird, kann man die Funktion drawCanvas() immer wieder aufrufen um das Ganze in 90°-Schritten immer weiter zu drehen.

### Beispiel 2: Eine kleine Animation

Animieren ist einfach: man zeichnet einfach Frame für Frame das <canvas>-Element neu. So kann man zum Beispiel [ein paar halbtransparente Quadrate durch die Gegend segeln lassen.](http://www.peterkroener.de/test/canvas/animation.html) Man nehme einfach wieder eine drawCanvas()-Funktion …

**function drawCanvas(){**

**var canvas = document.getElementById('testcanvas1');**

**if(canvas.getContext){**

**var context = canvas.getContext('2d');**

**animate(context, 31, 69, 1, -1, 8, 25, 1, 1);**

**}**

**}**

…  und eine mit einem Timer versehene animate()-Funktion, die nichts weiter macht, als anhand eines Haufens von Koordinaten Frame für Frame alle neu zu zeichnen.

**function animate(context, ax, ay, adx, ady, bx, by, bdx, bdy){**

**setTimeout(function(){**

**//Bewegen und Abprallen für Quadrat A**

**if(ax == 150){**

**adx = -1;**

**}**

**else{**

**if(ax == 0){**

**adx = 1;**

**}**

**}**

**if(ay == 150){**

**ady = -1;**

**}**

**else{**

**if(ay == 0){**

**ady = 1;**

**}**

**}**

**ax = ax+adx;**

**ay = ay+ady;**

**//Bewegen und Abprallen für Quadrat B**

**if(bx == 150){**

**bdx = -1;**

**}**

**else{**

**if(bx == 0){**

**bdx = 1;**

**}**

**}**

**if(by == 150){**

**bdy = -1;**

**}**

**else{**

**if(by == 0){**

**bdy = 1;**

**}**

**}**

**bx = bx+bdx;**

**by = by+bdy;**

**// Alles neu zeichnen**

**context.clearRect(0, 0, 200, 200);**

**context.fillStyle = "rgba(0, 0, 255, 0.5)";**

**context.fillRect(ax, ay, 50, 50);**

**context.fillStyle = "rgba(0, 255, 0, 0.5)";**

**context.fillRect(bx, by, 50, 50);**

**self.animate(context, ax, ay, adx, ady, bx, by, bdx, bdy);**

**}, 0);**

**}**

Einfach bei jedem Frame mit context.clearRect(0, 0, 200, 200) die Canvas putzen und die Quadrate an ihren neuen Koordinaten zeichnen.

### Beispiel 3: Paint

Richtig Interessant wird <canvas> erst, wenn man den Benutzer richtig mitmachen lässt. So ist zum Beispiel [ein (sehr sehr rudimentäres) Malprogramm](http://www.peterkroener.de/test/canvas/paint.html) recht einfach machbar. Man nehme bei Mausbewegung die Koordinaten …

**var x, y;**

**canvas.onmousemove = function(e){**

**x = e.clientX-canvas.offsetLeft;**

**y = e.clientY-canvas.offsetTop;**

**kx.innerHTML = x;**

**ky.innerHTML = y;**

**paint();**

**}**

…  und male, falls die Maustaste gedrückt ist, einfach an der Mausposition eine Fläche:

**var active = false;**

**canvas.onmousedown = function(){ active = true; }**

**canvas.onmouseup = function(){ active = false; }**

**function paint(){**

**if(active){**

**context.fillRect(x, y, 10, 10);**

**}**

**}**

Quelle: http://www.peterkroener.de/eine-kleine-canvas-einfuehrung/

## Statistiken grafisch darstellen

Eine der wichtigsten Funktionen in der Wirtschaft ist wohl die grafische Darstellung von Statistiken. Die Daten kommen mittels Serverscript oder Ajax und werden am Client aufbereitet und dargestellt.

<!DOCTYPE html>

<html>

<head>

<title>Graph</title>

<script src="javascripts/RGraph.common.js" ></script>

<script src="javascripts/RGraph.bar.js" ></script>

</head>

<body>

<canvas width="500" height="500" id="test">[no canvas support]</canvas>

<script type="text/javascript" charset="utf-8">

var bar = new RGraph.Bar('test', [100,25]);

bar.Set('chart.gutter', 100);

bar.Set('chart.colors', ['red']);

bar.Set('chart.title', "Anteil der Prozente");

bar.Set('chart.labels', ["Maximal", "Anteil"]);

bar.Draw();

</script>

</body>

</html>

# Video und Audio einbetten

Um ein Video zu integrieren musste man es früher erst auf Youtube hochladen, um es dann in die Website einzubetten. Hatte den Nachteil, dass die User (Schüler/innen) verleitet werden, sich weitere Videos auf Youtube anzusehen.

Für Audios musste man selbst einen eigenen Flashcode programmieren. Das hat sich nun in HTML5 wesentlich vereinfacht!

Der HTML 5 Befehl <video> benötigt auf jeden Fall noch die Angabe zum Video. Diese Angabe wird zwischen dem Anfangs-Tag und dem End-TAG eingebettet in Form von

<video>

<source src="video.ogg" type="video/ogg" />

</video>

Für das Format des Videos haben sich 2 verschiedene Formate herauskristallisiert. Einerseits das Format Ogg Theora (OGG) und auf der anderen Seite MP4 (h.264).

Warum 2 verschiedene Formate? Das liegt an Patente und rechtliche Einschränkungen, die mit dem Format H.264 (MP4) zusammenhängen – dagegen punktet das Ogg Theora (OGG) Format als Open Source. In dem HTML-Befehl <video> können beide Dateien eingebunden werden:

<video>

<source src="video.ogg" type="video/ogg" />

<source src="video.mp4" type="video/mp4" />

</video>

Damit die Sache nicht zu einfach wird, die verschiedene Browser unterstützen je nach Hersteller nur das eine oder andere Format (genau aus diesen lizenzrechtlichen Gründen).

| **Browser** | **Ogg Theora (.ogg)** | **h.264 (.mp4)** |
| --- | --- | --- |
| Firefox | Ja |  |
| Internetexplorer |  |  |
| Opera | Ja |  |
| Chrome | Ja | Ja |
| Safari |  | Ja |
| iPhone |  | Ja |
| Android |  | Ja |

Stand: Februar 2010

Was bedeutet das für den Einsatz von Videos auf Webseiten. Will man so viele Browser wie möglich unterstützen, muss das Video in beiden Formaten vorgehalten werden. Das hat den Nachteil, dass man annähernd doppelte so viel Speicherplatz benötigt und 2 Videodateien erstellen bzw. pflegen muss. Weiterhin hat man auch als Privatperson ein rechtliches Thema – auch wenn der Zeitpunkt für Lizenzkosten erst dieses Jahr auf das Jahr 2015 verschoben wurde, auch dieses Jahr kommt einmal.

Daher werden hier im Folgenden nur mit dem Format .ogg gearbeitet – die Beispieldatei mit der Maus können Sie herunterladen und im Firefox damit den <video>-TAG testen. Firefox ist zum Webseiten entwickeln eh die bessere Variante.

Die Integration des Abspielens im Browser hat den riesigen Vorteil, dass keinerlei Plug-ins installiert werden müssen, es schneller i.d.R. lädt und man keine Bedenken haben muss, dass ein Plug-in den Browser „crashed“.

## Attribute des HTML-TAGs <video>

Dem HTML-TAG <video> können die Attribute width, height, src, poster, autobuffer, autoplay und controls mitgegeben werden.

Im folgenden Beispiel sieht man den Einsatz

<video src="video.ogg" width="320" height="240"

poster="video-standbild.jpg" autobuffer autoplay

controls>

<div>Schade – hier käme ein Video, wenn Ihr

Browser HTML5 Unterstützung hätte, wie z.B. der

aktuelle Firefox</div>

</video>

Bedeutung der einzelnen Attribute:

**width** – die angezeigt Breite der Videos

**height** – die angezeigt Höhe des Videos

**poster** – ein Bild, das angezeigt wird, wenn das Video noch nicht gestartet wurde. Das Attribut poster wird von nicht allen Browsern unterstützt.

**autoplay** – das Video wird sofort gestartet, wenn ein Besucher die HTML-Seite aufruft

**controls** – es werden für die Steuerung Steuerelemente (controls) angezeigt, dass das Video gestoppt, gestartet und an eine andere Stelle gesprungen werden kann.

**autobuffer** - Zwischenspeicherung aktivieren, damit Video möglichst ruckelfrei wiedergegeben werden kann.

Eine wirklich ideale Seite um Formate zu konvertieren ist: [http://video.online-convert.com/^](http://video.online-convert.com/%5e)

## Audio einbetten

Bei Audios ist es ähnlich einfach.

Es gibt 3 Formate für das Audio-Element:

| **Format** | **IE 9** | **Firefox 3.5** | **Opera 10.5** | **Chrome 3.0** | **Safari 3.0** |
| --- | --- | --- | --- | --- | --- |
| Ogg Vorbis | Nein | Ja | Ja | Ja | Nein |
| MP3 | Ja | Nein | Nein | Ja | Ja |
| Wav | Nein | Ja | Ja | Ja | Ja |

Beispiel in HTML5

<audio controls="controls">

<source src="musik.ogg" type="audio/ogg" />

<source src="musik.mp3" type="audio/mpeg" />

Sorry - Ihre Browser hat keine Unterstützung für dieses Audio-Format.

</audio>

Anmerkung: Folgende Parameter sind möglich

|  |  |
| --- | --- |
| autoplay | Startet sofort beim Laden der Seite |
| loop | Spielt das Audio immer wieder ab |
| controls | Blendet die Steuerung ein |
| preload | Ladet im voraus (für größere Audios sinnvoll) |

# Sonstige Neuerungen in HTML5

## iFrames

Neben embed und object stellt das hier beschriebene iframe-Element die dritte Möglichkeit dar, beliebige andere Ressourcen in HTML einzubetten. Bei diesem Element besteht der Haupt-Einsatzzweck darin, vollständige anderen Webseiten einzubinden. Das kann beispielsweise eine auf jeder Seite benötigte Navigation sein, oder ein separat vom übrigen Inhalt dynamisch erzeugter Inhalt, etwa ein Newsticker.

Eingebettete Frames werden seit langem von allen Browsern interpretiert und gehören seit Version 4.0 zum HTML-Standard. In HTML 4.0 wurden sie allerdings in eine Schublade mit den mittlerweile verpönten und nicht mehr im Standard enthaltenen Framesets gesteckt und nur in der nicht wirklich erwünschten Frameset-Variante von HTML 4.0 mitgeführt. Mit HTML5 gehört das iframe-Element endlich zum offiziellen Sprachstandard. Dabei wurden auch einige neue Attribute und damit verbundene Konzepte eingeführt, die allerdings teilweise nicht ganz unumstritten sind und sich eher an zukünftige Browser richten.

Teil 1: Dokument mit dem iframe-Element

<!DOCTYPE html>

<html>

<head>

<meta charset="utf-8">

<title>Eingebettete Frames</title>

</head>

<body>

<header>

<h1>HOME</h1>

</header>

<div style="width: 250px; float: left;">

<nav>

<iframe src="navigation.html" width="220"

height="350" style="border: 0;">

<a href="navigation.html">Navigation anzeigen</a>

</iframe>

</nav>

</div>

<div style="float: left;">

<p>Viel Inhalt</p>

</div>

</body>

</html>

Teil 2: Eingebettetes Dokument

<!DOCTYPE html>

<html>

<head>

<meta charset="utf-8">

<base target="\_parent">

<title>Navigation</title>

</head>

<body>

<ul>

<li><a href="home.html">HOME</a></li>

<li><a href="produkte.html">Produkte</a></li>

<li><a href="referenzen.html">Referenzen</a></li>

<li><a href="downloads.html">Downloads</a></li>

<li><a href="rueckschau.html">Rückschau</a></li>

<li><a href="vorschau.html">Vorschau</a></li>

<li><a href="kontakt.html">Kontakt</a></li>

<li><a href="impressum.html">Impressum</a></li>

</ul>

</body>

</html>

Das Zusammenspiel der beiden HTML Dokumente kann mit einigen Parametern geregelt werden. Weitere Informationen findet ihr bei:

<http://webkompetenz.wikidot.com/html-handbuch:eingebettete-frames>

## Sonstige Erweiterungen HTML5

### WebStorage und Datenbanken

HTML5 bringt neue Speichertechniken als Alternative zu den aufwändigen Verfahren wie Cookies.

* *WebStorage*: LocalStorage und SessionStorage
* *Datenbanken im Browser des Benutzers*: Indexed DB und Web SQL

HTML-Seiten haben kein Gedächtnis: Sie sind zustandslos. WebStorage verleiht Webseiten ein Gedächtnis über mehrere Seiten und bietet dafür wesentlich mehr Platz im Browser des Benutzers als Cookies.

Web Storage und Datenbanken sind im Browser integriert und speichern, ändern und löschen Daten, auch wenn die Anwendung nicht online durchgeführt wird.

Das garantiert eine schnellere Verarbeitung und erleichtert die Programmierung von Anwendungen im Vergleich zu Cookies.

* Web Storage und Datenbanken werden nicht wie Cookies auf den Server übertragen, sondern auf dem System des Benutzers gespeichert.
* Während die Nutzlast des Cookies auf 4 KB beschränkt ist, können Web Storage und Datenbanken 5 bis 10 MB zur Verfügung stellen. Das reduziert die Last auf den Internetverbindungen.
* WebStorage und Datenbanken speichern Daten sicher beim Benutzer und übertragen sie nicht wie Cookies unverschlüsselt über das Internet.

Weitere Informationen findet ihr bei:

<http://www.mediaevent.de/javascript/web-storage.html>

### WebSocket

Ermöglichen eine dauerhafte Verbindung zwischen den Clients und dem Webserver. Die bekannteste Anwendung dafür ist ein Chat.

Setzt allerdings recht gute Programmierkenntnisse in Javascript voraus. Für Interessierte eine sehr gute Einführung:

<http://www.html5rocks.com/de/tutorials/websockets/basics/>

# Einführung in Jquery

## Das Framework

Das von [John Resig](http://ejohn.org/) entwickelte JavaScript-Framework [jQuery](http://jquery.com/) zählt mittlerweile zu den beliebtesten Bibliotheken und ist zudem sehr einfach zu benutzen. Für alle, die den Einstieg in die wunderbare Welt von jQuery planen, habe ich eine Einführung in die Grundladen des Frameworks verfasst. Allem voran soll die Namensgebung "AJAX-Framework" geklärt sein, da unter Anderem bei Google die Rede von [Google AJAX Libraries](http://code.google.com/intl/de-DE/apis/ajaxlibs/) ist: Ajax (Asynchronous JavaScript and XML) bezeichnet den Vorgang einer asynchronen Datenübertragung zwischen Server und Browser, die durch das XMLHttpRequest-Objekt (XHR) bewerkstelligt wird. Ein JavaScript-Framework wie jQuery bietet hervorragende Möglichkeiten für Ajax-Anwendungen, kann aber auch für reines DOM-Scripting, also das Manipulieren von HTML-Elementen, verwendet werden.



Das Projekt glänzt vor allem durch seine tolle Dokumentation und so gibt es im [Tutorials-Bereich](http://docs.jquery.com/Tutorials) der Website schon viele Anleitungen (allerdings in Englisch). Die eigentliche [Dokumentation](http://docs.jquery.com/Main_Page) ist bestens organisiert und sollte bei der jQuery-Entwicklung stets geöffnet sein. Neben der Referenz finden sich in der Dokumentation auch einfache Beispiele inklusive Quelltext.  
Dieses Tutorial richtet sich an Webdesigner und Entwickler, die mit HTML/CSS und JavaScript vertraut sind. Auch wenn sich die Programmierung in jQuery simpel gestaltet, sollte man Kenntnisse über die DOM-Struktur und die allgemeine JavaScript-Syntax besitzen. Ein Besuch bei [SelfHTML](http://de.selfhtml.org/javascript/index.htm) lohnt sich.

Um mit der Arbeit beginnen zu können, wird die jQuery-Bibliothek benötigt: Bei [Google](http://code.google.com/intl/de-DE/apis/ajaxlibs/) wird die aktuelle Version der Bibliothek gehostet und lässt sich somit direkt in Projekten einsetzen. Natürlich kann die Datei auch direkt von der [Website](http://jquery.com/) heruntergeladen werden. Für dieses Tutorial empfehle ich die Benutzung von [JS Bin](http://jsbin.com/) — damit lässt sich HTML-, CSS- und JavaScript-Code direkt im Browser testen, ohne dass dafür Dateien erstellt werden müssen. JS Bin bietet das Einbetten diverser JavaScript-Frameworks per Knopfdruck (Dropdown-Menü "Include Library").  
In jedem Fall muss also die Hauptbibliothek von jQuery im <head>-Bereich der Website eingebettet werden (JS Bin erledigt dies automatisch). Das HTML-Konstrukt sieht dann in etwa so aus (Charset-Encoding fehlt):

<!DOCTYPE html">

<html>

<head>

<title>jQuery Beispiel</title>

<script src="jquery.min.js" type="text/javascript"></script>

</head>

<body>

</body>

</html>

## Selektoren und Funktionen

In diesem Tutorial werden hauptsächlich die Syntax und die Selektoren von jQuery behandelt. Ich werde lediglich mit Inline-Styles und -Scriptcode arbeiten — beides lässt sich natürlich als externe Datei einbetten.  
Wir erstellen daher noch folgende Elemente (HTML/CSS):

<head>

<title>jQuery Beispiel</title>

<script src="jquery.min.js" type="text/javascript"></script>

<script type="text/javascript">

// jQuery-Code

</script>

<style type="text/css">

div#test {

width:100px;

height:100px;

background:#FF0000;

}

</style>

</head>

<body>

<div id="test">Ein Beispiel</div>

</body>

Das Konstrukt enthält jetzt eine rote Box mit der ID "test", jedoch noch keine JavaScript-Aktionen. Diese werden später dem noch leeren <script>-Bereich hinzugefügt. Die Platzierung dieses JavaScript-Bereichs (oder der externen Datei) innerhalb des HTML-Konstrukts ist für den zeitlichen Aufruf des Scripts verantwortlich — wird ein Script per <script> am Ende des HTML-Codes (zum Beispiel vor </body>) eingebettet, so wird es erst ausgeführt, nachdem die HTML-Datei geladen wurde. Eine bessere Möglichkeit bietet jedoch jQuery mit einer "Bereit"-Funktionalität, die ich gleich beschreiben werde.  
In jQuery ist das Dollarzeichen ($) ein zentrales Element, da es als Selektor für DOM-Elemente eingesetzt wird. Ein wichtiger Code-Baustein ist also Folgender:

$("element")

Hieraus ergeben sich verschiedene Punkte, die wichtig bei der Verwendung von Selektoren sind:

* Anführungszeichen müssen beachtet werden — die Elemente window, document, this sowie Funktionsparameter (Argumente) benötigen keine Anführungszeichen
* neben CSS-Selektoren (ID, Klasse, Child, etc.) ist auch die Angabe von Standard-Elementen (div, p, a, usw.) möglich
* innerhalb der Anführungszeichen können komplexe Selektoren (mehrere Elemente, Child-Elemente, uvm.) angegeben werden

In der Klammer können also neben einfachen Ausdrücken auch erweiterte Filter verwendet werden, wie man in den späteren Tutorials noch sehen wird. Die am häufigsten angewandten Selektoren dürften Folgende sein:

$("#id")

$(".klasse")

$("div, p, a, span, li")

$("p"); *// Alle p-Tags*

$(".meineListe"); *// Elemente der Klasse meineListe*

$("#des"); *// Das Element mit der ID "desc"*

$("ul p"); *// p-Tags innerhalb einer Liste*

Attribute und spezifische Selektoren werden im nächsten Tutorial behandelt. Nach Möglichkeit sollte man stets IDs ("#id") verwenden, da diese Elemente schneller angesprochen werden als Klassen oder Standard-Elemente — das macht sich allerdings nur in sehr großen HTML-Strukturen bemerkbar.  
Wie eben schon erwähnt, kann mit jQuery definiert werden, wann JavaScript-Code ausgeführt wird. Der logischste Zeitpunkt dafür tritt ein, sobald die HTML-Datei und alle darin befindlichen Elemente (wie Bilder) geladen wurden: Das Dokument ist dann "fertig". So heißt auch die entscheidende Funktion in jQuery, die das Script zum Rollen bringt, wenn der HTML-Code vollständig übertragen wurde:

$(document).ready(function(){

// jQuery-Code

});

Dieser Aufruf sollte den kompletten jQuery-Code umschließen und die Basis für jedes jQuery-Projekt sein. Bevor wir den Code für unser kleines Beispiel finalisieren, müssen noch ein paar Dinge zur Syntax geklärt werden. Also zurück zum Dollarzeichen: Nur mit der Selektor-Funktion lässt sich noch nicht viel anstellen, da die Anweisungen in Form von DOM-Operationen fehlen. Diese folgen unmittelbar mit einem Punkt (.) auf den Selektor:

$("#test").css("color", "#FFFFFF");

Mit dieser Zeile würde also die Schriftfarbe des div-Containers mit der ID "test" auf "weiß" gesetzt werden. Die Terminologie von jQuery hat auch bei dieser Funktionsweise ein paar Besonderheiten:

* die Werte (oder Funktionsparameter) sollten und müssen zum größten Teil in Anführungszeichen angegeben werden (ein Ausnahme bilden zum Beispiel einfache Integer-Zahlen)
* jede DOM-Manipulation hat eine indiviuelle Anzahl von Argumenten
* bei dem ".css()"-Manipulator (und auch bei ".animate()") können neben CSS-Eigenschaften auch JavaScript-typische [Style-Eigenschaften](http://de.selfhtml.org/javascript/objekte/style.htm) verwendet werden

Bei dem Verändern und Animieren von bestehenden CSS-Einstellungen sollen oft mehrere Parameter übergeben werden. Im obigen Beispiel wird lediglich einer CSS-Eigenschaft ("color") ein Wert ("#FFFFFF") zugewiesen. Soll mehr als eine Eigenschaft verarbeitet werden, müssen geschweifte Klammern ({…}) in Verbindung mit einem Doppelpunkt (:) eingesetzt werden. Hier ein Beispiel:

$("#test").css({ "color": "#FFFFFF", "height": "25px" });

Hiermit wird also die Textfarbe wieder auf weiß eingestellt und die Höhe des div-Elements wird auf 25 Pixel gesetzt.  
Nun gibt es noch die Möglichkeit, mehrere DOM-Manipulationen auf einmal vorzunehmen — im Beispiel eben verändert lediglich eine DOM-Manipulation (".css()") mehrere Attribute. Möchte man nun also noch zusätzlich den Text des Elements verändern, sieht der Code so aus:

$("#test").css({ "color": "#FFFFFF", "height": "25px" }).html("Neu");

Der DOM-Manipulator ".html()" stellt das Pendant zur JavaScript-Eigenschaft "innerHTML" dar und verändert den Text eines DOM-Elements. In unserem Beispiel wird also der Text des Test-Elements in "Neu" geändert. Die DOM-Funktionen wurden einfach aneinander gehängt (Chaining): Sie werden in jQuery nacheinander ausgeführt. So sind beliebig große Verkettungen möglich:

$("element").funktion1().funktion2().funktion3().funktion4();

Zur besseren Ansicht können die verketteten Funktionen in separate Zeilen geschrieben werden. In Verbindung damit werden wir zum Abschluss unseren Beispiel-Code mit den eben genannten Funktionen komplettieren:

…

<head>

<title>jQuery Beispiel</title>

<script src="jquery.min.js" type="text/javascript"></script>

<script type="text/javascript">

$(document).ready(function(){

$("#test")

.css({ "color": "#FFFFFF", "height": "25px" })

.html("Neu");

});

</script>

<style type="text/css">

div#test {

width:100px;

height:100px;

background:#FF0000;

}

</style>

</head>

<body>

<div id="test">Ein Beispiel</div>

</body>

* …

<http://design-creators.net/tutorials/jquery-fur-einsteiger-teil-1-die-erste-animation/>

## Reagieren auf Events

jQuery bietet eine einfache und komfortable Möglichkeit mit Browser-Events, aber auch vom Programmierer selbst erstellten Events umzugehen. Ein Event ist dabei ein Ereignis, dass durch eine Aktion oder Vorgang ausgelöst wurde. Solch ein Ereignis kann sowohl durch den User eines Browsers, als auch den Browser oder den Code ausgelöst werden. Einige der bekanntesten und am häufigsten benötigten Events dabei sind:

* Click: Wird durch das Klicken des Users mittels Maus auf ein Element ausgelöst
* Hover: Wird dann ausgelöst, wenn der User mit dem Mauszeiger über ein Element fährt
* Keypress: Wenn ein User eine Taste der Tastatur betätigt
* Submit: Wenn beispielsweise ein Formular gesendet werden soll
* Ready: Wenn der DOM des Elements vollständig geladen wurde

Neben den Standard-JavaScript Events kann ein Programmierer jedoch auch selbstständig eigene Events auslösen. Dieser Vorgang wird ebenso wie bei Standard-Evens „triggern“ genannt. Soll eine bestimmte Aktion ausgeführt werden, wenn ein bestimmtes Event eintritt, so muss der Programmierer diesen Code an den jeweiligen Event binden. Dabei kann diese Bindung auch nur an bestimmte Events von bestimmten Elementen im HTML-Code erfolgen. jQuery unterstützt dabei zwei Arten von Event-Bindung.

Das folgende Codebeispiel stellt die erste Variante dar, die für Standard-Events eingesetzt werden kann. Dabei wird der Funktionsname des Events an ein jQuery Objekt gehängt, wie es im Beispiel mit dem ready-Event gezeigt wird. Im Selektor wird dabei das document-Objekt verwendet. Hierbei handelt es sich um das document-Objekt von JavaScript. Tritt nun das Event ready am document-Objekt auf, wird die zwischen den runden Klammern definierte Funktion aufgerufen. Die ready-Funktion erhält als Parameter entweder einen Funktionsnamen, der aufgerufen werden soll, oder eine anonyme Funktion die direkt ausgeführt wird. Im konkreten Beispiel wird mittels der JavaScript Alert-Funktion der Text „DOM wurde geladen!“ ausgegeben.

|  |
| --- |
| $(document).ready(**function**(){  *//Ausgabe einer Meldung*      alert(“DOM wurde geladen!”);  }); |

Die zweite Variante wird folgenden Codebeispiel dargestellt. Dabei wird die Funktion „bind“ verwendet, die als ersten Parameter den Namen des Events erwartet und als zweiten Parameter analog zur ersten Variante entweder einen Funktionsnamen oder eine anonyme Funktion.

Im Beispiel wird an alle p-Tags im HTML Code das Event „click“ gebunden. Klickt der User nun mit der Maus auf einen der Absätze (p-Tags), so wird das Event getriggert und mittels der JavaScript Alert-Funktion der Text „Listenelement wurde geklickt“ ausgegeben.

|  |
| --- |
| $("p").bind("click",**function**(){      alert("Listenelement wurde geklickt");  }); |

## [DOM-Manipulation](http://www.web-couch.com/2012/01/jquery-tutorial-teil-3-dom-manipulation/)

Mittels jQuery ist es möglich, das Dokument auch nach der Auslieferung zu ändern. Dies kann zum Beispiel der Fall sein, wenn nach einem Klick auf ein Bild weitere Informationen zu diesem Bild eingeblendet werden sollen oder um beispielsweise bei Formularen nach einem Klick auf einen Button eine weitere Eingabebox einzublenden.

Im folgenden Codebeispiel wird ein ähnlicher Code wie aus dem vorangegangenen Beispiel verwendet. Mittels des Selektors „#add“ wird das Element mit der ID add ausgewählt. Hierbei handelt es sich um den im folgenden Codebeispiel dargestellten HTML-Button. An dieses Element wird ein Click-Event gebunden. Sobald ein User auf diesen Button klickt, wird an alle Listen auf dieser HTML-Seite ein weiteres Listenelement mit dem Text „Weiterer Eintrag“ angehängt. Dieses Beispiel stellt natürlich nur einen der zahlreichen Anwendungsfälle und Möglichkeiten der DOM-Manipulation dar.

|  |
| --- |
| $("#add").bind("click", **function**(){      $("ul").append("&lt;li&gt;Weiterer Eintrag&lt;/i&gt;");  }); |

## Animationen

Jquery bietet eine Vielzahl von einfach zu programmierenden Animationen an.

Beispiel für animierte Boxen:

$(document).ready(function() {

var x =0;

$('#box1').click(function() {

if (x==0) { // x schaltet einfach um

$(this)

.animate( { width:'500px' }, 1000) // verändern der Breite in einer Sek.

.animate( {opacity:'0.3'}, 1500)

.animate( { height:'150px'}, {duration:1500, queue:false})

x=1;

}

else {

$(this)

.animate( { width:'250px' }, 1000)

.animate( {opacity:'0.9'}, 1500)

.animate( { height:'50px'}, {duration:1500, queue:false})

x=0;

}

});

$('#box2').click(function() {

$(this)

.animate( { width:'500px',

height:'150px' }, 1000);

});

$('#box3').click(function() {

$(this)

.animate( {opacity:'0',height:'0'},

1500,

'linear',

function(){

alert('Bin dann mal weg!');

}

);

});

$('#box4').click(function() {

$(this)

.animate(

{width:'+=20px'},

500

);

});

$('#box5').hover(

function() {

$(this).animate(

{width:'500px'},

500

)},

function() {

$(this).animate(

{width:'250px'},

500

)}

);

});

</script>

</head>

<body>

<div id="wrap">

<div id="box1">Dies ist Box 1.</div>

<div id="box2">Box 2 ist besser.</div>

<div id="box3">Box 3 ist noch besser!</div>

<div id="box4">Hier ist Box 4. Ich wachse noch.</div>

<div id="box5">Box 5 hovert.</div>

</div>

</body>

</html>

### Praktisches Beispiel: Slideshow

Die einfachste Slideshow ist mit wenigen Befehlen programmiert:

<!DOCTYPE html>

<html>

<head>

<title>Bildgalerie</title>

<script src="http://ajax.googleapis.com/ajax/libs/jquery/1/jquery.min.js"></script>

<script language="javascript" type="text/javascript">

var i =2;

$( init );

function init()

{

$("#bild").fadeOut(1000,function()

{

$("#bild").attr("src","Jordanien/" + i + ".jpg");

}

).fadeIn(1000);;

i++;

if (i>26) i=1;

setTimeout("init()",3000);

}

</script>

</head>

<body >

<p align="center"><img align="middle" id="bild" src="Jordanien/1.jpg" width="800" height="480" ></p>

</body>

</html>

In dem Ordner „Jordanien“ befinden sich 27 Fotos, die alle nummeriert sind. Die verkürzte Schreibweise $( init ); kommt dem $(document).ready(function() gleich. Nach dem Laden wird die Function „Init“ aufgerufen, die erst das aktuelle Bild ausblendet. Die Function für die Änderung der Bildquelle muss in dieser Form geschrieben werde, da sonst die Sekunde nicht gewartet wird, sondern der Ablauf sofort weiter geht. Das heißt, dass das Laden des nächsten Bildes erst erfolgt, wenn die Animation abgeschlossen ist.

Dann wird die neue Quelle eingeblendet und nach 3 Sekunden die Function „Init“ wieder aufgerufen. Das Programm läuft somit in einer Schleife.

Zu dieser – zugegeben sehr einfachen Slideshow – gibt es zahlreiche Modifikationen, ja sogar eine Jquery-plugins (siehe Beispiel Video auf DVD).

Wenn man zusätzlich eine Beschreibung einblenden möchte, würde sich eine Lösung über eine Array anbieten.

### Praktisches Beispiel „tooltipp“

In vielen Anwendungen sieht man sogenannte „Tooltipps“ wenn man mit der Maus über einen Begriff oder ein Bild fährt. Es erscheint ein Div mit einer Erklärung bzw. weiteren Informationen oder auch Bildern. Das ist sehr benutzerfreundlich, da kein eigenes Fenster geöffnet werden muss.

Ich habe fast alle meine Programme damit ausgestattet und erhalte sehr positive Rückmeldungen.

Beispiel Landkarte.htm

## Professionelle Jquery Anwendungen

### Puzzle

Wie bereits erwähnt, gibt es für Jquery zahlreiche Erweiterungen (plugins), wie z.B. für die Programmierung von Puzzles.

Ein einfaches Puzzle sei hier vorgestellt:

<http://www.script-tutorials.com/how-to-create-a-game-using-jquery/>

Der HTML Teil ist sehr klein:

<!DOCTYPE html>

<html lang="en" >

<head>

<meta charset="utf-8" />

<meta name="author" content="Script Tutorials" />

<title>How to create a game using jQuery | Script Tutorials</title>

<!-- add styles -->

<link href="css/main1.css" rel="stylesheet" type="text/css" />

<!-- add scripts -->

<script src="http://code.jquery.com/jquery-1.7.1.min.js"></script>

<script src="js/main1.js"></script>

</head>

<body>

<header>

<h2>Einfaches Puzzle</h2>

</header>

<!-- Hier wird das Puzzle integriert -->

<div id="game\_object"></div>

</body>

</html>

Der Skriptteil:

// Wenn das Dokument geladen wurde, gehts los

$(function() {

// Initialisieren und das Plugin erstellen

var zi = 1; // Erhöhen z-index wenn ein Teil ausgetauscht wird

// Der Index des leeren Teils ist immer 16

var EmptySquare = 16;

//Hier erstellen wir ein Plugin und nennen es fifteen.

$.fn.extend({ fifteen:

function(square\_size) {

// Zuordnung des HTML elements in welches wir das Spiel speichern wollen,

// Hier der selector - "#game\_object" (siehe Konstruktor ganz unten)

var gameObjectElement = '#' + $(this).attr('id');

var sqSize = square\_size + 'px';

var boardSize = (square\_size \* 4) + 'px'; // die Größe eines Teils wird mitgegeben

// In den selektor wird ein DIV Objekt "board" eingesetzt und das CSS dafür gesetzt

$(gameObjectElement).html('<div id="board"></div>');

$('#board').css({ position:'absolute', width: boardSize, height: boardSize, border: '1px solid gray' });

// dann in den neuen board DIV die 16 Teile angehängt

for (var i = 0; i < 16; i++) {

// Der Trick ist, dass jedes Teil das gesamte Hintergrundbild hat, jedoch mit einem anderen Offset

$('#board').append("<div style='left: " + ((i % 4) \* square\_size) + "px; top: " + Math.floor(i / 4) \* square\_size + "px; width: " + square\_size + "px; height: " + square\_size + "px; background-position: " + (-(i % 4) \* square\_size) + "px " + -Math.floor(i / 4) \* square\_size + "px '></div>");

}

// Das leere Teil Nr 16 erhält ein anderes CSS

$('#board').children("div:nth-child(" + EmptySquare + ")").css({backgroundImage: "", background: "#ffffff"});

// Ein click event für jeden Teil

$('#board').children('div').click(function() {

Move(this, square\_size);

});

}

});

// Move() ist die function wenn ein Teil geklickt wurde

// Es hat 2 parameter,

// 1. welches Teil clicked, und

// 2. die Breite eines Teils

function Move(clicked\_square, square\_size) {

// jetzt wird geprüft, ob das geclickte Teil an das leere angrenzt,

var movable = false;

//dazu werden die Koordinaten ermittelt und in Variablen gespeichert

var oldx = $('#board').children("div:nth-child(" + EmptySquare + ")").css('left');

var oldy = $('#board').children("div:nth-child(" + EmptySquare + ")").css('top');

var newx = $(clicked\_square).css('left');

var newy = $(clicked\_square).css('top');

// Ist das leere im Norden?

if (oldx == newx && newy == (parseInt(oldy) - square\_size) + 'px')

movable = true;

// Ist das leere im Süden?

if (oldx == newx && newy == (parseInt(oldy) + square\_size) + 'px')

movable = true;

// Ist das leere im Westen?

if ((parseInt(oldx) - square\_size) + 'px' == newx && newy == oldy)

movable = true;

// oder im Osten?

if ((parseInt(oldx) + square\_size) + 'px' == newx && newy == oldy)

movable = true;

// Wenn eines zutrifft, dann ....

if (movable) {

// erhöhen des zindex damit das Teil im Vordergrund

$(clicked\_square).css('z-index', zi++);

// Animation zum vertauschen

$(clicked\_square).animate({ left: oldx, top: oldy }, 200, function() {

//und das leere Teil erhält die neue Position

$('#board').children("div:nth-child(" + EmptySquare + ")").css('left', newx);

$('#board').children("div:nth-child(" + EmptySquare + ")").css('top', newy);

});

}

}

// Damit gehts los unser neues plugin wird aufgerufen

$('#game\_object').fifteen(120);

});

Das Stylesheet für die Teile beinhaltet für jedes Teil das gesamte Hintergrundbild, welches dann nur entsprechend im Skript positioniert wird.

#board div {

background: url("../images/kueste.jpg") no-repeat scroll 0 0 #ffffff;

cursor: pointer;

height: 120px;

line-height: 120px;

position: absolute;

text-align: center;

width: 120px;

/\* css3 shadow \*/

-moz-box-shadow: inset 0 0 20px #555555;

-webkit-box-shadow: inset 0 0 20px #555555;

-ms-box-shadow: inset 0 0 20px #555555;

-o-box-shadow: inset 0 0 20px #555555;

box-shadow: inset 0 0 20px #555555;

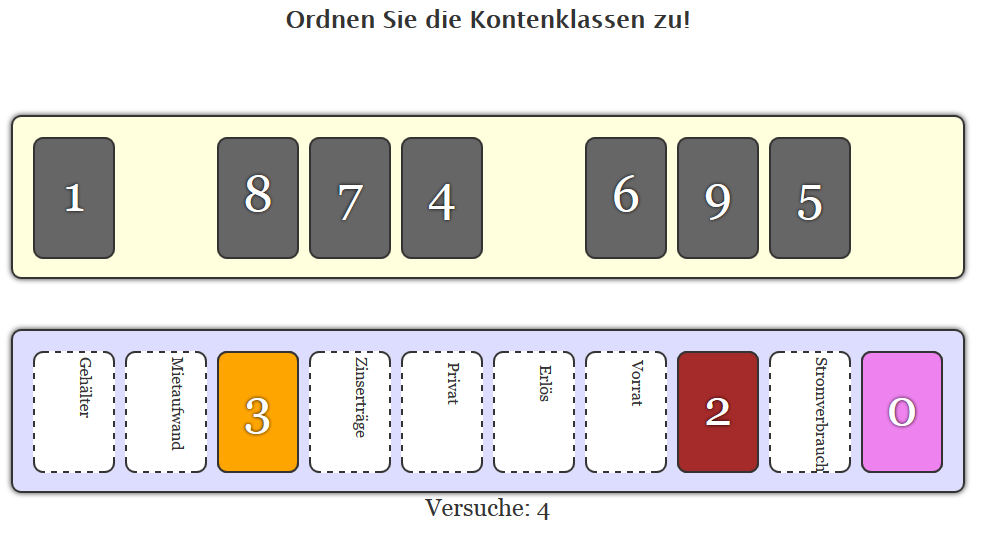
}

Habe für die Entwicklung dieses Beispiels auch ein Video gemacht: www.ats.eu.com/puzzle

Darüber hinaus gibt es sehr komfortable Plugins für Puzzle mit wesentlich mehr Funktionen, ein Beispiel finden Sie auf der DVD unter „Profi\_puzzle“.

### Drag & drop

Für das abschließende Beispiel für „drag & drop“ habe ich ein Video gemacht, welches Sie unter [www.ats.eu.com/drop](http://www.ats.eu.com/drop) ansehen können. Es zeigt die ideale Kombination aus HTML & CSS & Jquery. Wobei beim HTML Teil nicht mehr viel übrig geblieben ist.



# Werkzeuge

Ich arbeite am liebsten mit dem Dreamveawer, da er ab der Version CS5 am aktuellen Stand ist und z.B. bei CSS gute Vorschläge macht. Nachteil: nicht kostenlos!

**Ein Tipp**: die Adobe-Produkte der Serie CS2 gibt es jetzt, da nicht mehr im Verkauf, kostenlos als Downloads: <http://www.adobe.com/de/downloads/cs2_downloads/index.html>

Es gibt aber zahlreiche kostenlose Tools:

Die großen Hersteller wie Oracle (netbeans) und Microsoft (Visual Studio) bieten mittlerweile auch kostenlose Werkzeuge zur professionellen Entwicklung an. Diese orientieren sich allerdings eher an deren Flaggschiffen (Java oder ASP.net) und sind keine reinen HTML Editoren.

Manche Editoren wie der Free Javascript editor von Yaldex haben sich eben z.B. auf Javascript spezialisiert oder sind wie der kostenlose HTML Editor Phase5 leider nicht auf aktuellem Stand.

Bluefish für Linux…..

# Quellen & Links:

<http://www.mediaevent.de/css/>

[http://webkrauts.de](http://webkrauts.de/)

<http://css-tricks.com>

<http://www.w3schools.com/html/html5_intro.asp>

<http://www.html-world.de/>

<http://procssor.com/> (zum ordnen von umfangreichen CSS)

<http://t3n.de/news/css3-neue-moglichkeiten-hintergrundgrafiken-328060/>

<http://webkompetenz.wikidot.com/>

<http://www.html5rocks.com/de/tutorials/>