Hochschule Mittweida (FH)
University of Applied Sciences

Fachbereich Medien


Tanneberger, Sandro

# Weblayouts mit Cascading Stylesheets Level3 - Dokumentation, Anwendung & Analyse

**- eingereicht als Bachelorarbeit -**

Hochschule Mittweida – University of Applied Science (FH)

| **Erstprüfer** | **Zweitprüfer** |
|---|---|
| Prof. Dr.-Ing. Robert J. Wierzbicki | Dipl.-Ing. Sieglinde Klimant |

Chemnitz, 2010

# I. Bibliografische Beschreibung / Referat

Tanneberger, Sandro:
**Weblayouts mit Cascading Stylesheets Level3 -
Dokumentation, Analyse und Anwendung**

2010 - 121 Seiten, Hochschule Mittweida (FH), Fachbereich Medien,
Bachelorarbeit

## I.1. Referat

Diese Bachelorarbeit beschäftigt sich intensiv mit der Level 3 - Version des Webstandards CSS (Cascading Stylesheets) und überprüft, inwieweit er gegenwärtige Layouterfordernisse erfüllt. Die Arbeit stellt Probleme der Vorgängerversion CSS2.1 heraus, reflektiert die Ursachen und überprüft, welche Auswirkungen diese auf den Stand des Webdesigns im Jahre 2010 haben. Im Anschluss werden die CSS Level3 Merkmale in Form einer praxisnahen Referenz dokumentiert, anhand von Anwendungs-Beispielen getestet und auf ihre Zweckmäßigkeit hin analysiert.
Danach widmet sich der Autor der Frage, inwieweit der Standard zum Zeitpunkt dieser Arbeit einsatzbereit ist. Der Autor geht dabei auf externe Faktoren, wie Webbrowser und Nutzungsverhalten, sowie auf interne Faktoren, wie den technischen und organisatorischen Stand der Arbeitsgruppen, die die CSS3-Module entwickeln, ein. Mithilfe eines Praxisbeispiels, das einige der CSS3-Merkmale verwendet, wird der Autor Methoden vorstellen, wie CSS3 schon zum jetzigen Zeitpunkt in Webseiten eingebunden werden kann, ohne ältere Webbrowser zu vernachlässigen.

# II. Inhaltsverzeichnis

## II. Inhaltsverzeichnis

# III. Abbildungsverzeichnis

## III. Abbildungsverzeichnis

# IV. Tabellenverzeichnis

# V. Quellcodeverzeichnis

## V. Quellcodeverzeichnis

# VI. Abkürzungsverzeichnis

| Abkürzung | Bedeutung |
|---|---|
| CMS | Content Management System |
| CR | Candidate Recommendation |
| DSSSL | Document Style Semantics and Specification Language |
| DOM | Dynamic Object Model |
| EULA | End User License Agreement |
| HTML | Hypertext Markup Language |
| JS | Java Script |
| OTF | Open Type Font |
| SGML | Standard Generalized Markup Language |
| SVG | Scalable Vector Graphics |
| TTF | Truetype Font |
| WaSP | Web Standards Project |
| W3C | World Wide Web Consortium |
| WOFF | Web Open Font Format |
| XML | Extensible Markup Language |

# 1. Einleitung

Der Webstandard „Cascading Stylesheets“ hat in den 2000er Jahren einen Aufschwung hinsichtlich Nutzung und Verbreitung erlebt[1] und hat als visuelle Beschreibungssprache für HTML- und XML-Dokumente einen festen Platz in der Webentwicklung eingenommen. Die Philosophie, den mit (X)HTML strukturierten Inhalt von der durch CSS-Regeln definierten Gestaltung zu trennen, hat die frontendbezogene Webentwicklung auf ein professionelleres Level gehoben. Während sich Designkonzepte in den letzten Jahren stetig weiterentwickelt haben, ist CSS2.1 in vielerlei Hinsicht in die Jahre gekommen und offenbart insbesondere im Hinblick auf komplexere Layouts große Defizite. Die Spezifikation von CSS Level 3 wächst zunehmend und bietet mit durchdachten grafischen un dekorativen Eigenschaften, Typographie-Verbesserungen, Animationen & Übergängen, Medien Abfragen und den 3 neuen Layoutkonzepten intuitive und zeitsparende Konzepte für modernes Webdesign. Einige Module und Eigenschaften werden bereits von modernen Webbrowsern unterstützt und das Zusammenspiel mit HTML5 scheint das Frontend-Development für die Zukunft zu rüsten. Könnte CSS3 aus Sicht der Webentwickler nicht früh genug vollständig implementiert werden, bleibt ein fader Beigeschmack im Hinblick auf die momentane Browserverbreitung. Der Internet Explorer gestaltet sich als zu langsam in der Implementierung neuer Eigenschaften heraus. Dazu gesellen sich interne Probleme bei den CSS-Arbeitsgruppen, die die Entwicklung bei einigen Modulen verlangsamen.

## 1.1. Ziel und Aufbau der Arbeit

Ziel der vorliegenden Arbeit ist es, einen praxisorientierten und umfangreichen Überblick über die neueste Spezifikation des CSS-Webstandards zu liefern. Anhand von Anwendungsbeispielen werden die neuen Module mit ihren Eigenschaften dokumentiert, getestet, analysiert und kritisch hinterfragt. Der Autor prüft, ob die neue Spezifikation sinnvolle Antworten auf gegenwärtige Layouterfordernisse liefert, die mit CSS2.1 teilweise nur umständlich gelöst werden können. Diese Bachelorarbeit unterbreitet mit den gewonnenen Erkenntnissen einen Einblick, wie CSS3 schon heute in Teilen eingesetzt werden kann und welchen Methoden dafür zum Einsatz kommen sollten.

Die Arbeit wird mit einem kurzen allgemeinen Teil über die Grundkonzeption und bisherige Entwicklung von Stylesheets beginnen.
Darauf aufbauend analysiert der Autor, ob die gegenwärtige CSS2.1-Spezifikation überhaupt ausreichend ist, um aktuelle Layouts sinnvoll umsetzen zu können, oder ob und warum Notlösungen notwendig sind, um zum Ziel zu gelangen.

---

[1] [ZDNET 2006]

Daraus entwickelt der Autor eine Aufbereitung der aktuellen Probleme, die entweder im Standard selbst begründet sind oder sich durch unzureichende Browser-Implementierung ergeben.
Der Hauptteil der Arbeit beschäftigt sich mit der Dokumentation, Anwendung und Analyse der dritten Spezifikation des Web-Standards. Die im vorhergehenden Teil aufgestellten Erfordernisse werden mit den neuen Eigenschaften abgeglichen und es wird geprüft, ob und in welchem Umfang CSS3 Antworten auf diese und andere Probleme liefert. Im vorletzten Teil wird anhand eines Praxisbeispiels erörtert, inwiefern CSS3 schon einsetzbar ist, mit welchen Konzepten dies geschehen könnte und welche Faktoren einer Nutzung der stabilen Merkmale im Wege stehen können.

## 1.2. Methodik

Die Dokumentation und Anwendung des CSS3-Standards wird anhand der offen verfügbaren Publikationen des W3C vorgenommen[2] Diese beschreiben die einzelnen Module der CSS3-Spezifikation sehr genau – direkt von der Quelle.
Bei den Praxisbeispielen ist zu beachten, dass derzeitig alle CSS3-Eigenschaften und Werte mit einem entsprechenden „Browser-Prefix" gekennzeichnet werden müssen, da sie größtenteils noch von experimenteller Natur sind. Der Autor wird diese in den Codebeispielen entsprechend markieren. Aufgrund der technischen Überlegenheit von Apples Safari 5 Browser im Hinblick auf CSS3, werden alle schon nutzbaren Merkmale mit ihm getestet, da dieser die meisten CSS3-Merkmale interpretieren kann.
Gleichzeitig erfolgt ein Vergleichstest mit den anderen derzeitig verbreiteten Browser-Engines. Diese sind Trident (Internet Explorer 6.0-8.0), Gecko (Mozilla Firefox 3.6), Presto (Opera 10.5), und Webkit (Safari 5.0, Google Chrome 5.0)[3] . Die strukturelle Auszeichnung der in den Praxisbeispielen verwendeten Elemente wird unter der HTML 5-Spezifikation[4] erfolgen. Dies geschieht im Hinblick auf die Einstellung von XHTML2 Ende 2009 seitens des W3C[5] und garantiert damit ein Höchstmaß an Aktualität.

## 1.3. Vorraussetzungen

Aufgrund des Umfangs des Themengebiets wird eine grundlegende Vertrautheit mit den CSS Schlüsselkonzepten vorrausgesetzt. Dem/Der LeserIn sollten die Konzepte der Selektion, des Box Models, der Kaskade, der Spezifität, den Floats und den grafischen Eigenschaften (Background, Borders) in Theorie und Praxis bekannt sein. Ein Verständnis von HTML, insbesondere der neuen HTML5-Version, ist außerdem von Vorteil[6]

---

[2] [W3C] 001 http://www.w3.org/Style/CSS/current-work
[3] [HITSLINK] [BROWSER-STATISTIKEN] Die Versionsnummern in Klammern stehen für die in dieser Arbeit verwendeten Browser
[4] Vgl. A4
[5] [W3C NEWS]
[6] HTML5-Überblick siehe A4

# 2. Cascading Stylesheets bis Level 2.1

## 2.1. Allgemeines

Cascading Stylesheets – frei übersetzt: Verkettete Formatierungsvorgaben – ist eine vom World Wide Web Consortium (W3C) verabschiedete offen-dokumentierte Formatierungs-Sprache, mit der es möglich ist, die visuelle Darstellung strukturierter HTML- oder XML-Dokumente für verschiedene Ausgabemedien1[7] zu beschreiben und zu definieren. CSS ist aber nicht nur auf HTML anwendbar, sondern auch auf reine XML-Dokumente, sowie die xml-basierte Multimedia-Auzeichnungssprache SMIL und das Vektor-Grafikformat SVG. Neben CSS existieren noch die transformierende Stylesheetsprache XSL (Extensible Style Language) für XML-Dokumente und DSSSL für SGML-Dokumente.
CSS wurde 1994 von Håkon Wium Lie entworfen[8] . Dies geschah als Alternative auf die stärkerwerdende Tendenz in Webautorenkreisen, Inhalte direkt mit HTML zu formatieren[9] und damit die eigentliche Funktion von HTML , die reine logische und semantische Strukturierung von Inhalten, zu verwässern.[10]

## 2.2. Das W3C

Das W3C (Word Wide Web Consortium) ist eine Vereinigung verschiedener Experten, Vereinen, Entwicklern und IT-Unternehmen und wurde 1994 von Tim-Berners-Lee[11] gegründet. Die Hauptaufgabe liegt in der gemeinsamen Entwicklung und Betreuung von offenen Webstandards, Webtechnologien und Protokollspezifikationen. Oberste Zielstellungen sind hierbei die Gewährleistung der Kompatibilität der Standards zueinander sowie die Offenhaltung der Technologien. Dies soll verhindern, dass sich einzelne Unternehmen Webstandards zu eigen machen können, um damit kommerzielle Interessen zu verfolgen und den Markt aufzusplittern.[12]
Ziel des W3C ist das „One Web" – die Interoperabilität von Webtechnologien, die eine stetige Weiterentwicklung des gesamten Webs garantiert.
Die vom W3C entwickelten Webstandards sind keine Vorschriften, sie werden nach einem festgelegten Prozess als Empfehlungen – sogenannte Recommendations – veröffentlicht. Eine „Recommendation"[13] ist dabei die letzte Stufe in einem mehrstufigen Entwicklungsprozess[14] , sie ist die finale Version eines Standards.

---

[7] z.B. Bildschirme, Drucklayouts, Handhelds, Mobilgeräte, SPrache
[8] [W3C] 002 http://www.w3.org/People/howcome/p/cascade.html
[9] [WIUM LIE & BOS 1999]
[10] Vgl. [WIUM LIE]
[11] Der Erfinder des WWW und HTML-Standards
[12] [W3C 023] „Web for all - Web on everything"
[13] Kurz REC
[14] Vgl. [W3C 003] und A1

## 2.3. Aufbau, Struktur und Syntax – Die Grundzüge von CSS

Das Stylesheet ist eine Sammlung von Anweisungen, die das Aussehen und die Position eines (X)HTML-Elementes bzw. dessen Inhalt beschreiben.

### 2.3.1. Selektion

Eine Stylesheetanweisung besteht dabei aus 2 Teilen, dem Selektor, dieser benennt ein oder mehrere zu formatierende Elemente. Selektoren können Typselektoren, Universal Selektoren, Attribut-Selektoren, Klassen/ID-Selektoren Pseudo-Klassen, Pseudo-Elemente oder eine Kombination aus diesen sein.
Der zweite Teil ist der Deklarationsblock, bestehend aus einer oder mehreren Eigenschaften und den dazugehörigen Werten.

```
p { color: #FF0000; };
```

Q1: Einfache Selektion eine <p>-Tags und Zuweisung der Schriftfarbe rot

```
#topthema {font-weight:bold;}
```

Q2: ID-Selektion

```
div#hauptartikel > h2
```

Q3: Direkte Unterordnungs-Selektion: Nur direkt untergeordnete h2-Kindelemente des div-Containers #hauptartikel werden ausgewählt.

### 2.3.2. Spezifität

Durch die flexiblen Möglichkeiten der Selektion kann es zu Konflikten kommen, wenn ein und dasselbe Elemente über mehrere verschiedene Selektionswege ausgewählt und visuell formatiert werden soll. Eine Deklaration muss sich durchsetzen.
Die Kriterien hierfür liefert die Spezifität – die Wichtung von CSS-Selektoren.

Je nach Zusammensetzung des Selektors wird ihm eine Wertung innerhalb eines viergliedrigen Variablensystems erteilt:

**a** Handelt es sich um eine Inline-Anweisung mit dem `style`-Attribut, erhält diese Variable eine **1**, ansonsten eine **0** (innerhalb eines separaten Stylesheets oder Style-Anweisungen im Kopf des Dokumentes)

**b** Die Anzahl der ID's innerhalb des Selektors

**c** Die Anzahl der Klassen, Pseudoklassen und Attribute innerhalb des Selektors

**d** Die Anzahl der einfachen Elemente sowie Pseudoelemente innerhalb des Selektors. Die Kombinatoren selbst haben keinerlei Einfluss auf die Wichtung

Die Wichtung anhand der CSS Spezifität. Beide Codes selektieren das gleiche Element .

```
#hauptartikel h2 strong {color:#ff0000;}
```

Q4: a=0 b=1 c=0 d=2 (1 ID - 0 Klassen - 2 einfache Elemente)

```
CSS2 div h2.politik strong {color:#00ff00;}
```

Q5: a=0 b=0 c=1 d=3 (0 Ids - 1 Klasse - 3 einfach Elemente)

Die erste CSS-Anweisung ist bei diesem Vergleich die Spezifischere[15] und ihr Deklarationsblock wird angewandt. Die Wichtung der 4 Variablen erfolgt von links nach rechts absteigend, eine ID ist immer höherwertiger als die folgenden Elemente.

### 2.3.3. Die Kaskade

Stildeklaration laufen wie ein Wasserfall von oben nach unten. Erscheint eine Anweisung später im Stylesheet und besitzt eine gleiche oder höhere Spezifität, erhält sie eine höhere Wichtung und übersteuert die vorhergehende Anweisung.

### 2.3.4. Vererbung

Ein weiteres Kernelement von CSS ist die „Vererbung". Bestimmte Werte werden vom jeweiligen Elternelement an die Kindelemente übertragen. Diese sind die Werte der Eigenschaften `font-family`, `font-size` und `color`.
Grafische Eigenschaften wie Hintergrundbilder, Rahmen oder
Innen- und Außen-Abstände werden standardmäßig nicht übernommen[16]

[15] 1 ID vs. 0 IDs
[16] außer es ist der Wert `inherit` angegeben (übernehmen)

### 2.3.5. Das Box-Modell

Das Box-Modell bildet ein zentrales Grundkonzept in der CSS-Spezifikation.
Die Dimensionen des Inhaltes eines jeden HTML-Elementes werden als rechteckig betrachtet – es entsteht eine den Inhalt umschließende Box mit den Eigenschaften: Breite und Höhe (`width & height)`, Innenabstand (`padding`), Außenabstand (`margin`) und Umrahmung (`border`). Es werden dabei Inline- und Blockboxes unterschieden. Die Art des Box kann mit der CSS-Eigenschaft `display` verändert werden. Mit dem Wert `inline` verhält sie sich wie eine Inline-Box, mit dem Wert `block` wie eine Blockbox.

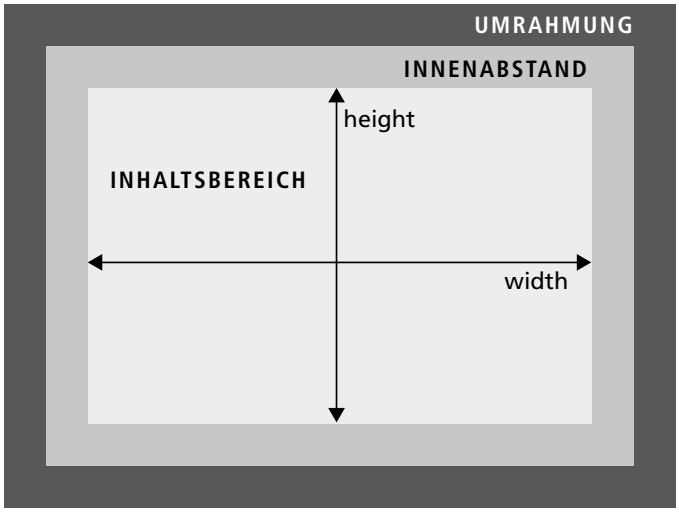



Abb. 1: Das Box-Modell. width & height entsprechen den Dimensionen des Inhaltsbereiches. Innenabstände und Umrahmungen werden hinzuaddiert

### 2.3.6. Positionierung

Die Positionierung von Elementen wird mit Hilfe der CSS-Eigenschaft `position` sowie den Verschiebungs-Eigenschaften `top, left, right bottom` realisiert – im Zusammenspiel mit den Möglichkeiten, die das Boxmodell, die Dimensionseigenschaften[17] und die grafischen Eigenschaften von CSS bieten, können damit auch komplexe Layoutaufgaben gelöst werden. Es können 4 Arten der Positierung unterschieden werden

`position:static` Dies ist der Standardwert von jedem Element. Die Boxes positionieren sich innerhalb des normalen Dokumentsfluß und hinsichtlich ihres Box-Typs [18]

[17] `width, min-width,max-width, height, min-height, max-height`
[18] Vgl. 2.3.6. Inline-Boxes und Block-Boxes

`position:relative` Mit diesem Positionierungstyp wird die Position des Elements in Bezug zu seinem normalen Platz im Dokumentfluss[19] definiert. Mit den Werten der Verschiebungs-seigenschaften `top, right, bottom, left` kann das Element nun relativ zu seiner ursprünglichen Position verschoben werden.

`position:absolute` Mit diesem Positionierungstyp wird das Element aus dem Dokumentfluss herausgelöst. Es positioniert sich absolut an einem umschließenden Elternelement, sofern dieses ebenfalls mit der `position`-Eigenschaft versehen ist. Andere Elemente ignorieren das so positionierte Element und verhalten sich so, als wäre es überhaupt nicht vorhanden. Mit den Verschiebungs-eigenschaften `top`, `right`, `bottom`, und `left` kann das Element wiederum bezüglich den Dimensionen des umschließenden Elementes verschoben werden.

`position:fixed` Dieser Typ weist die gleichen Merkmale wie die absolute Positionierung auf, mit dem Unterschied, dass sich das so positionierte Element immer am Viewport[20] des Ausgabemediums ausrichtet, und damit beim Scrollen „mitläuft".

### 2.3.7. Floats

Mit der Eigenschaft `float`[21] wird ein Element an der linken oder rechten Seite des umschließenden Elements positioniert. Text und andere Inline-Inhalte nehmen dabei den restlichen zur Verfügung stehenden Platz ein – sie umfließen das Float-Element.

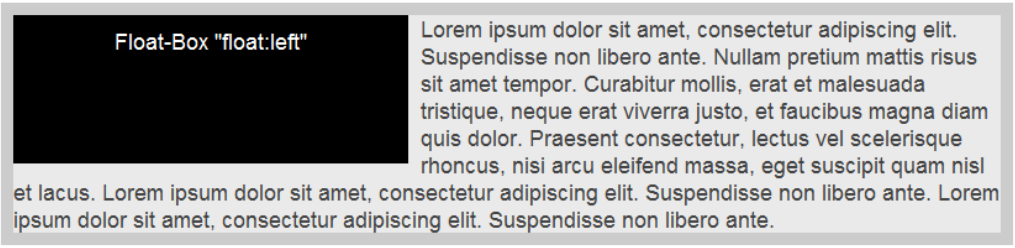



Abb2.: Standard-Float-Verhalten - Linksfloatendes Element wird vom Text umschlossen

[19] die Position, die es mit `position:static` hätte
[20] siehe VI
[21] engl. für „fließen"

Die float-Eigenschaft wird im Webdesign verwendet, um mehrspaltige Layouts mit nebeneinanderstehenden Containerboxen zu erstellen.


Abb3.: Spaltenlayouts mit Hilfe der `float`-Eigenschaft

## 2.4. Die Trennung von Inhalt und Präsentation / Semantik in der Webentwicklung

Der Grund für den Einsatz von Stylesheets liegt der Philoshophie zugrunde, (X)HTML nur für die logische/semantische Strukturierung der Inhalte einzusetzen. Die visuelle Formatierung ist reine Aufgabe von CSS. Semantische Strukturierung bedeutet, dass jeder Inhaltsbaustein im Dokument mit dem Tag umschlossen wird, das seiner inhaltlichen Bedeutung – seinem Sinn – gerecht wird. Die Hauptüberschrift beispielsweise wird mit den HTML-Tag <h1> ausgezeichnet, weitere Unterüberschriften mit den folgenden Tags <h2> bis <h6>. Ein Textabsatz wird mit dem dafür vorgesehenen <p>-Tag ausgezeichnet, Hervorhebungen innerhalb eines Absatzes werden
wiederum. mit  oder  ausgezeichnet.

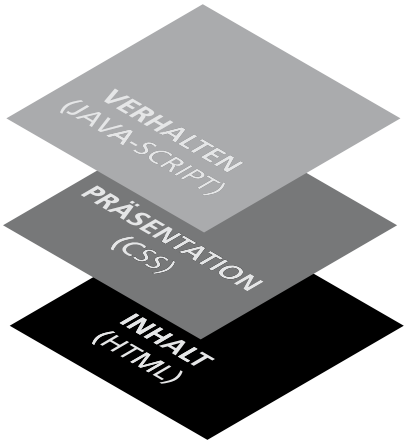



Abb4.: Das 3-Ebenen-Modell der Webtechnologien[22]

[22] Vglt. [SITEPOINT 2007]

Vorteile dieser Philosophie sind die verbesserte Maschinenlesbarkeit und -interpretierbarkeit, z.B. im Hinblick auf Suchmaschinen[23] . Dadurch, dass sich in der HTML-Datei nur der reine Inhalt und seine durch die entsprechenden Tags ausgezeichnete Struktur befinden, verbessern sich die Möglichkeiten des Datenaustauschs, beispielsweise bei der Generierung eines RSS-Feeds (bei XML-konformen Markup).

**Die Vorteile kristallisieren sich wie folgt heraus.**
Durch die Verwendung von separaten CSS-Definitionen, die in einer oder einer kleinen Gruppe von CSS-Datei/en zentral abgelegt werden, verringert sich der Aufwand bei Layoutveränderungen von umfangreichen HTML-Webseiten immens. Eine Anpassung muß nur im CSS-File definiert bzw. verändert werden und wirkt sich auf alle Seiten der Präsenz aus. Selbst komplexe Redesigns können so zentral aus einer CSS-Datei verwaltet werden. Einher geht damit die Übersichtlichkeit des HTML-Quellcodes. Durch das Fehlen von präsentationellem Markup und die Auslagerung in eine externe CSS-Datei werden HTML-Dokumente deutlich schlanker und für Entwickler überschaubarer.[24]
CSS Probleme und der Status Quo

[23] Vgl. [GOOGLE]
[24] Vgl. A1 Tabellenlayout vs. CSS-Layout

## 2.5. Das Browser-Problem

Als clientseitige[25] Beschreibungssprache ist CSS und die korrekte, standardkonforme Darstellung von dem verwendeten Browser des Nutzers abhängig. Dies führt seit der ersten CSS-Version zu Darstellungs- und Implementationsproblemen, die einer schnellen und umfassenden Verbreitung als Web-Layoutsprache im Wege standen.
Diese sind aber nicht im Standard selbst begründet, sondern vielmehr in der vergangenen Politik bestimmter Browserentwickler sowie in der Art und Weise der Browsernutzung eines Teils der Internetuser.
Als CSS1 im Dezember des Jahres 1996 die Freigabe vom W3C erhielt, war Netscape Navigator 3.0 der dominierende WWWW-Browser. [26]
Im Laufe des gleichen Jahres entwickelte Microsoft den Internet Explorer, der in seiner Version 3.0 der erste kommerzielle Browser war, der Stylesheets überhaupt unterstützte bzw. interpretieren konnte.[27]
Im sogenannten „Browserkrieg" um Markanteile zwischen beiden Browsern, zog Netscape mit der Navigator 4.0 Version nach. Doch durch die standardmäßige Installation des Internet Explorers 4.0 mit dem Windows95Plus und Windows98-Betriebssystemen sowie dem MS Office-Paket konnte Microsoft seine Marktanteile bis 1998 auf 50/50 ausgleichen und mit dem Erscheinen des IE5 und später IE6 bis 2002 auf knapp 90% anheben.[28] Der Kampf um Marktanteile war zugunsten Microsofts entschieden.

Der Browserkrieg war gekoppelt mit dem gegenseitigen Übertrumpfen durch mehr oder weniger sinnvolle neue Features und Technologien. Dies ging einher mit der Implementierung und Neuinterpretation noch nicht freigegebener W3C-Standards und vor allem der Entwicklung eigener, proprietärer HTML-Elemente wie den präsentationellen `<font>` oder `<marquee>` Tags des Internet Explorers.
Die ursprüngliche Idee des W3C der Interoperabilität von Webtechnologien – das „One Web" – ging im „Browserkrieg" unter, da Webseiten in dieser Zeit faktisch gezielt für einen der beiden Browser optimiert werden mussten.

Besonders hart hat diese Entwicklung die Verbreitung der Stylesheets getroffen. Mit dem IE4 (1997) und seiner komplett neu aufgesetzten „Trident"-Rendering Engine[29] traten massive Darstellungsprobleme auf, die auch bei nachfolgenden Versionen nicht konsequent bereinigt wurden.[30] Durch das bis in die frühen 2000er Jahre andauernde

[25] Ausführung auf Nutzerseite
[26] 1996 IE 6.93% Netscpace 82.77%
[27] [WIUM LIE & BOS 1999] „Browsers"
[28] [WIKIPEDIA]
[29] Name eines Programmmoduls, welches zur Darstellung des HTML-Codes einer Webseite oder einer grafischen Benutzeroberfläche verwendet wird

[30] [WEBSTANDARDS]

Quasi-Monopol des IE5 (90% Marktanteil)[31] stagnierte die Verbreitung von Stylesheets und damit auch die konsequente Weiterentwicklung. Table-Layouts und Frame-basierte Webseiten waren bis dato die zuverlässigste Variante, komplexere Weblayouts browserübergreifend umzusetzen.[32] Erst mit dem Erscheinen des IE6 im Dezember 2001, der nun erstmalig CSS1 vollständig und CSS2.1 zumindest teilweise unterstützte, veränderte sich die Nutzung von CSS in den Webentwicklerkreisen zum Positiven.

Bis 2006 stieg der Anteil von CSS-Webseiten von 10% auf 60%.[33] Außerdem sorgten die zunehmende Verbreitung neuer webstandardkonformer Browser wie Opera und Mozilla / Firefox für wachsenden Druck auf Microsoft.
Microsoft verbesserte zwar ebenfallls mit jeder nachfolgenden Version die CSS-Unterstützung des Internet Explorers, aber erst mit dem Erscheinen des Internet-Explorer 8.0 im Jahr 2009 wurde endlich die standardkonforme Unterstützung der CSS2.1 Spezifikation erreicht. Doch das zähe Update-Verhalten der IE-Nutzer[34] sorgt nach wie vor dafür, dass auf einen beachtlichen Teil der Surfer, die auf die technisch veraltete Browsern zurückgreifen, Rücksicht genommen werden muß.
Im Jahr 2010 sieht das ganze nicht deutlich positiver aus.

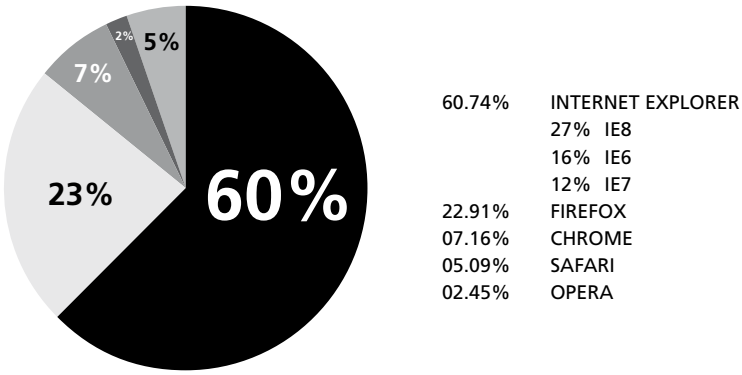



Abb5.: Globale Browser-Marktanteile Juli 2010[35]

2010 ist ein vollständig auf CSS2.1 Eigenschaften basierendes, stabiles Layout möglich, allerdings müssen nach wie vor alternative Stylesheets für einen großen Teil der Webuser (28% IE6 + IE7) angeboten werden, die das Aussehen der Webseiten entsprechend vereinfachen oder zumindest eine Informationszugänglichkeit gewähren

[31] [WIKIPEDIA]
[32] Vgl. A2
[33] [ZDNET 2006]
[34] Bis eine neue IE-Version ihre Vorgänger hinsichtlich der Nutzungsanteil überholt, vergehen im Schnitt 2-3 Jahre. 2009 betrug der Anteil an IE6 Usern 10% und IE7 knapp 30%
[35] [HITSLINK 2010]

## 2.6. Was fehlt CSS?

Bestimmte Merkmale der CSS-Spezifikation verkomplizieren den Umgang zu Layoutzwecken unnötig. Die CSS-Arbeitsgruppe des W3C bezieht – hinsichtlich der Entwicklung von CSS Level3 – Meinungen und Vebesserungswünsche der Webentwicklergemeinde ein, um auf die gestiegenen Ansprüche reagieren zu können.[36]
Die Hauptkritikpunkte an CSS2.1 stellen sich wie folgt dar.

### 2.6.1. Die Eigenheiten der Float-Layouts

#### a) Containing Floats

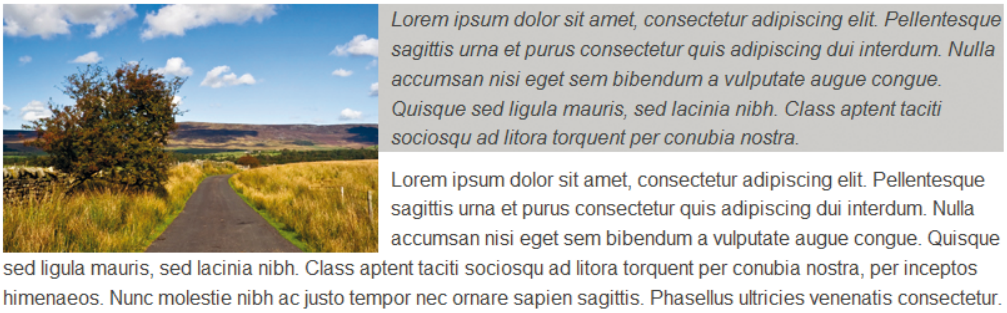



Abb. 6: Containing Floats. Überlappung des Bildes.

Wird einem `<div>`-Container, der ein Bild und den kursiven Einleitungs-Text umschließt , eine Hintergrundfarbe gegeben, fällt auf, dass die Höhe des Containers nur den Textabschnitt umschließt, das Bild aber über diesen hinausragt. Bei einer klassischen Artikelansicht ohne grafische Hervorhebungen ist dies erwünscht und diese Überlappung im Float-Verhalten der CSS-Spezifikation vorgesehen. Bei typischen Vorschaulisten, wie sie auf Blogs oder Nachrichtenportalen eingesetzt werden, ist dieses Verhalten allerdings unerwünscht. Es kommt zu Verschiebungen, weil sie nachfolgende Elemente ebenfalls an dem fließenden Bild ausrichten.

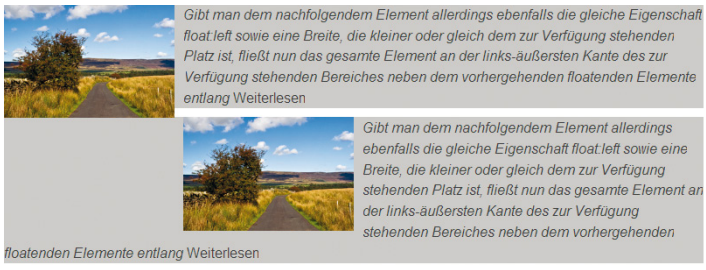



Abb. 6: Containing Floats 2 - Unschöne Verschiebungen durch die Float-Eigenschaft

[36] [WEBSTANDARDS FEEDBACK]

Dies ist ein Problem, dass bei der Konzeption des Float-Verhaltens in CSS1 und CSS2.1 nicht bedacht wurde – Anders gesagt: Die `float`-Eigenschaft, die rein für fließende Blockelemente in Textabschnitten angedacht war, wurde für die Erstellung von mehrspaltigen Layouts mehr oder weniger zweckentfremdet.

Es wird ein zusätzliches (logisch und semantisch nutzloses) Block-Element kurz vor dem Schließen des Elternelementes hinzugefügt. Dieses Blockelement besitzt die `clear`-Eigenschaft mit dem Wert `left` und die Höhe 0. Dadurch schließt das Elternelement am unteren Rand des floatenden Bildes ab. Eine andere Möglichkeit nutzt das Pseudoelement `after`

Nur durch diese sog. Clearfix-Methode[37], werden die Ansichten in Reihe gebracht.


Abb. 7: Die Clearfix-Methode bringt das Layout in die gewünschte Bahnen

[37] [WEBTOOLKIT 001]

### b) Gleichmäßige Höhen nebeneinander gestellter Spalten

Dieses Problem resultiert aus dem CSS-Merkmal, dass `<div>`-Container oder andere Blockelemente standardmäßig immer so hoch sind, wie der Inhalt, der sich in ihnen befindet[38] . Sie werden außerdem nicht von nebenstehenden Boxen in ihrer Höhe beeinflußt. Wird aber ein typisches zwei- oder dreispaltiges Layout mit farblich von einander abgehobenen Spalten erstellt, wirkt sich dies als unschönes Problem aus.

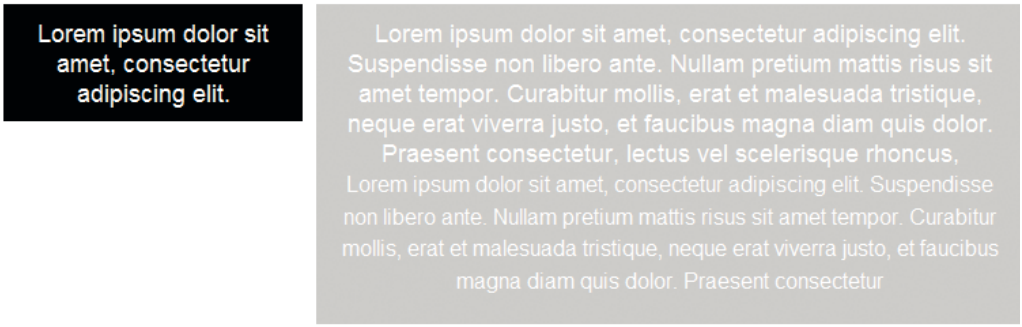



Abb. 8: Ungleichmäßige Höhen zweier Floatboxen im Spaltenkontext

Für diese Situation ist ein CSS-Merkmal wünschenswert, das die Höhe der kürzeren Spalten an die der längsten Spalte anpasst, um einen gleichmäßigen Abschluss der Elemente zu erreichen. Um dies zu erreichen, gibt es momentan nur einige Tricks, wie z.B. die Faux Columns-Methode[39] . Bei dieser Methode wird ein semantisch nutzloser umschließender `<div>`-Container um alle betroffenen Spalten gesetzt. Dieser Container besitzt ein vertikal gekacheltes Hintergrundbild, das das Spaltenaussehen abbildet, und somit gleichlange Spalten simuliert.

a) und b) beweisen in Ansätzen, das die `float`-Eigenschaft nicht genug ist, um mehrspaltige Layouts zu realisieren.

## 2.6.2. Vertikale Zentrierung

Ein Element horizontal zu zentrieren ist in CSS kein Problem.[40] Anders sieht es mit der vertikalen Zentrierung aus.[41] Nur durch Tricks, die ein umschließendes Element benötigen – dem die Eigenschaft `display:table-cell` gegeben wird, obwohl es sich

[38] Außer es ist eine feste Höhe angegeben
[39] [ALISTAPART 001]
[40] `margin:0 auto;`
[41] [WSSEXPERT] CSS2 besitzt keine Eigenschaft um Elemente vertikal zu zentrieren.

nicht um eine Tabellenzelle handelt, kann ein vertikales Zentrieren erreicht werden.[42] Weitere Umwege erfordern wieder ein inhaltlich nutzloses Element.[43] Auch das vertikale Zentrieren ist einer der wenigen Vorteile von Tabellenlayouts. [44]

### 2.6.3. Erweiterte Selektionsmöglichkeiten

Zurzeit fehlen noch einige Selektions-Optionen, die die spezifischere Auswahl eines gewünschten Elements stark vereinfachen würden. CSS fehlt zur Zeit eine Logik, um Selektionsmöglichkeiten zu verfeinern, darunter fallen:

**a) Die „Außer"-Selektion (Elemente auswählen, die nicht … sind / haben)**
Eine „Nicht"-Selektion ist derzeitig nur umständlich möglich. Zuerst müssen alle Elemente der Menge mit Stilanweisungen versehen werden. Dem Element, dass ausgeschlossen werden soll, muß eine Extra-ID vergeben werden, mit deren Hilfe die vorher getätigten Anweisungen wieder überschrieben werden.

**b) Selektion eines n-ten Elementes**
Weiterhin fehlt die Möglichkeit, durch Zählen ein spezifisches Element aus einer Menge von Elementen des gleichen Typs zu selektieren. z.B. den 2ten Absatz in einer Gruppe aus 5 Absätzen (`<p>`). Eine Möglichkeit das Auftreten von HTML-Tags zu zählen, würde zusätzliche sinnvolle Gestaltungsmöglichkeiten geben. So könnten bei einer Auflistung von Elementen jedem ungeraden Element eine andere Hintergrundfarbe zugewiesen werden – der klassische „Zebraeffekt".

**c) Elternselektion**
Eine weiteres sinnvolles Feature für das Layouten mit CSS ist die Eltern-Selektion. Zurzeit ist es nicht möglich, ein Element erst dann zu selektieren, wenn ein anderes sich in diesem „befindet" - z.B. nur `div`-Container auswählen, in denen sich eine h2-Überschrift befindet. Auch dieses Szenario lässt sich zurzeit nur mit einer unnützen Klassenvergabe an das Element mit der Überschrift lösen.

Sinnvoller wäre z.B. eine zukünftige Selektion mittels eines neuer Pseudoklasse

```
div:has("h2") { ... }
```

Q6: Beispielhafte Elternselektion

[42] allerdings nicht im Internet Explorer 6-7.
[43] [WEBMASTERPRO 001]
[44] `vertical-align="center"`

### 2.6.4. Fortgeschrittene Positionierung und Transformationen

Die Möglichkeiten der Positionierung[45] von Elementen in einem Layoutkontext sind mit CSS begrenzt. Es fehlt momentan die Möglichkeit, Elemente relativ zu einem bestimmten anderen HTML-Element zu positionieren. Die momentanen Positierungs-Methoden richten sich meistens nach dem linearen Dokumentfluß.
Elemente, die nacheinander im Markup auftreten, werden auch nacheinander abfolgend dargestellt (`position:static`). Einzig die `position:relative` Deklaration erlaubt eine Verschiebung, allerdings auch nur Bezug nehmend auf die ursprüngliche Position des Elements. Die weiteren Positionierungs-Methoden (`absolute` und `fixed`) lösen das Element komplett aus dem Dokumentfluss heraus und beziehen sich bestenfalls auf das umschließende Element[46] Mit der `float`-Eigenschaft können wiederum Elemente nur hinsichtlich ihres direkten Nachfolgers positioniert werden.

Soll z.B. eine Überschrift, die als Letztes im HTML-Markup auftaucht nach oben neben dem Hauptlogo positioniert werden, bleibt nichts anderes übrig, als sie mit `position:absolute` aus dem Dokumentfluß zu lösen und mit den `margin`-Werten an die gewünschte Position zu verschieben. Dabei müssen allerdings alle anderen Elemente, die „oben" bei dem Logo auftauchen und sich korrekterweise nach diesem ausrichten, ebenfalls verschoben und teilweise absolut positioniert werden, damit sie den erforderlichen Platz für die Überschrift schaffen.
Sinnvoller wäre eine zusätzliche Option, mit der dieses Element bezüglich der Position des Logo-Elementes verteilt werden kann.

Zurzeit ist also die visuelle Position eines HTML-Elements stark an die Position im HTML-Markup gekoppelt. Dies macht hinsichtlich der ursprünglichen Aufgabe von CSS[47] durchaus Sinn, da wissenschaftliche Dokumente einen streng linearen Fluß von oben nach unten aufweisen. Für mehrspaltige Weblayouts kann dies aber unter Umständen zu Problemen führen. Eine element-relative Positionierung würde mehr zur Inhalt-Design-Separation beitragen, da das Layout komplett von der Markup-Position eines Inhaltselementes losgelöst werden könnte.
Damit könnte ein und dasselbe HTML-Dokument komplett anders gelayoutet werden. Auch im Sinne der Internationalisierung, ergäben sich Vorteile:
In Ländern, in denen der Lesefluss rechts nach links orientiert ist, könnte das gleiche Dokument sozusagen visuell gespiegelt werden. Auf dem jetzigen Stand ist eine wirkliche Inhalt-Design-Separation mit CSS nicht möglich – CSS ist noch keine wirkliche Layoutsprache.

---

[45] Vgl. 2.3.6.

[46] bei `position:fixed` der Viewport, bei `position:absolute` das positierende umschließende Element)

[47] die Stylen von wissenschaftlichen Dokumenten und Artikeln, bei denen ein linearer Informationsfluss vorhanden ist

### 2.6.5. Erweiterte Background Kontrolle / Grafische & Dekorative Eigenschaften

Die `background`-Eigenschaft gehört zu den wichtigsten grafischen Eigenschaften von CSS. Mit `background` können individuelle Grafiken (Logos / Illustrationen / Formen, Fotos) aus dem Screendesign Teil des CSS-Layouts werden. Die bisherigen Background-Optionen sind zwar nützlich, aber mit dem steigenden Anspruch an Webdesigns doch etwas reduziert. Da Screendesigns oft mit professioneller Grafiksoftware erstellt werden, wäre es sinnvoll, einige der Features, die diese Tools bieten, direkt mit CSS möglich zu machen.

Grafisch aufwändige Screen-Designs machen es oft notwendig, mehrere Hintergrundbilder in einem Element verwenden zu können. Weiterhin fehlen zur Zeit sinnvolle Hintergrundtransformationen wie z.B. das Skalieren von Hintergrundbildern.
Insbesondere bei flexiblen Layouts, die sich in ihren Dimensionen an der Auflösung des Nutzers orientieren, wäre diese Option, sofern diese mit Prozentangaben möglich ist, ein sinnvolle Möglichkeit. Würde man diese BG-Transformationen mit der Unterstützung des Vektorgrafikformates SVG verbinden, wären dadurch komplett auflösungsunabhängige Designs möglich, in denen sich die Grafiken ohne Verluste hoch- und herunterskalieren lassen.
Weitere Transformationen, die momentan fehlen, sind Rotationen und Spiegelungen.

Farbverläufe sind wesentlicher Bestandlteil von modernen Weblayouts. Als Spalten- und Box-Hintergründe suggerieren Sie dem Nutzer eine gewisse visuelle Tiefe und Greifbarkeit dieser Elemente. Momentan müssen Farbverläufe als Grafik eingebunden werden. Weiterhin ist die pixelgenaue Hintergrund-Positionierung derzeit nur aus Sicht der oberen linken Ecke der Box möglich. Hat man eine Box mit flexibler Höhe und möchte eine Hintergrundgrafik 10px von der rechten unteren Ecke verschieben, geht das momentan nur, in dem man den Abstand in das Hintergrundbild einbindet.

Ein weiteres sinnvolles fehlendes Feature ist die Möglichkeit eigene Hintergrund-Grafiken für Umrahmungen zu bestimmen, um z.B. Bilderrahmen zu simulieren.
Weitere gewünschte grafische CSS-Eigenschaften aus Webentwicklerkreisen sind abgerundete Ecken mit einem zu definierenden Radius sowie Rendereffekte wie Schlagschatten, außenliegende Scheine und erweiterte Transparenzen.

Möchte sich CSS als Layoutsprache durchsetzen, sind diese Eigenschaften von großer Bedeutung, um den Workflow[48] beim Frontend-Design deutlich zu vereinfachen.

[48] Arbeitsablauf

### 2.6.6. CSS-Konstanten, erweiterte Vererbung

Bei längeren CSS-Dateien, in denen sich bestimmte Werte, wie Farbwerte oder Verzeichnisse für Hintergrundbilder wiederholen, würde es Sinn machen, diese als Konstanten einmalig definieren zu können, um diese dann als Platzhalter für den eigentlichen Wert einsetzen zu können.

```
@vars {
leftcolor: #ff00ff;
}

a {
color: var(leftcolor);
}
```

Q7 Beispielhafte Konstatenangabe mit CSS

Damit könnten Farbänderungen zentral an einer Stelle geändert werden und müssen nicht für jedes Element einzeln deklariert werden, wodurch der Wartungsaufwand sehr klein gehalten werden würde.
Momentan geht dies nur über eine gruppierte Selektion

```
a, p, div {
color: #ff00ff;
}
```

Q8 gruppierte Selektion als Alternative

Je komplexer das Stylesheet wird und je mehr Elemente dazu kommen, desto schwieriger kann es werden, den Überblick zu behalten. Konstanten würden der besseren Wart- und Überschaubarkeit dienlich sein.

### 2.6.7. Berechnungen

CSS bietet unterschiedliche fixe und relative Maße, um Dimensionen und Abstände anzugeben.[49] Probleme ergeben sich oftmals bei der Vermischung von fixen und relativen Größen. Ein klassisches Beispiel hierbei ist eine Box mit relativer Breite (z.B. 50%), die allerdings einen pixelgenauen Innenabstand (z.B. 5px an jeder Seite) bekommen soll.
Laut Box-Model (Vgl. 2.3.5.) wird der Innenabstand zur angegebenen Breite addiert, um die zu sehende Gesamtbreite zu erreichen.

Wenn man eine Box exakt 50% GesamtBreite erhalten soll, müßten die insgesamt 10px (`padding-left:5px, padding-right:5px`) von den 50% abziehbar sein

Mit festen Pixelmaßen bei der `width` ist das natürlich kein Problem, eine Rechnung würde sich wie folgt ergeben.

**gesamtbreite = width + padding-left + padding-right + border-left + border-right**

Aber: Die 50% entsprechen natürlich – je nach Browserfenstergröße – unterschiedlichen Pixelgrößen, sodass die eigentliche anzugebende Breite ebenfalls schwankt[50]

Momentan ist es nicht möglich, den festen Px-Wert von einem variablen Wert abzuziehen, sodass ein ebenso relatives – aber um 10px kleineres – berechnetes Endergebnis entsteht. Interessant wäre die Möglichkeit eine `width` in dieser Form anzugeben:

```
width: 50%-10px.
```

So könnte das geschilderte Problem komfortabel gelöst werden.
Eine mathematisch Berechnung für die Angabe von dynamischen Werten ist für bestimmte Layout-Szenarios unabdingbar.

[49] Vgl. A3
[50] Je nach Auflösung

### 2.6.8. Typographie

Typographie umfasst das komplette Schriftbild eines zu gestaltenden Mediums. Dazu gehören nicht nur die Auswahl der Schriftarten und -schnitte und -größen, sondern auch Zeilenabstände, Kerning[51] sowie die Silbentrennung.

Ein Hauptaugenmerk liegt bei der Textgestaltung mit CSS auf der Auswahl der Schriftfamilie bzw. Schriftart. Die Möglichkeiten beschränken sich – aufgrund der clientseitigen Verarbeitung – auf die im System des Nutzers vorhanden Schriften.[52]

Mit der CSS-Eigenschaft `font-family` kann die zu verwendende Schriftfamilie für HTML-Elemente angegeben werden, sowie Alternativschriften (sogenannte Fallback-Schriften) und der allgemeine Schrifttypus.

Die Folge daraus ist eine relativ eingeschränkte Nutzung einer Auswahl aus ca.
10 Standard-Schriften, von denen Webentwickler sicher gehen können, dass sie auf den meisten Systemen der Besucher installiert sind.[53]
Um mehr gestalterischen Spielraum zu ermöglichen, wäre es sinnvoll, Schriften mittels CSS einbetten zu können. Momentan lässt sich dieses Problem nur mit der Einbindung von Java-Script-Lösungen wie Cufon oder Flashlösungen wie sIFR beheben[54] .

Natürlich muß hierbei sichergestellt werden, dass kommerzielle Schriften von Besuchern nicht heruntergeladen bzw. weiterverwendet werden können, beispielsweise durch ein geschütztes Dateiformat, mit denen Schriftarten auf eine oder mehrere Domains beschränkt werden können.

---

[51] Buchstabenabstand
[52] Natürlich gibt es durch die Einbindung von flash/Javascript „Text-Replacement"-Techniken wie SIFR oder Cufon die Möglichkeiten, alternative Schriften einzubinden.
[53] [WEBMASTERPRO]
[54] [CUFON] [SIFR]

# 3. Cascading Stylesheets Level 3

## 3.1. Allgemeines & Entstehung

Der erste Entwurf für Cascading Stylesheets Level3 wurde am 14.April 2000 von Eric A. Meyer, der Teil der „CSS working group" ist, veröffentlicht.[55] Die Arbeitsgruppe(n) setzen sich aus Mitgliedern und Vertretern verschiedener Browser-, Software- und IT-Unternehmen zusammen, zu den bekanntesten gehören Apple, Google, HP, die Mozilla Foundation und Microsoft.[56]

Das charakteristische Merkmal der Weiterentwicklung ist die Modularisierung des Webstandards in einzelne Teilaspekte. Dadurch ist eine effektive Arbeit an der CSS-Spezifikation möglich. Der wesentlichste Vorteil ist die mögliche Stück-für-Stück Implementierung schon fertiger oder fortgeschrittener Module in die Rendering-Engines der Webbrowser, ohne das auf andere Module, die noch mitten in der Entwicklung stecken, „gewartet" werden muß. Der Standard wird also nicht – so wie es bei den Vorgängerversionen üblich war – als eine komplette Empfehlung veröffentlicht, sondern in regelmäßigen Abständen sukzessive freigegeben.

Nachteile dieser Vorgehensweise sind eher organisatorischer Natur. Bei der Entwicklung muß gewährleistet werden, dass die einzelnen Module miteinander kompatibel sind. Dies wird gewährleistet durch die Arbeitsweise des W3C. Die Entwicklung der Teilmodule unterliegt ebenfalls dem Veröffentlichungsprozess des W3C.[57]

Insgesamt besteht CSS3 aus 51 Modulen, die von den dazugehörigen (Teil-)Arbeitsgruppen, den sogenannten „Advocates" bearbeitet werden.
Die Teilarbeitsgruppen bestehen meist aus 1-3 Mitgliedern.
Die einzelnen Module wurden nach Priorität kategorisiert. Die Prioritäten richten sich nach der momentanen Einfachheit der Implementation (softwareseitig), der Performancelastigkeit (Hardware) sowie der generellen Nachfrage nach einer bestimmten CSS-Eigenschaft. Alle Module entwickeln entweder CSS2.1 Merkmale weiter oder definieren ganz neue Aspekte.

[55] [W3C 004]
[56] [W3C 005]
[57] Vgl A1

## 3.2. Die Module von CSS3 - Dokumentation, Anwendung & Analyse

Im Folgenden stellt der Autor dieser Arbeit einen Großteil der Module der CSS Level3 Spezifikation vor und wird diese an Praxisbeispielen anwenden und auf Ihre Nutzbarkeit hin analysieren. Die Module hat der Autor hinsichtlich ihres höherstehenden Zwecks gruppiert. Anzumerken ist an dieser Stelle, dass der Fokus auf Layout und designspezifische Module sowie eventuelle Lösungen zu den in 2.6. geschilderten Problemen liegt. Module, deren Dokumentation noch zu unausgereift ist oder die sich nicht auf die Darstellung konzentrieren werden nicht oder nur sehr oberflächlich betrachtet.

### 3.2.1. Selektoren (Selectors Level3)

#### a) Attributselektoren

| Selektor | Funktion |
|---|---|
| `a[title^="Lesen"]` | Selektiert alle a-Elemente, deren „title“-Attribut mit „Lesen“ beginnt |
| `a[title*="Lesen"]` | Selektiert alle a-Elemente, in deren „title“-Attribut „Lesen“ enthalten ist (mindestens einmal) |
| `a[title$="weiter"]` | Selektiert alle a-Elemente, deren „title“-Attribut mit „weiter“ endet. |
| `a[title\|="Lesen"]` | Selektiert alle a-Elemente, deren „title“-Attribut mit „Lesen“ anfängt oder „Lesen“ ist. |

Tabelle1.: Neue Attributsselektoren in CSS3

Mit den neuen Attributselektoren sind umfangreichere Selektionsmöglichkeiten gegeben. Die Namensräume[58] , die eine auszuwählende Menge bestimmen, wurden erweitert. Insbesondere der Attributselektor `*=`, mit dem geprüft werden kann, ob ein Attributwert einen bestimmten String[59] enthält, verbessert die Möglichkeiten, auf bestimmte Merkmale des HTML-Inhalts Einfluss zu nehmen. So kann beispielsweise das `href`-Attribut eines `<a>`-Elements, wenn es auf eine bestimmte Domain zeigt (z.B. google.de spiegel.de) per CSS mit einem Icon versehen werden.

```
a[href*="spiegel.de"] { background: url(spiegel.png); }
a[href*="google.de"] { background: url(google.png); }
```

Q9: Praktische Anwendung der Attributselektoren

[58] [W3C 006]
[59] Zeichenkette

### b) Pseudoklassen

Neu bei der Selektion über Pseudoklassen sind die zusätzlichen strukturbasierten Pseudoklassen. Erstmalig ist es nun möglich, die im Dokument vorhandenen Elemente zu zählen und hinsichtlich ihrer Position innerhalb der Elementhierarchichie zu bestimmen, z.b. ein n-tes Kindelement oder eines n-tes Element seines Typs.

| Strukturbasierte Pseudoklassen | Funktion |
|---|---|
| `a:root` | Selektiert ein a-Element in der obersten Hierarchieebene des Dokumentes (innerhalb des `<HTML>`-Tags) |
| `a:nth-child(n)` | Selektiert ein a-Element, das das nte-Kindelement seines Elternelementes ist |
| `a:nth-last-child(n)` | Selektiert das a-Element, das das erste Element seines Typs im Dokument ist. |
| `a:nth-of-type(n)` | Selektiert ein a-Element, das das nte-Kindelement seines Elternelementes ist, vom letzten Element aus gezählt. |
| `a:last-child` | Selektiert ein a-Element, das das letzte Kindelement seines Elternelements ist |
| `a:first-of-type`<br>`a:last-of-type`<br>`a:only-of-type` | Selektiert das a-Element, das das erste / letzte/ einzige Element seines Typs im Dokument ist. |
| `a:empty` | Selektiert das a-Element, dass keine Kindelemente besitzt (auch keinen Text) |
| `a:not(x)` | Selektiert a-Element(e), die nicht x sind/haben. X steht hierbei für einen einfachen Selektor und kann Typselektor, Klassen/ID-Selektor, Attributselektor oder ein Pseudoelement/-Klassen-Selektor sein |

Tabelle2.: Eine Auswahl Strukturbasierter Pseudoklassen und ihre Funktion

`a:nth-child(n)` sowie `a:nth-of-type(n)` und ihre Variationen sind die mächtigsten neuen Pseudoklassen in der CSS Level 3 Spezifikation. Sie erlauben, bestimmte Elemente gezielt nach ihrer abgezählten Position anzusprechen.

```
ul li:nth-child(2) { ... } /* Auswahl des 2ten Kindelements */
ul li:nth-child(2n) { ... } /* alle geraden Kindelemente */
ul li:nth-child(2n+1) { ... } /* alle ungeraden Kindelemente */
ul li:nth-child(-n+3) { ... } /* die 3te, 2te, 1 Reihe */

input:not(.shortfield) { ... }
/* Alle input-Elemente, die nicht die Klasse shortfield besitzen */
ul li:not(:nth-child(2)) { ... }
/* Alle li-Elemente, die nicht gerade sind */
```

Q10: Praxisbeispiel nth-child und Not-Selektion

Doch nicht nur exakt gezählte Elemente sind selektierbar, es ist außerdem eine Mengenangabe möglich, so z.B. jedes gerade Element (2n) oder jedes ungerade Element (2n+1). Damit kann der sog. „Zebra"-Effekt1[60] ohne zusätzliche Klassen erzielt werden. Der Wert des Fragmentes `nth-child()` muß dabei stets größer 0 sein. Die ausschließende Selektion mit der Pseudoklasse `:not(x)` ermöglicht nun auch die in 2.6.3 a) beschriebene Außer-Selektion.

### c) Zustands-Pseudoklassen

Zustands-Pseudoklassen sind dynamisch und abhängig von einer Nutzerinteraktion [61] und können sich durch diese verändern.

| Zustands-Pseudoklassen (Auswahl) | Funktion |
|---|---|
| **:target** | Selektiert ein h1-Element, das Ziel eines URI-Links ist |
| **:checked** | Selektiert ein input-Element, das vom Nutzer ausgewählt wurde |
| `:required` | Selektiert Formular-Elemente, deren Eingabe Pflicht ist |
| `:optional` | Selektiert Formular-Elemente, deren Eingabe optional ist |

Tabelle 3: eine Auswahl der neuen Zustands-Pseudoklassen

Neue Zustands-Pseudoklassen betreffen insbesondere die Verarbeitung von Formularen, so kann über `:required` ein Formular-Pflichtfeld entsprechend formatiert werden. Im Zusammenspiel mit den Web-Forms 2.0 von HTML5[62] (Überprüfung von ein-

[60] abwechselnde Hintergründe bei Tabellen
[61] Klicken, Hover
[62] Vgl. A4

gegebenen Daten) lassen sich so nutzerfreundliche Formulare entwerfen.

### d) Pseudo-Elemente

Eine wesentliche Neuerung betrifft die Kennzeichnung der Pseudo-Elemente. Pseudo-Elemente, mit denen ein bestimmter Teil des Inhalts selektiert werden kann[63], werden fortan mit einem doppelten Doppelpunkt gekennzeichnet, um eine Unterscheidung zu Pseudoklassen zu ermöglichen.

| Pseudo-Elemente (Auswahl) | Funktion |
|---|---|
| `::value` | Selektiert nur den Wert eines Formular-Elements (value="...") |
| `::slot` | Template Layout Modul, selektiert Layoutbereiche |
| `::choices` | Selektiert die Auswahlmöglichkeiten von Checklisten oder einer Radiobutton-Gruppe |

Tabelle 4: Eine Auswahl neuer Pseude-Element-Selektoren

### e) Kombinatoren

Neu bei den Kombinatoren in CSS3 ist die „Tilde"-Kombination, auch bekannt als Nachbars-Selektion. Diese weist Ähnlichkeiten zur direkten Vorgänger-Kombination[64] auf, hat aber einen wesentlichen Unterschied: Das vorangestellte Element muss nicht der direkte Vorgänger sein, es können auch weitere Elemente dazwischen liegen.

```
h2 ~ p { ... }
```

Q11: Nachbars-Selektion über den Tilde-Kombinator

### f) Spezifität

Hinsichtlich der Spezifität ist der Not-Selektor[65] ein Ausnahmefall. Dies ist darin begründet, dass der Not-Selector immer einen anderen Selektor in sich trägt.
Diese Selektoren innerhalb des „Not-Selectors"[66] werden wiederum wie normal auftretende Selektoren hinsichtlich ihres Typs gewichtet und mitgezählt.[67]

[63] die sog. Fragmente

[64] z.B. h2 + p - Selektion eines p-Elementes, dem eine h2 direkt voran gestellt ist

[65] Vgl. 3.2.1 b)

[66] z.B. `a:not(#extlink)`

[67] Vgl. 2.3.2.

### 3.2.2. Values & Units (Werte und Maßeinheiten)

Das Values & Units – Modul enthält sämtliche Neuerungen, die Werte und Wertetypen betreffen.[68]

Neu ist die funktionale Notation `calc()`
Diese Funktion löst das Problem der „Berechnungen“[69] . Besonders einfach ist die Kombination von relativen Größenangaben und absoluten Größenangaben.
Mit der `calc`-Funktion ist nun folgende Notation möglich

```
.halb {
background:#000000;
width: calc(50% - 2*10px); /* 10px Padding für jede Seite =
20px */
padding:10px;
}
```

Q12: Die calc()-Funktion im Einsatz - exakt 50% breite Boxen

Das Abziehen der insgesamt 20px Innenabstand ist nun direkter Bestandteil der `width`-Eigenschaft. Die 10px, die für `padding` gesetzt sind (also insgesamt 20px), werden laut Box-Modell[70] wieder hinzu addiert, wodurch sich folgende einfache Rechnung ergibt:

**50%** **- 20px** /* gemäß `calc()` werden die 20px direkt subtrahiert */
**+ 20px** /* padding wird dazuaddiert (Box-Modell)

---

**== 50%** **Es entstehen immer genau 50% Gesamtbreite**

Weiterhin haben noch mehr Wertetypen ihren Weg in die Spezifikation gefunden, so sind jetzt Angaben in „`deg`“[71] möglich, um Winkel darzustellen.
Es wurden Farb/Transparenz-funktionen `rgba()` und `hsla()` eingeführt.
Der Autor wird diese neuen Wertetypen bei den entsprechenden Modulen näher erläutern.

---

[68] siehe A3 - Wertetypen von CSS
[69] Vgl. 2.6.7.
[70] Vgl. 2.3.5.
[71] engl. Degree - Grad

### 3.2.3. Die Layoutmodule

#### 3.2.3.1. Das erweiterte Box-Modell

Das Box-Modell[72] wurde in der neuen CSS3 Spezifikation um die Eigenschaft `box-sizing` erweitert. Mit dieser Eigenschaft kann die Art und Weise der Dimensionierung einer Box beeinflusst werden.[73] Die Eigenschaft kann die Werte `content-box` und `border-box` annehmen.

**content-box (Initialwert)** entspricht hierbei dem aus CSS2.1 bekannten Box-Model Die Angaben von `width` und `height` beziehen sich auf den Inhaltsbereich. Innenabstände und Umrahmungen werden diesem Bereich hinzuaddiert.

**border-box** Ist dieser Wert gesetzt, entsprechen die Angaben von `width` und `height` der Gesamtbreite **inklusive** den Angaben von Innenabständen und Umrahmungen. Die Dimensionen des Inhaltsbereiches ergeben sich damit aus der Subtraktion der Innenabstände und Umrahmungen von der Gesamtbreite:

Gesamtbreite = width
Gesamthöhe = height

Breite des Inhaltsbereiches = width – padding-left – padding-right – border-left – border-right
Höhe des Inhaltsbereiches = height – padding-top – padding-bottom – border-top – border-bottom

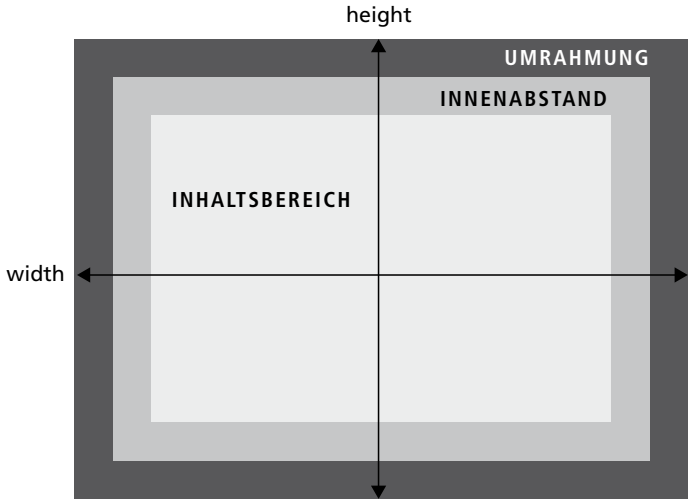



Abb. 9: Das „neue“ Box Modell mit der Eigenschaft box-sizing

[72] Vgl. 2.3.5.
[73] [W3C 007]

Durch diese elegante Lösung wird das Arbeiten mit Boxen deutlich erleichtert. Das in der CSS2.1.-Praxis lästige Errechnen der realen `width`- und `height`-Werte für Boxen mit Innenabständen oder/und Umrahmungen entfällt durch die Angabe von `border-box`. Die `box-sizing` Eigenschaft löst außerdem das in 2.6.7 beschriebene Berechnungs-Problem von  Boxen mit relativer Breite und festen Innenabständen:

Mit Angabe von `border-box` ist die Box immer 50% groß, egal wie groß der Innenabstand ist.

Das W3C-Team hat mit der `calc`-Funktion und `box-sizing` also gleich 2 Lösungen für das beschriebene Problem gefunden.

### 3.2.3.2. Flexible Box Layout

Das Flexible Box Layout Modul ist ein neues Layoutsystem, mit dem Boxen innerhalb des umschließenden Elternelementes positioniert, dimensioniert und aneinander ausgerichtet werden können. Dies geschieht bei Bedarf unabhängig vom eigentlichen Dokumentfluss[74] und der Reihenfolge der HTML-Elemente im Markup.
Das Modul bietet sich für die Darstellung mehrspaltiger Weblayouts an, die bisher nur umständlich zu bewerkstelligen waren.[75]

Ausgangspunkt für die Initialisierung eines Flexible Box Layouts ist ein Elternelement (`#flexwrap`), dem die Eigenschaft „`display`" mit dem Wert „`box`" gegeben wird.

```
<div  id="flexwrap">
      <div id="box1" class="flexbox">Box 1</div>
      <div id="box2" class="flexbox">Box 2</div>
      <div id="box3" class="flexbox">Box 3</div>
</div>
```

```
#flexwrap {
display: -webkit-box;
-webkit-box-orient: horizontal; /* Horizontale Orientierung
(Spalten) */
width:900px; /* Feste Breite der Elternbox */
}
```

Q12: HTML-Markup und zugehörige CSS-Anweisung für die Flexbox

Mit der `box-orient` Eigenschaft, wird angegeben, ob sich die Kindelemente horizontal oder vertikal orientieren. Im Folgenden wird nur die horizontale Verteilung begutachtet. Die folgenden Erklärungen sind aber ebenso zulässig für Boxen mit vertikaler Orientierung.

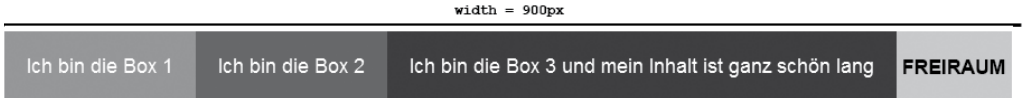



Abb10 - Standardverhalten von Flexbox (Vgl. Q12) -Die Kindelemente (.flexbox) werden bei horizontaler Box-Orientation nebeneinander positioniert

[74] Im Standard-HTML-Dokumentfluß werden im Quellcode nacheinander folgende Block-Elemente vertikal untereinander verteilt, es sei denn, sie besitzen die float-Eigenschaft oder sie werden mittels der postion-Eigenschaft aus eben diesem Fluß herausgelöst
[75] Vgl. 2.6.1. a) und b), Vgl. 2.6.2.

### a) Reihenfolge & Verteilung der Boxen

Mit der Eigenschaft `box-direction` kann die automatische Reihenfolge der Kindboxen definiert werden - `normal` oder `reverse`. Dies macht für internationale Layouts Sinn, die eine rechts-nach-links Leserichtung verwenden

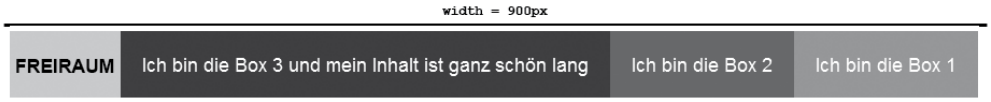



Abb. 12: `box-direction: reverse`

Die Eigenschaft `box-ordinal-group` bestimmt die manuelle Reihenfolge der Kindboxen. Je kleiner der Wert, desto weiter vorn steht die Box:


Abb. 13: Einsatz von `box-ordinal-group` um Kindboxen manuell zu sortieren box2 hat dabei einen Wert von 1, Box3 einen Wert von 2 und box1 einen Wert von3

### b) Ausrichtung und Anpassung der Boxen

Eine Standardeigenschaft von Flexible Box Layouts ist die automatische Höhenanpassung aller Kindboxen an die Höhe des Elternelements. Ist keine Höhe für das Elternelement gesetzt, sind alle Kindboxen so hoch wie das höchste Kindelement. Das Problem der gleichhohen Spalten[76] wird damit gelöst. Die Eigenschaft `box-align` bestimmt dieses Verhalten. Zulässig sind die Werte `start` (Ausrichtung an den oberen Kanten) `end` (Ausrichtung an den unteren Kanten) `center` (vertikale Zentrierung) `baseline` (Ausrichtung anhand der Schrift-Grundlinie) und `stretch` (Elemente „dehnen“ sich auf eine gemeinsame Höhe[77]).

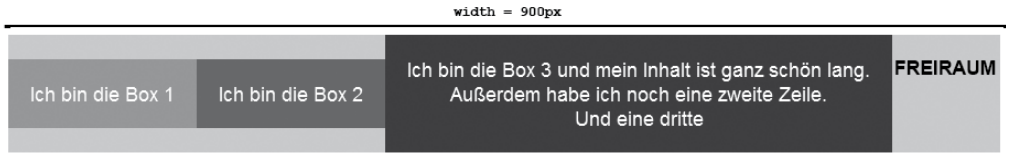



Abb. 14: vertikale Zentrierung aller Boxen mittels `box-align:center`

[76] Vgl. 2.6.1.
[77] Vgl. Abb. 13
[78] Vgl. 2.6.2.

### c) box-pack - Ausrichtung entlang der Achse

Eine weiter Möglichkeit der Ausrichtung von Kindboxen bietet die Eigenschaft „`box-pack`", mit der Kindboxen ENTLANG der bei box-orient angegebenen Achse ausgerichtet werden können, in dem vorliegenden Fall also die horizontale Achse. Zulässig sind die Werte `start` (Ausrichtung an den linken Seite) `end` (Ausrichtung an der rechten Seite) `center` (horitontale Zentrierung) und `justify` (Blocksatz - Verteilung von links nach rechts)


Abb. 15: horizontale Zentrierung aller Boxen mittels `box-pack:center`

### d) Ausdehnung und Flexibilität - Die Eigenschaft box-flex

Die horizontale Ausbreitung der jeweiligen Kindbox in dem vorliegenden Beispiel entspricht der Länge ihres Inhalts. Dadurch ist die dritte Box mit der längeren Textzeile natürlich wesentlich breiter als die ersten zwei Boxen. Außerdem entsteht dadurch rechts ein Freiraum[79], da das Elternelement 900px breit ist und die 3 Kindboxen diese Breite durch den insgesamt eher wenigen Inhalt nicht erreichen. Abhilfe schafft hier die Eigenschaft `box-flex`, die die Ausdehnung des Kindelements im Verhältnis zu ihrer Elternbox und den anderen Kindelementen angibt.

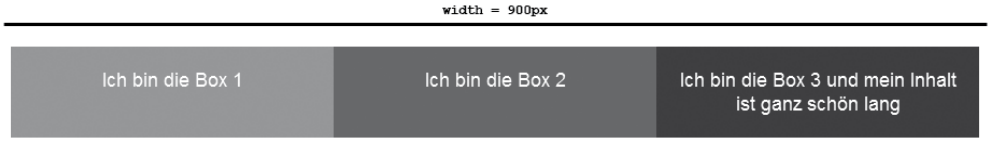



Abb. 16: `box-flex:1.0` spannt die Boxen gleichmäßig auf

Wird nun der Box3 ein `box-flex`-Wert von `4.0` gegeben, erhält diese einen vier Mal so großen Anteil am Freiraum als die anderen zwei Kindboxen.


Abb. 17: `box-flex:4.0` gibt der Box3 mehr Anteil am Freiraum

[79] Vgl. Abb10 - FREIRAUM

### e) Mehrzeilige Verteilung - box-lines

Standardmäßig werden alle Kindboxen auf einer einzigen horizontalen Linie verteilt[80]. Unter Umständen kann es aber sinnvoll sein, eine oder mehrere Kindboxen in eine neue horizontale Reihe zu bewegen. Dies geschieht mit der `box-lines` Eigenschaft, die dem Elternelement übergeben wird. Um mehrzeilige Boxen zu definieren, wird der Wert auf `multiple` gesetzt. Zum Zeitpunkt dieser Arbeit ist die Entwicklung dieser Eigenschaft leider noch nicht weiter fortgeschritten.

### f) Zwischenfazit

Das „Flexible Box Layout" löst klassisch auftretende Probleme auf einen Schlag[81] und leistet einen großen Schritt zur wirklich hundertprozentigen Trennung von Inhalt (HTML) und Design (CSS). Relativ unabhängig von der Reihenfolge der HTML-Elemente im globalen Dokumentfluss, können diese rein mit CSS-Eigenschaften untereinander positioniert, sortiert und aneinander ausgerichtet werden. Hervorzuheben ist hierbei, dass dies nur „untereinander" und „aneinander" funktioniert. Bei flexiblen Boxen entsteht ein eigener Fluss innerhalb des Elternelements, der zwar sehr flexibel anpassbar ist, aber trotzdem die Kindelemente miteinander in Verbindung setzt.[82] So kann z.B. eine dritte Box nicht komplett aus diesem Fluss herausgelöst werden und im Elternelement „unten rechts" platziert werden, während eine zweite Box „oben in die Mitte" gesetzt wird. Für diese Aufgaben hält CSS3 allerdings einige andere Layoutmodule wie das Template Layout Modul[83] bereit. Das Flexible Box Layout weist Ähnlichkeiten zum Verhalten von HTML-Tabellen zu Layoutzwecken auf und nimmt sich die Vorteile aus dieser veralteten Methode heraus[84]

| Flexible Box | <table> Layout |
|---|---|
| automatische Höhenanpassung der Kindelemente an die Höhe des Elternelement (bzw. das höchste Kindelement) durch box-align:stretch | Standardmäßig werden alle Höhen der Tabellenspalten (<td>) innerhalb einer Tabellenzeile (<tr>) an die höchsten Tabellenspalte angepasst |
| flexible Verteilung der Kindelemente innerhalb des kompletten Elternelements (box-flex und box-pack) | Standardmäßig werden alle Tabellenspalten gleichmäßig anhand ihrer Reihenfolge im Markup innerhalb der Tabelle verteilt. |
| flexible Möglichkeiten der Ausrichtung der Kindelemente zueinander (Oberkante, Unterkante, Mittelpunkt, Zeilenhöhe) | Durch das HTML-Attribut „valign" können die Inhalte der Tabellenspalten aneinander ausgerichtet werden. |

Tabelle5: Flexible Box Layout vs. <table>-Layout

[80] Bei vertikaler Box-Orientierung auf einer vertikalen Linie

[81] vertikale Zentrierung, gleiche Spaltenhöhen, Ausrichtung an Ober-/Unter-/Mittelkanten von nebenstehenden Elementen Vgl. Abb13-16

[82] z.B. bei der Errechnung des Freiraums für die box-flex-Eigenschaft, Vgl. Abb 16

[83] Siehe 3.2.3.4.

[84] Vgl. A2

### 3.2.3.3. Multi-Column-Layout

Mit den neuen Eigenschaften des Multi-Column Layouts ist es erstmals rein durch CSS möglich, Inhalte von HTML-Elementen automatisch in mehrere gleich große Spalten zu verteilen. Das HTML-Element, dessen Inhalte in Spalten eingeteilt werden, wird damit zu einer sog. „Spaltenbox". Den einzelnen Spalten dieser Spaltenbox selbst können keine individuellen Eigenschaften übergeben werden. Die Spalten haben immer die gleichen Breiten, Höhen und Abstände zueinander. Diese werden durch die im Folgenden erklärten Eigenschaften der umgebenden Spaltenbox definiert. Die Inhalte der Spaltenbox fließen automatisch von der linken zur rechten Spalte.[85]

#### a.) column-width Methode

Bei dieser Methode wird mit Hilfe der `column-width` Eigenschaft eine Spaltenmindestbreite festgelegt. Die daraus resultierende Spaltenanzahl ist je nach vorhandenem Platz[86] und dem zu verteilenden Inhalt variabel.

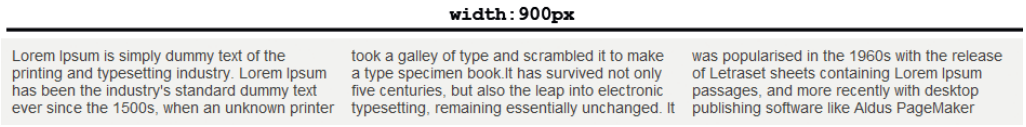



Abb. 18: Mit `column-width:300px;` wird der Inhalt eines 900px breiten div-Containers in 3 gleichgroße Spalten aufgeteilt.

Je schmaler die Gesamtbreite ist, desto weniger Spalten werden erstellt und das Element wächst in seiner Höhe, um seinen Inhalt gleichmäßig zu fassen. Ein auffälliges Verhalten bei dieser Methode ist die automatische Vergrößerung der Spaltenbreite, wenn die Gesamtbreite nicht (ohne Rest) in eine bestimmte Anzahl von Spalten mit `column-width`-Breite einteilbar ist.
Im nun vorgeführten Beispiel wird der Wert der Gesamtbreite des Multicol-Elements auf 800px gesetzt, die gewünschte `column-width` beträgt 300px. Laut Berechnung passen nur zwei jeweils 300px breite Spalten in das Element, und es würde theoretisch ein Freiraum von 200px entstehen. Das MultiCol-Modul verhält sich anders[87]: Jede Spalte wird nun automatisch um 100px vergrößert (die reale `column-width` ist also 400px), so dass die 800px Gesamtbreite gleichmäßig von zwei Spalten ausgenutzt werden. Dieses Verhalten weist Ähnlichkeiten zur `box-flex` Eigenschaft im Flexible Box Layout[88] auf.

[85] http://www.w3.org/TR/css3-multicol/#the-multicolumn-model
[86] bestimmt durch die Gesamtbreite oder, falls diese nicht angegeben ist, den Viewport
[87] Vgl. Abb. 19
[88] Vgl. 3.2.3.2 d)

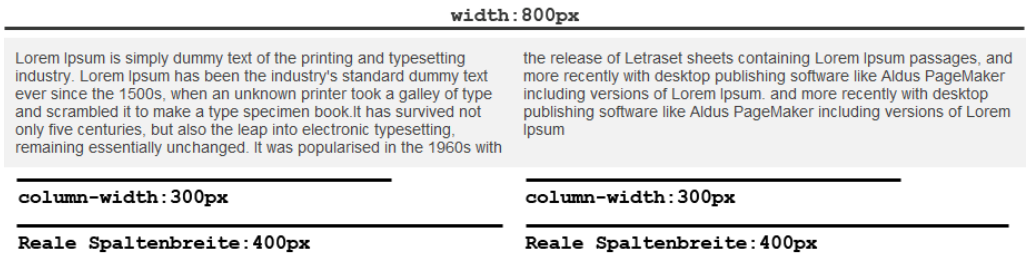



Abb 19: Vergrößerung der realen Spaltenbreite, wenn die Gesamtbreite nicht exakt in Spalten eingeteilt werden kann.

### b) column-count Methode

Mithilfe der `column-count`-Eigenschaft wird eine festgelegte Anzahl von Spalten definiert. Die resultierenden Spaltenbreiten variieren abhängig von der zur Verfügung stehenden Gesamtbreite des Elements bzw. des Viewports.

```
.multicol {
      -webkit-column-count:8;
}
```

Q13. CSS-Angabe der benötigten Spalten


Abb20.: `column-count: 8` Das Ergebnis

Bei der Kombination beider Methoden wird die column-count Angabe nur unter bestimmten Umständen einbezogen. Am wichtigsten ist immer die Angabe der Spaltenbreite. Auch hier gilt: Die Spaltenbreite wird nie kleiner als der angegebene Wert, nur größer.

### c) Zusätzliche Spalteneigenschaften

Mit der CSS-Eigenschaft `column-gap` kann der Abstand zwischen den einzelnen Spalten definiert werden. Mit der CSS-Eigenschaft `column-rule` wird eine Trennlinie 1 genau in der Mitte des bei `column-gap` 2 angegebenen Wertes platziert. Es können Breite, Art und Farbe der Trennlinie angegeben werden.

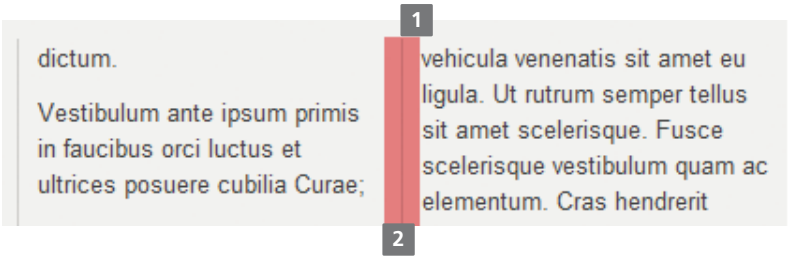



Abb21.: column-gap & column-rule

### e.) Fazit

Das Multi-Column-Layout bietet die Möglichkeit, die Vorteile von mehrspaltigen Layouts, wie sie in Druckerzeugnissen seit Jahrhunderten Gang und Gebe sind, nun auch ohne zusätzliches HTML-Markup bei Weblayouts zu nutzen. Die Vorteile liegen auf der Hand:

- effektivere Verteilung von Mengentext auf dem Medium
  (Folge: weniger Scrollen, kompakteres Layout)
- besseres Erfassen einen kompletten Textes durch den Nutzer
  (Verbesserung des sog. Querlesens)
- bessere Möglichkeiten der visuellen Strukturierung
  (visuelle Auflockerung bei inhaltsintensiven Webseiten)

Der wesentliche Vorteil liegt aber im Grundprinzip des Multicol-Moduls verankert: Der automatische Fluß des Inhaltes von der vorherigen zur nächsten Spalte sowie die Eigenschaften zur Steuerung der Spaltenumbrüche erleichtern den Einsatz in der Praxis massiv. Ein solches flexibles Verhalten bei mehrspaltigen Layouts war vorher nur durch das manuelle Eingreifen in das HTML-Markup[89] , durch Java-Scipt-Plugins[90] oder durch den Einsatz von PHP[91] möglich

[89] Manuelles Setzen des Inhalts, der in die nächste Spalte soll, in den jeweilige Spalten-Container
[90] Columnizer Jquery Plugin http://welcome.totheinter.net/columnizer-jquery-plugin/
[91] Nutzung der String-manipulations-Funktionen von PHP (insbesondere substr()) um die Text-inhalte anhand ihrer Zeichenzahl auf die jeweiligen Spalten-Container zu verteilen.

Diese Umwege fallen jetzt weg und nun können auch automatisch, oder durch ein CMS generierte Texte, ohne manuelles Eingreifen in ein ansprechendes Mehrspaltenlayout gesetzt werden.


Abb. 22: Das MultiCol Layout zur Formatierung von Artikeln

Weiterhin sollte angemerkt werden, dass es sich bei dem Multicol-Modul nicht unbedingt um ein komplettes Layoutsystem für komplette Seitenbereiche handelt, wie etwa dem Flexible Box Layout[92] oder dem Template Layout[93]. Das Ziel dieses Moduls ist das horizontale, gleichmäßige Verteilen der Inhalte (insbesondere Text) eines Containers in mehrere Spalten – nicht mehr, nicht weniger.

Blickt man auf die Internationalisierung von Inhalten, erkennt man einen großen Nachteil dieses Moduls: Die Richtung der Verteilung der Inhalte findet zur Zeit nur von links nach rechts statt. Bisher gibt es keine Möglichkeit dieses Verhalten zu beeinflussen. Insbesondere bei einigen arabischen oder asiatischen Sprachen, in denen von rechts nach links gelesen wird, wäre es sinnvoll gewesen, eine Eigenschaft `column-direction` einzuführen[94] mit der die Verteilung auch in die andere Richtung funktionieren würde. Für diese Sprachräume ist das Modul auf dem jetzigen Stand nahezu unbrauchbar.

[92] Vgl. 3.2.3.2.
[93] Vgl. 3.2.3.4.
[94] (Ähnl. Flexibl. Box Layout 3.2.3.2)

### 3.2.3.4. Das Template Layout

Mit Hilfe des Template Layout Modul können eigene Layout-Bereiche auf Basis einer CSS-Schablone[95] erstellt werden (sog. Slots). Durch die feste Benennung können HTML-Elemente in diese Layout-Bereiche beliebig verteilt und somit unabhängig voneinander und dem Dokumentfluß positioniert und dargestellt werden.

Das Template Layout führt erstmalig das vorlagenbasierte Positionieren in den CSS-Standard ein und ist eines der revolutionärsten neuen Module in der Spezifikation. Bislang wurde es von keiner Browser-Engine implementiert und die folgenden Beispiele können nur durch die Einbindung des Java-Script-Plugins „css-template“[96] angewandt werden. Leider hat auch das Plugin seine Grenzen und ist auf seine Basis-Anwendung reduziert, so ist die im Folgenden dokumentierte Dimensionierung der einzelnen Bereiche, das neue `::slot-`Pseudoelement, sowie die Verschachtelung von mehreren Templates nicht anwendbar.

#### a) Die Schablone – Erstellen des Templates

Um ein Template Layout zu initialisieren, wird dem Container-Element unter Zuhilfenahme der `display`-Eigenschaft eine Schablone 1 verordnet

```
.template {
      display:    "a  b"   1
                  "c  d"
 }
```

Q14: CSS-Deklaration der Schablone über die `display` Eigenschaft

Die Schablone 1 ist eine reduzierte Veranschaulichung des Komplett-Layouts in Form einer Matrix[97]. In diesem Beispiel ist die obige Schablonen-Deklaration wie folgt zu verstehen:

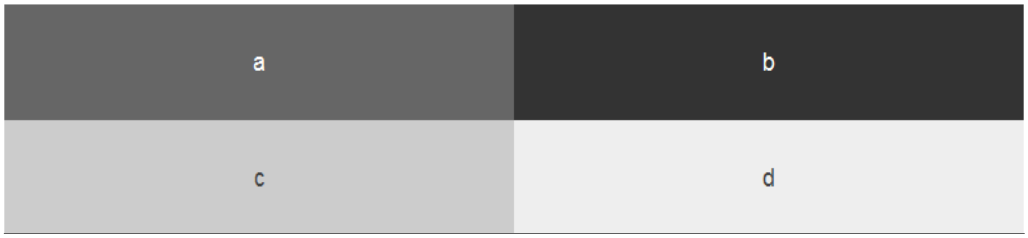



Abb. 23: Die Anwendung der Schablone aus Q14

[95] Vgl. Q14
[96] http://code.google.com/p/css-template-layout/ Dieses Plugin parst die neuen CSS-Eigenschaften und gibt diese der Spezifikation entsprechend aus.
[97] die Anordnung von Zahlenwerten oder anderen Objekten in Tabellenform

Jeder neue Kleinbuchstabe in Q14 steht für einen neuen Layoutbereich. In dem vorliegenden Beispiel existieren also 4 Layoutbereiche `a b c d`, auf die die HTML-Elemente später verteilt und positioniert werden.
Jede in " " eingeschlossene Zeichenkette entspricht einer komplette Zeile des Layouts – im vorliegenden Beispiel existieren also 2 Zeilen: „`a b`“ und „`c d`“
Jedes Zeichen in dieser Zeichenkette repräsentiert außerdem die Spalte, in der es steht. [98] - Im vorliegenden Beispiel exisistieren also 2 Spalten: „`a c`“ sowie „`b d`“.

Die Layoutbereiche stellen immer einen rechteckigen Block dar, allerdings können einzelne Layoutbereiche auch mehrere Spalten überspannen[99]

```
display:    "a  a  b"
            "c  .  d"
```

Q15: CSS-Code für Abb 24

Das Ergebnis gestaltet sich wie folgt: Der Layoutbereich a überspannt wie in Q15 angegeben den c-Bereich sowie den Leerraum


Abb24: Layoutbereich `a` überspannt 2 Spalten - `c` und einen Leeraum

[98] a steht über b und beide repräsentieren die erste Spalte, b steht über c und repräsentieren die zweite Spalte
[99] Vgl Abb24

### b) Dimensionierung der Layoutbereiche

Jede Spalte und jede Zeile der Schablonen-Matrix kann beliebige Breiten bzw Höhen annehmen. Der Höhen-Wert wird hinter der entsprechenden Zeile 1, der Breiten-Wert wird unter der entsprechenden Spalte 2 angegegeben[100]

```
.template {
      width:      800px
      display:    "a    a     b"    /75px   1
                  "c    .     d"    /150px  1
                  200px 400px 200px;
                  2     2     2
      }
```

Q16: Dimensionangaben in der Template Layout Schablone

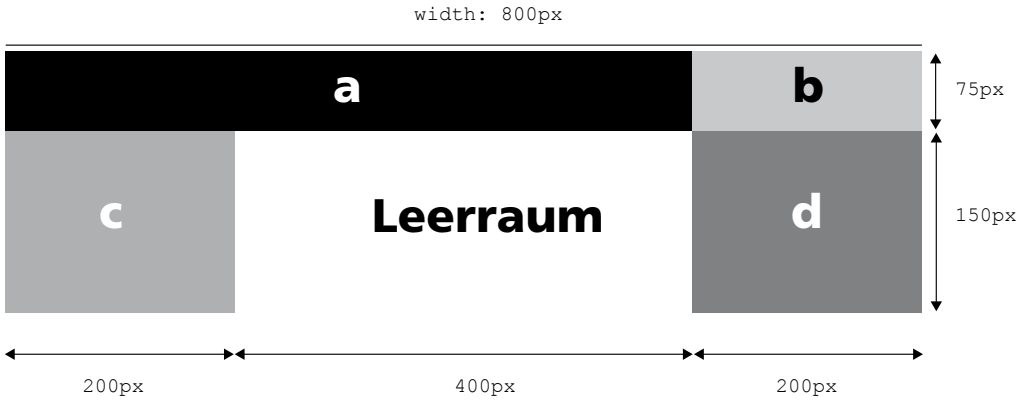



Abb. 25: Visualisierung der Layoutbereiche aus der CSS-Schablone (Vgl. Q16) inkl. ihrer Dimensionsangaben

### c) Belegung der Layoutbereiche

Mit Hilfe der erweiterten `position`-Eigenschaft können nun HTML-Elemente, die sich in dem Template-Container `.template` befinden, den in Q16 angelegten Layoutbereichen frei zugewiesen werden. Die Elementen werden über die Syntax `position: kleinbuchstabe_des_layoutbereiches` aus dem normalen Dokumentfluss herausgerissen und in den gewünschten Bereich platziert.

[100] Die Wertetypen entsprechen dabei den Wertetypen der width/height-Eigenschaften

Als Schablone wird der Code aus Q16 verwendet. Für die zuzuteilenden Elemente wird folgendes HTML-Markup benutzt:

```
<div class="template">
      <div id="box1">Ich bin #Box 1<br />im Bereich d</div>
      <div id="box2">Ich bin #Box 2<br />im Bereich c</div>
      <div id="box3">Ich bin #Box 3<br />im Bereich b</div>
      <div id="box4">Ich bin #Box 4<br />im Bereich a</div>

</div>
```

Q17: HTML-Markup der zuzuteilenden Elemente

Rein css-basiert werden die Boxen nun den jeweiligen Bereichen - wie in einem Setzbaukasten - zugeteilt:

```
#box1 { position: d; }
#box2 { position: c; }
#box3 { position: b; }
#box4 { position: a; }
```

Q18: Zuweisung der Boxen in die Layoutbereiche über die `position`-Eigenschaft

Das Ergebnis gestaltet sich wie folgt:

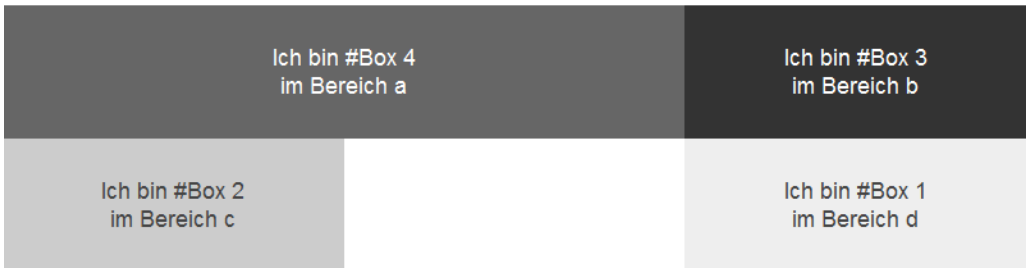



Abb26.: Die Boxen werden auf die Layoutbereiche verteilt

#box1 wird auf den Bereich d angewandt, box#2 in den Bereich c, #box3 in den Bereich b. #box4 verteilt sich in dem Bereich a und überspannt, wie in Q16 angegeben, zwei Spalten.

### d.) ::slot() Pseudo-Element

Die Bereiche des Templates können zusätzlich über die neue ::slot-Pseudoklasse adressiert und mit vorgegebenen CSS-Eigenschaften versehen werden. Das Pseudo-Element kann nur direkt auf den Template-Container angewandt werden. Der Sinn dieses Elements liegt darin, eine übergeordnete Möglichkeit zu besitzen, um den gesamten Slot anpassen zu können, falls sich mehrere Elemente den Slot teilen.
So kann z.B. eine Hintergrundfarbe für den gesamten Slot gegeben werden, in dem die Elemente liegen.[101] Aus diesem Grund sind auch die CSS-Eigenschaften begrenzt und eher zusammenfassender Natur.

```
.template::slot(a) {
      background: #ff00;
}
```

Q19: Dem Layoutbereich `a` wird eine rote Hintergrundfarbe zugewiesen

### e.) Templates für Seitenbasierte Medien

Die Eigenschaften des Template Layouts sind geeignet für die Verwendung in seitenbasierten Medien (Ebooks, Druckversion einer Website, generell: PDF-Alternative). Mit dem Medienselektor `@page`[102] wird den Seiten ein Template zugewiesen.

### f.) Verschachtelung mehrerer Templates

Es ist außerdem möglich, mehrere Template-Layouts ineinander zu verschachteln, so z.B. ein umgebenden Container mit 2 Spalten, der in seiner rechten Spalte einen weiteren Template-Container aufzieht, in dem ein weitere Schablone angewandt wird.

### g) Fazit

Der fundamentale Fortschritt des Template Layout Moduls ist die hunderprozentige Trennung von Inhalt (HTML) und Layout (CSS). Für die Anwendung spielt es keine Rolle mehr, wo ein Element im HTML-Dokumentfluss erscheint[103].
Durch die flexible Zuweisung in die vorher definierten Layoutbereiche, können Elemente nach Belieben positioniert werden. Es liefert eine übersichtliche und zeitsparende Variante, komplexe Layouts zu erstellen. Dadurch, dass bei der Dimensionierung natürlich auch relative Wertetypen einsetzbar sind, tragen sie eine Menge zu der Philosophie

[101] Vgl. Q19
[102] Vgl. 3.2.7.2.
[103] Es muß natürlich innerhalb des Template-Containers auftreten

der sog. Liquid Layouts[104] bei. Es ist nun möglich, ein auf horizontaler und vertikaler Achse komplett skalierbares Design umzusetzen, dass aber trotzdem auf einem „Baukastenprinzip" beruht und die einzelnen Elemente zusammenhält.

Ähnlich wie das Flexible Box Layout bringt auch das Template Layout durch die Spalten- und Zeilenwirkung die Vorteile einen `<table>`-Layouts in Spiel[105] .
Der entscheidende Unterschied zu Flexbox liegt aber darin, dass die positionierten Kind-Elemente keinen zusammengehörigen Fluß mehr bilden. Die Matrix-Schablone mit den zugehörigen Dimensionen definiert das Layout und die Kindelemente beziehen sich nicht aufeinander wie beim Flexbox-Modul.
Durch die Möglichkeiten der vertikalen Ausrichtung über das ::slot(Pseudoelemente) können die Elemente in ihrer Gesamtheit vertikal zentriert werden, auch die automatische Höhenanpassung nebeneinander stehender Spalten[106] , stellt kein Problem mehr dar.
Trotzdem deuten sich einige Nachteile an. Das Definieren der Schablonen-Matrix ist untypisch für CSS-Anweisungen, denn der Wert also solches steht in seiner Gesamtheit für eine Verbildlichung des Gesamt-Layouts. Dies ist insofern untypisch, da die CSS-Werte sonst eher Maße, Funktionen oder Schlüsselwörter darstellen[107] Um Übersicht und geringen Wartungsaufwand zu gewährleisten, ist es sinnvoll, die Schablone immer in einer ähnlichen Form, wie in den obigen Beispielen aufgezeigt, anzugeben – Laut Spezifikation ist dies allerdings keine Pflicht . Eine Schablonen-Definition in Form von: `display: "a.b" ".c." "d.."` ist genauso zulässig.
Allerdings treten Notations-Probleme auf, will man die Zeilen und Spalten dimensionieren.[108] Bei einer einzeiligen bzw gekürzten Notation der Layoutbereiche, gibt es keine Möglichkeit, die individuellen Dimensionen sinnvoll zu notieren, insbesondere die Spaltenbreiten, die immer als letztes unter der jeweiligen Spalte auftauchen sind auf die verbildlichende Darstellung in Matrixform angewiesen. Es müsste ein zusätzliches Zeichen eingeführt werden, was beim Parsen der Schablone auf eine Spaltenbreitendefinition hinweist, z.B. ein Komma. Die Schablonenmatrix ist konzeptionell sehr durchdacht, allerdings fordert sie durch solche Ungereimtheiten Implementationsschwierigkeiten heraus, scheinbar einer der Gründe, warum das „Template Layout Modul" noch nicht implementiert wurde.
Weiterhin können zur Zeit nur Layout-Bereiche mit rechtwinkligen Ausmaßen definiert werden. Hier würde sich die Chance ergeben, auch indivuellere Formen zu definieren, wie z.B. ein L-förmiger Layoutbereich.
Die Autoren des Moduls schließen diese Option für die Zukunft aber nicht aus[109]

---

[104] „Flüssige Layouts" - CSS-Layouts, die sich durch Verwendung von relativen Maßangaben, den Gegebenheiten des Users anpassen, so z.B. Anpassung der Breiten an die Auflösung
[105] Vgl. Tabelle5
[106] Vgl. 2.6.1. b)
[107] siehe A3 „Wertetypen in CSS"
[108] Vgl. 3.2.3.4. b)
[109] [W3C 009]

### 3.2.3.5. Grid Positioning

Dieses Modul ermöglicht das Erstellen von CSS-Layouts auf Basis eines Gestaltungsrasters (Layout-Grid) Ein Gestaltungsraster teilt Layouts mittels vertikalen und horiontalen Linien in eine geordnete Struktur – es ist eine unsichtbare geometrische Orientierungshilfe.[110] Das Ausrichten und die Dimensionierung von Elementen, erfolgt anhand den Linien dieses Rasters. Ein Grid trägt damit zur Übersichtlichkeit eines Designs bei.
Auch dieses Modul wurde noch nicht implementiert.

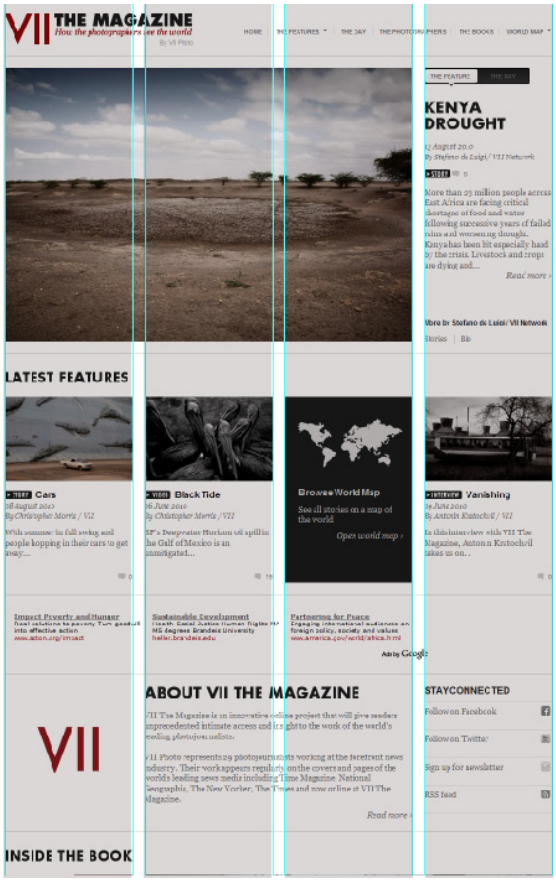



Abb27: Typisches Layout-Grid bei Webseiten http://magazine.viiphoto.com. Das Raster gibt vor, wie sich Bereiche aneinander ordnen und ausrichten und welche Größen sie einnehmen.

Das Modul führt zu diesem Zweck die neue Maßeinheit „`gr`" (Grid Unit- eine vom Raster abhängige Maßeinheit) ein, mit der Elemente anhand der Linien des Grids ausgerichtet werden.
Das Modul wird mit den `columns` und `coloumn-gap`-Eigenschaften des Multi-Column-Layout[111] kombiniert und erweitert diese mit den eigenen Eigenschaften.

[110] Vgl. Abb 27
[111] Vgl. 3.2.3.3.

### a) Erstellen des Layout-Grids

Mit den Eigenschaften des Multi-Column Layout Modul werden zunächst die 3 Hauptspalten definiert, die einen Abstand von 12px zueinander haben. In diesen 3 Haupt-Spalten befindet sich später der jeweilige Inhalt:[112]

```
body { columns:3; column-gap:12px; }
```

Q20: Definition des 3spaltigen Grundlayouts mit den Eigenschaften des MultiCol-Moduls

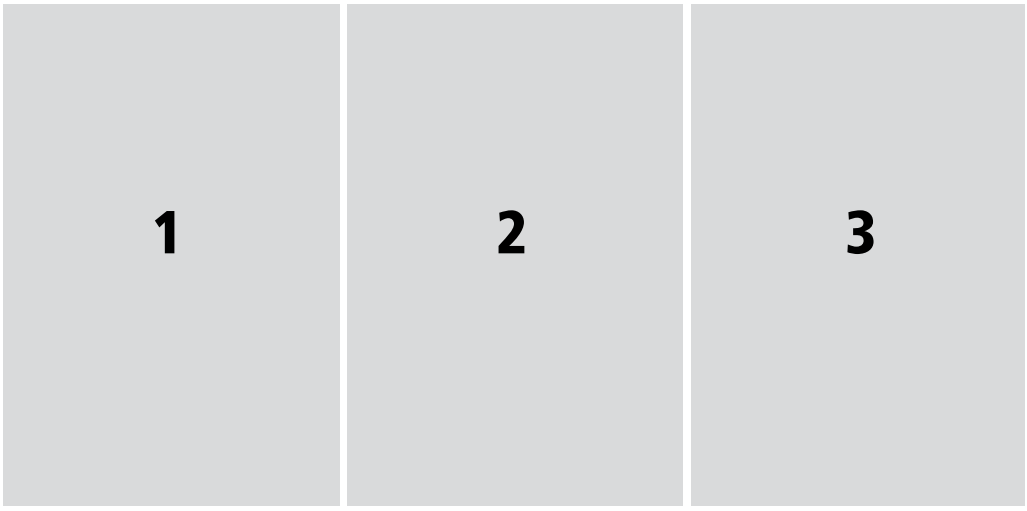



Abb. 28: Initialisierung des 3 spaltigen Multi Column Elements

[112] Das folgende Beispiel ist [HAY] entlehnt

Mit den Eigenschaften `grid-columns` und `grid-rows` wird nun ein feineres Raster unter diese 3 groben Inhalts-Spalten gelegt.

```
body {
                    1       2       3
      grid-columns: 70px (12px 70px) [11];
      grid-rows: 170px *;
      columns:3; column-gap:12px;
      }
```

Q21: Anlegen des Layout-Grids. Insgesamt 12 (70 + 12px) Spalten und zwei 170px + * Zeilen

Der erste Wert der `grid-columns` Eigenschaft 1 steht für die Dimension der ersten Spalte. Die 2 Werte in Klammern 2 stehen für die Dimensionen aller folgenden Spaltenpaare. Diese zwei Werte werden bis zum Ende des Containers abwechselnd wiederholt. Mit dem Wert in den eckigen Klammern 3 kann die Anzahl dieser Wiederholung auf eine bestimmte Anzahl begrenzt werden.

Der obige `grid-columns`-Wert ist also wie folgt zu lesen: Ziehe eine Spalte mit 70px auf und wiederhole danach 11mal die in Klammern stehende Spalten-Paarung.Diese Art der Schreibweise kann ebenfalls für die `grid-rows` gesetzt werden[113]

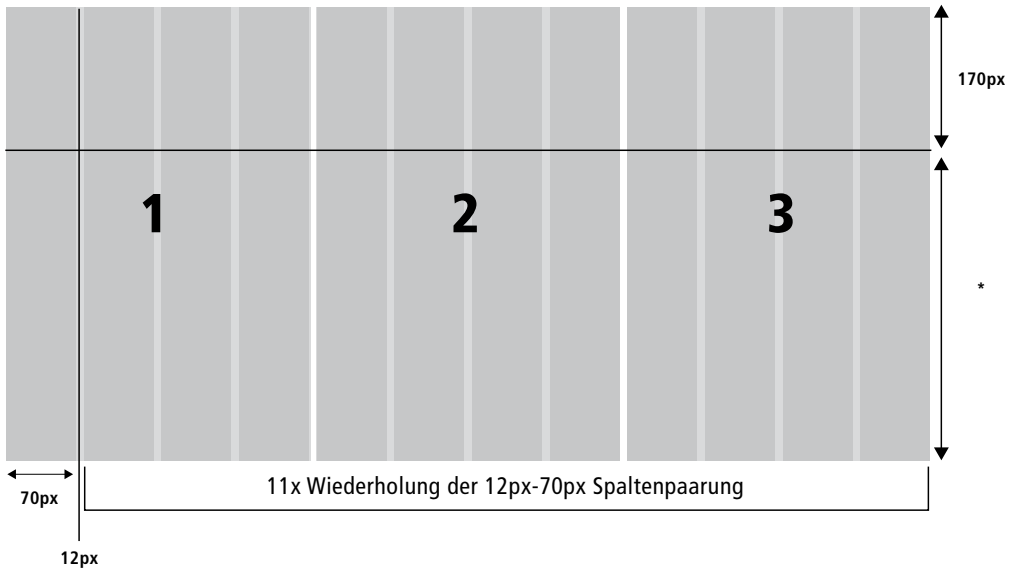



Abb. 29: Visualisierung des in Q21 angelegten Grid

[113] nur dass dort die px-Werte für die Höhen der Zeilen stehen

Der `grid-rows` Wert verwendet in dem Beispiel eine andere Schreibweise:
Die erste Zeile hat eine Höhe von 170px, Mit dem *-Wert wird der zweiten Zeile einfach der restliche Platz zugewiesen.

**b) Die Grid Unit**

Die Vorteile dieses Grid-Layouts kommen durch die neue Maßeinheit `gr` zum Vorschein. Mit dieser ist es möglich, sämtliche Positions- und Dimensions-Angaben der Inhalte des Spaltencontainers auf das soeben aufgezogene Raster und dessen vertikale und horizontale Linien zu beziehen – die Elemente orientieren sich mit Hilfe der `gr` ausschließlich am Raster.

- `gr` steht dabei für die Distanz von einer Grid-Linie zur Nächsten.
- `1 gr` umfasst damit eine Grid-Spalte bzw. Grid-Zeile
- `3 Gr` würde 3 Spalten/Zeile umfassen, wobei zu beachten ist, dass die Spaltenlücke zwischen den Spalten[114] ebenfall als Spalte mitgezählt wird.
- Die Zahlen sind dabei Fließkommawerte, also können auch 1.5 gr angegebenen werden.

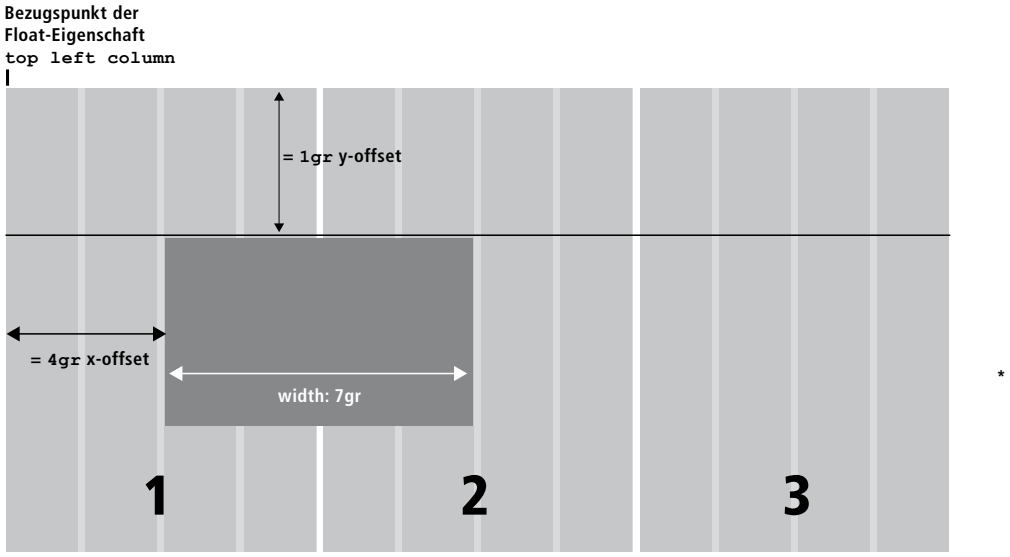



Abb. 30: Die Grid Unit im Einsatz, Positionierung eines Blockelements im Layout-Grid

Im Beispiel wird ein Blockelement innerhalb des Rasters positioniert. Ihm wird ein `float` vergeben, so dass andere Elemente (insbesondere Texte) es umfließen können.
Der `float`-Eigenschaft wird im Rahmen des Grid Layouts in CSS3 eine erweitere Wertedefinition vergegeben. Float- funktioniert dabei ausgehend von einem Bezugspunkt (sieh Abb 30.) der mit einer dreigliedrigen Werteangabe, bestehend aus x-Position,

[114] in dem Beispiel also die 12px Lücke

y-Position und bezugnehmendem Element, angegeben wird. Im obigen Beispiel fließt das Block-Element also ausgehend von der oberen linken Spalte - `top left column`

Mit der neuen Eigenschaft `float-offset` kann das Element anhand der Grid-Linien verschoben und positioniert werden, im Beispiel `4gr` (4 Grid-Spalten) von links und `1gr` (1 Zeile) von oben.

Die `float`-Eigenschaft spielt bei Grid Layouts eine große Rolle, da damit vorher unmögliche Verhalten in die CSS-Praxis umgesetzt werden können, wie z.B. das linke und rechte Umfließen eines Bildes durch Text.

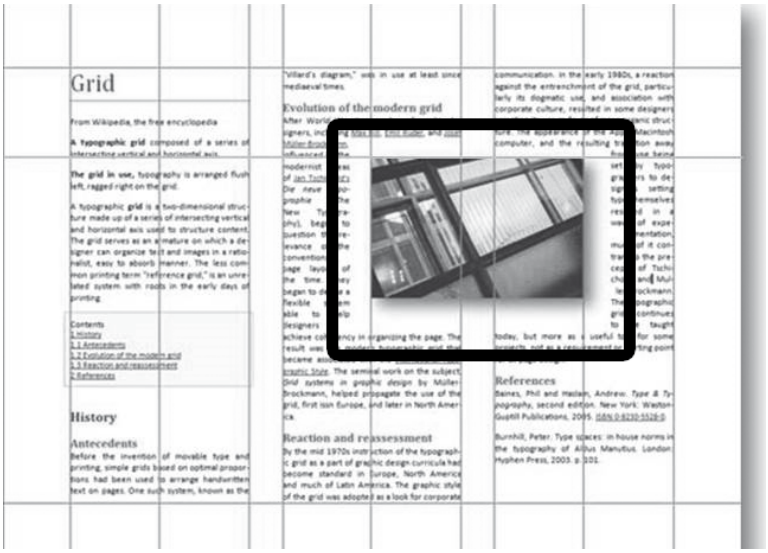



Abb. 31 - linkes und rechtes Umfließen eines mit float-offset positioniertem Bild[115]

Grid Positioning wird ein extrem flexibles Modul für dynamische Layouts. Elemente, die die neue `gr` Maßeinheit für ihre Dimensionen und ihre Positionierung verwenden, verhalten sich immer relativ zum Grid. Will man Layoutanpassungen vornehmen, reicht es, die Abmaße des Grids zu verändern. Die Elemente passen sich automatisch den Gegebenheiten an. Die Vorteile, die das Multicol-Modul von Haus aus schon mitbringt[116] sowie die neue Float-Positionierung und Float-Verschiebung[117] innerhalb des Grids machen dieses Modul gerade für Liquid Layouts sehr interessant.

Das Layout wächst und schrumpft mit den Gegebenheiten und passt sich optimal an Auslösung und Medientypus an.

[115] [W3C 010] http://www.w3.org/TR/css3-grid/
[116] wie u.A. das automatische Fließen von einer Spalte zur Nächsten
[117] `float-offset`

Werden für die Werte der Grid-Definition[118] ebenfalls relative Werte (%) genommen, entsteht ein magnetisches Layout, dass zwar dehnbar ist und sich auf verschiedene Auflösungen anpasst, trotzdem aber die Relationen der Elemente zueinander behält und Inhalte natürlich fließen lässt – ähnlich wie ein dehnbares Netz oder ein Akkordion.

Hervorzuheben ist der wesentliche Unterschied zum Template Layout Module:
Mit dem Einsatz des GP Moduls entstehen keine spezifisch zuweisbaren Bereiche (Zellen), in die die Elemente baukastenartig „eingesetzt" werden.[119]
Das GP Modul liefert viel mehr die Möglichkeit, flexibel mit dem Grid umzugehen und beispielsweise einzelne Elemente über mehrere Spalten/Zeilen oder zwischen zwei Spalten/Zeilen aufzuspannen. Es nutzt das geschaffene Grid als ordnungschaffende Orientierunghilfe, nicht aber als fixe tabellenartige Layoutstruktur, aus deren Zellen es kein „Ausbrechen" gibt. Dies gibt Designern eine Freiheit, die sonst nur mit professionellen Satzprogrammen wie Adobe Indesign, ermöglicht wird.

Wird das Modul so implementiert, ist seine Verwendung perfekt geeignet für Layouts, die sich aus generiertem Inhalt zusammensetzen und vor allem auch für aufwändige Artikel-layouts, die aus dem typisch „blockigen" rechtwinkligen Ansatz, der bei der Inhaltsgestaltung mit CSS oft zu finden ist, ausbrechen wollen.

Doch gerade die Implementierung scheint, wie auch beim Template Layout, ein Problem darzustellen – Der Working Draft, der hier vorgestellt wurde, stammt vom Mai 2007 und bisher gab es noch keine funktionierende Version in den modernen Browser-Engines. Alle in diesem Absatz vorgestellten Eigenschaften sind also nur theoretischer bzw konzeptioneller Natur.

[118] Vgl. Q21
[119] Vgl 3.2.3.4

## 3.2.4. Grafische & Dekorative Eigenschaften

### 3.2.4.1. Hintergrundeigenschaften

Alle neuen Eigenschaften und Werteangaben sind Teil des „Backgrounds & Borders" Modul[120], das zur Zeit als Working Draft in der letzten Version vom April 2010 besteht. Die Hintergrundeigenschaften von CSS bestimmen, welche Hintergrundfarben und Bilder für HTML-Elemente verwendet werden und wie diese positioniert, wiederholt und skaliert werden.

**a.) background-image:**
In CSS3 ist es erstmalig möglich, einem Element mehrere Hintergrundbilder zu vergeben. Multiple Bilder werden durch Komma getrennt hintereinander in der „`background-image`"-Eigenschaft[121] angegeben. Dabei ist das Bild, dass weiter vorne in der Deklaration erscheint, auch visuell den Anderen übergeordnet:

[120] [W3C 011] http://www.w3.org/TR/css3-background/
[121] oder der `background` Kurzschreibweise

```
.table {
      background-image: url(loeffel.png),
                        url(teller.png),
                        url(decke.gif);
}
```

Q22 Multiple Angabe von Hintergrundbildern

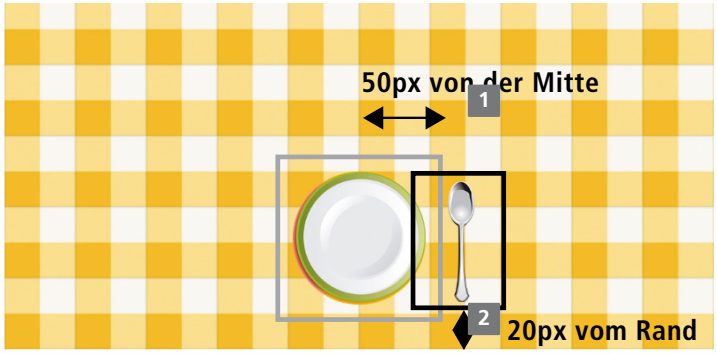



Abb32. Multiple Hintergründe in der Praxis (Q22)

### b.) background-position:

Diese Form der Angabe von multiplen Wertepaaren wiederholt sich auch für die anderen BG-Eigenschaften, wie `background-repeat` und `background-position`

```
background-position:   center 50px bottom 20px, 1 2
                       center 80%,
                       right top;
background-repeat:     no-repeat, no-repeat, repeat;
```

Die tischdecke.gif ist das unterste Hintergrund-Element und wiederholt sich endlos (repeat). Darüber liegt der Teller (PNG), der zentriert wird und sich auf 80% der Höhe befindet.

Der Löffel (loeffel.png) verwendet eine neue (optionale) Form der vierteiligen Positionsangabe, bei der es möglich ist, eine zusätzliche Verschiebung von der angegebenen Position einzusetzen, sofern die Position mit den Schlüsselwörtern `top, bottom, left, right, center` definiert wurde.

Die Angabe background-position: center 50px bottom 20px bedeutet also:

- horizontale Zentrierung (center) mit Verschiebung um 50px nach rechts. 1
- Vertikale Positionierung an den unteren Rand (bottom) mit Verschiebung um 20px nach oben. 2

Positive Werte stellen dabei immer eine nach innen erfolgende Verschiebung dar.[122] Negative Werte definieren eine Verschiebung nach außen.[123]

**c.) background-repeat**

Für diese Eigenschaften wurden die neuen Schlüsselwörter `space` und `round` eingeführt.

Mit `space` wird das Bild so oft wiederholt, wie es vollständig, ohne angeschnitten zu werden, in den Raum der Hintergrundpositionierung hineinpasst.
Mit `round` wird das Hintergrundbild solange verkleinert, bis es in vollständigen Wiederholungen in den Raum der Hintergrundpositionierung passt.

**e.) background-origin**

Die neue Eigenschaft definiert die äußeren Ränder der `background-position` Eigenschaft

Zulässige Werte sind:

- `border-box` Die äußeren Grenzen orientieren sich an den Umrahmungen (`border-`Eigenschaft)
- `padding-box` (initial) Die äußeren Grenzen orientieren sich an den Innenabständen (padding-Eigenschaft)
- `content-box` Die äußeren Grenzen orientieren sich am Inhalt (Inhaltsbereich)[124]

**f) background-clip:**

Die neue Eigenschaft definiert die gesamte Fläche, in der sich die Hintergründe befinden dürfen – sozusagen die Zeichenfläche des Hintergrunds[125] .

Zulässige Werte sind

- `border-box` (Die Hintergrundfläche erstreckt sich über die gesamte Box inkl. Umrahmungen)
- `padding-box` (Die Hintergrundfläche erstreckt sich bis zu den Grenzen der padding-Werte)
- `content-box` (Die Hintergrundfläche erstreckt sich bis zu den Grenzen des Inhaltsbereiches )

---

[122] bei `right` in die linke Richtung, bei `bottom` nach oben.
[123] [W3C 011] http://www.w3.org/TR/css3-background/
[124] Vgl. 2.3.5. Box Modell
[125] [W3C 011] Body Painting Area

**g) background-size:**

Mit der neuen Eigenschaft lassen sich nun auch Hintergrundbilder auf flexible Weise skalieren. Neben den Angaben mit verschiedenen relativen und festen Maßeinheiten erlaubt die Eigenschaft eine automatisierte Anpassung der Hintergrundbilder auf die Dimensionen der Box mit den Schlüsselworten „`contain`" und „`cover`".


`background-size: contain;`

`background-size: cover;`

Abb. 33.: Hintergrundbildskalierung - contain vs. cover

Mit dem Schlüsselwort `contain` wird das Hintergrundbild – unter Beibehalten des Größenverhältnis - soweit vergrößert, bis eine Seite des Bildes eine Seite der umschließenden Box komplett einnimmt. Das Bild wird immer vollständig angezeigt, es entsteht u.U. Freiraum. Ziel ist es, das komplette Hintergrundbild in die Maße der Box einzupassen und vollständig anzuzeigen, auch wenn dadurch Freiraum entsteht.

Mit dem Schlüsselwort `cover` wird das Hintergrundbild – unter Beibehalten des Größenverhältnis – soweit vergrößert, bis es die komplette Box überdeckt, dadurch wird das Bild natürlich angeschnitten, und ist nur teilweise zu sehen. Ziel ist es, die Box komplett zu überdecken, aber nicht unbedingt das komplette Hintergrundbild anzuzeigen.

Neben dieser automatisch generierten Anpassung, gibt es noch die Möglichkeit, feste und relative Maße für die background-size zu verwenden.

```
background-size: 400px 300px
```

skaliert das Bild auf 400px Breite und 300px Höhe.

### 3.2.4.2. Borders

Auch hinsichtlich der Rahmeneigenschaften hat sich in Level 3 einiges getan.
So sind jetzt nicht nur abgerundete Ecken möglich, sondern auch Hintergrundbilder für Rahmen.

#### a.) border-radius

Mit den neuen Eigenschaften `border-top-left-radius, border-top-right-radius, border-bottom-right-radius` und `border-bottom-left-radius` ist nun die Radius-Angabe für abgerundete Ecken für Boxen möglich. Jede Ecke wird mithilfe einer vertikalen und horizontalen Radiusangabe definiert. So kann die Form der runden Ecke verändert werden.

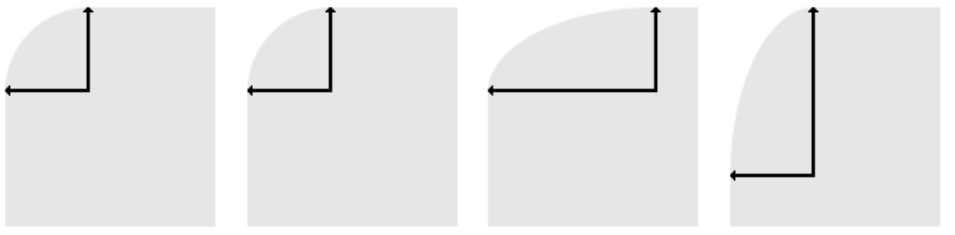



Abb. 34: border-radius[126]

Auch hier gibt es die Kurzschreibweise mittels der Eigenschaft `border-radius`. Hierbei werden zu erst die horizontalen Radien aller Ecken im Uhrzeigersinn beginnend von der linken oberen Ecke angegeben. Danach werden die vertikalen Radien im Uhrzeigersinn beginnend von der linken oberen Ecke mit vorangestelltem Schrägstrich angegeben.

```
border-radius: 50px 10px 10px 50px / 50px 10px 10px 50px;
```

Q23: border-radius Angabe in Kurzform

[126] entnommem [W3C 012]

Die abgerundeten Ecken schneiden den innenliegenden Inhalt ab. Sie geben dem Inhalt gewissermaßen eine Maske, so dass er nicht über die Rundungen hinausragt. Durch die Border-Radien bieten sich interessante Möglichkeiten, die Form von Boxen zu beeinflussen:


Abb. 35: reine auf border-radius basierende Kreisformen

### b) border-image

Mit den neuen Eigenschaften von border-image können statt den üblichen Border-Stilen Hintergrund-Bilder für die Rahmen benutzt werden. Das zu verwendende Bild stellt dabei den kompletten Rahmen dar und wird intern in 9 Bereiche unterteilt, die den jeweiligen Border-Bereichen zugewiesen werden.

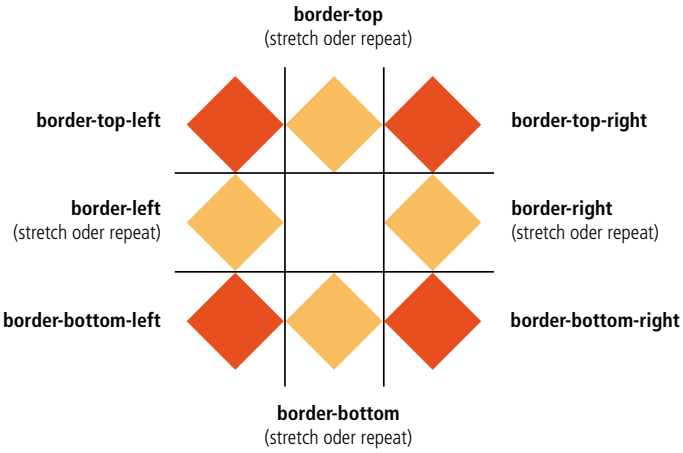



Abb. 36.: Die Einteilung der border-Grafik in 9 Teilbereiche- Die helleren Teile können wiederholt oder gedehnt werden, um die Länge auszufüllen.

Zunächst muß eine Rahmenbreite (`border-width`) angegeben werden, in die das Bild später eingepasst wird. Mit der Deklaration `border-image` werden nun die Hintergrundbilder für die Rahmen angewandt.

```
border-img {
      border-width:20px;
      width:400px;
      height:400px;
      border-image: url("border.png") 27 27 27 27 repeat;

}
```

1 2 3

Q24 CSS-Anweisung für border-images

1 `url(„border.png“)` ist die übliche Angabe der URL des Grafik

2 Die Angabe 27 27 27 27 definiert, wie die Eckbereiche[127] der Grafik eingeteilt (gesliced) werden. Der erste 27px Wert steht für die Ecke oben links, der zweite 27px Wert für die Ecke oben rechts, der dritte 27px Wert für die Ecke unten rechts und der vierte 27px Wert für die Ecke unten links. In dem Beispiel sind alle Bereiche gleichmäßig aufgeteilt, aber durch diese Möglichkeit des Slicens können auch Bilder mit ungleichen Eckbereichen exakt für ihre Verwendung in eine der vier Ecken vorbereitet werden, indem die entsprechenden Dimensionen dieser Bereiche deklariert werden.

3 Der dritte Teil `repeat` beschreibt, wie sich die Rahmen-Seiten der Grafik verhalten .[128] Wenn die Box sich in Höhe oder Breite vergrößert, muß es eine Definition geben, wie sich diese Elemente an die unterschiedliche Größe anpassen. Mit `repeat` werden die Bereiche der Grafik wiederholt. Mit `stretch` werden die Bereiche der Grafik über die komplette Länge der Seite gedehnt.[129] Die Eckgrafiken bleiben dabei unangetastet, da Größenänderungen der Box keinen Einfluß auf sie haben.

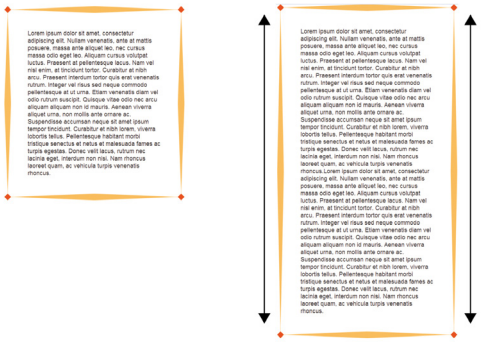


Abb. 37.: stretch-Verhalten von border-image - Ausdehnung der Rahmengrafiken

[127] border-top-left, border-top-right, border-bottom-right, border-bottom-left) Vgl Abb36.

[128] die hellen Bereiche in Abb36: border-top, border-right, border-bottom, border-left.

[129] Vgl. Abb. 37

### 3.2.4.3. Schlagschatten

Mit den neuen Eigenschaften `box-shadow` und `text-shadow` können Elemente mit innenliegenden und außenliegenden Schlagschatten versehen werden. Ein Schlagschatten ist der Schatten, den ein Objekt auf seinem Hintergrund hinterlässt – er bildet den Umriß des Objektes nach und lässt die Richtung des Lichtes erkennen. Im Designprozess wird er oftmals verwendet, um bestimmten Elementen Tiefe zu geben und sie vom Rest abzuheben.

#### a) box-shadow

`box-shadow` setzt sich aus einer kommagetrennten Liste der Schattendeklarationen zusammen, die wiederum mit 2-4 Werten, einer optionalen Farbangabe und einer optionalem Schlüsselwort „`inset`" angegeben werden. Der Schlagschatten wird hinter dem Element, das die Eigenschaft verwendet, angezeigt und wirkt sich, im Gegensatz zu Rahmen und Außenabständen, nicht auf das Layout und Verhalten anderer Elemente aus.[130]

```
                        1    2    3    4    5                    6
-webkit-box-shadow: 10px 10px 12px 10px rgba(0,20,10,0.5) inset;
```

Q24: Angabe eines box-shadows

**1** Definition der Verschiebung auf der x-Achse: positive Werte bewirken eine Verschiebung vom Ursprungsort nach rechts, negative Werte eine Verschiebung nach links.

**2** Definition der Verschiebung auf der y-Achse: positive Werte bewirken eine Verschiebung nach unten, negative Werte eine Verschiebung nach oben

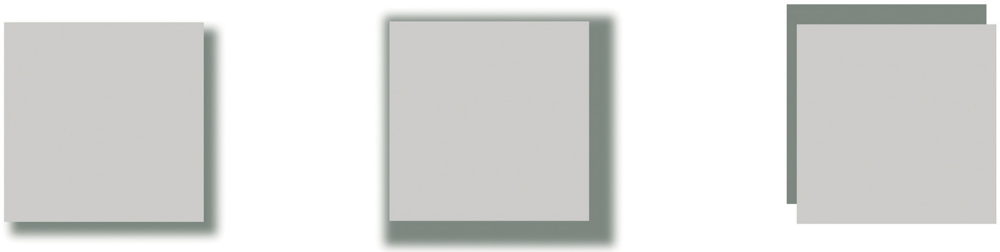
**3** steht für die Unschärfe des Schattens, je größer der Wert, desto verschwommener ist der Schatten

**4** definiert die Größe der Ausbreitung. Der Wert dehnt die komplette Form des Schattens nach allen Seiten aus, je größer der Wert, desto größer die Ausdehnung.

**5** Angabe des Farbwertes als Hex-Wert, oder mit den Funktionen rgb() und rgba();

**6** Mit der Angabe von „inset" wird der Schatten nach innen verlagert.

[130] [W3C013]



```
Abb38. 1 - box-shadow: 10px 10px 12px  rgba(0,20,10,0.5);
Abb38- 2 - box-shadow: 10px 10px 12px 15px  rgba(0,20,10,0.5);
Abb28. 3 - box-shadow: -10px 20px  rgba(0,20,10,0.5);
```

Abb. 38.: verschiedene box-shadows  und der dazugehörige CSS-Wert

Typisch für `border-shadow` ist die Anpassung des Schattens an den Umriss des Elements.  In Kombination mit border-radius stehen dem/der Designer/in viele Möglichkeiten offen. Es können mehrere Schatten, durch Komma getrennt, auf ein Element angewandt werden. Wie auch bei den multiplen Hintergrundbildern steht der erste definierte Schlagschatten auch visuell über den anderen.

Abb. 39.: Form mit abgerundeten Ecken und mehreren Schlagschatten, die einen Feuerball ergeben.[131]

```
.shadowbox {
      -webkit-box-shadow: 0 0 4px white,
                  0 -5px 4px #ff3,
                  2px -10px 6px #fd3,
                  -2px -15px 11px #f80,
                  2px -25px 18px #f20,
                  30px 60px rgba(0,0,0,0.2),
                  -30px -10px rgba(0,0,0,0.2) inset;
      -webkit-border-radius:100px 50px 200px 10px;
      width:200px;
      height:200px;
      background:#ccc;
}
```

Q25: Quellcode zu Abb. 39

[131] adaptiert von: http://www.css3.info/preview/text-shadow/

### b) text-shadow

Mit der Eigenschaft ist es möglich, auch Text mit einem oder mehreren Schlagschatten zu hinterlegen. Die Eigenschaft verwendet die gleiche Wert-Syntax wie `box-shadow` mit dem Unterschied, dass die Größe der Ausbreitung wegfällt und damit nur 3 Maßeinheiten (x-Versatz, y-Versatz, Unschärfe) angegeben werden können. Auch multiple Schatten sind möglich.


Abb. 40.: text-shadow im Einsatz

```
text-shadow: 2px 2px 5px rgba(0,0,0,0.7);
```

Q26: der zu Abb 40. gehörige CSS-Code

Neben den dekorativen und ästhetischen Gründen ist diese Eigenschaft auch aus Sicht der besseren Lesbarkeit sinnvoll. Beispielsweise kann ein Schlagschatten gesetzt werden, um einfarbigen Text von einem bunten Hintergrundbild hervorzuheben.

### 3.2.4.4. Farben & Transparenzen

Das Color Modul beinhaltet alle CSS-Eigenschaften, mit denen Farbwerte und Transparenzen[132] von Elementen definiert werden können. Außerdem beinhaltet es alle möglichen Arten der Farbangaben, wie die neuen Farbfunktionen rgba(), hsl() und hsla() sowie das neue Schlüsselwort `currentColor`.
Seit CSS2.1 können Farben im Hex-Format[133] , der rgb()-Funktion[134] oder vordefinierten Farb-Schlüsselworten[135] angegeben werden.

Der Grad der Transparenz kann derzeit mit der Eigenschaft `opacity` angegeben werden, die die Durchsichtigkeit des kompletten Elements auf einer Skala mit einem Alphawert von 0.0 – 1.0 definiert.

#### a.) RGBA Farb Werte

Neu ist die Erweiterung des RGB-Farbmodells und der dazugehörigen rgb()-Funktion mit einem zusätzlichen Alpha-Wert, mit dem die Transparenz definiert wird. Diese rgba()-Funktion macht es nun möglich, Bestandteile, wie Rahmen oder Hintergründe, mit Transparenzen zu versehen, ohne dass die anderen Bestandteile, wie z.B. der Text im Element, davon beeinflusst werden. Bisher war es nur möglich, komplette Elemente mit Transparenz zu versehen.


Abb. 41.: Definition der Hintergrundfarbe mit Alpha-Transparenz. Alle anderen Bestandteile behalten die volle Deckkraft

[132] Die Deckkraft / Durchsichtigkeit eines Elements
[133] 6stelliger Hexadezimal-Code mit vorangestelltem # = Enstprechung des RGB-Modells
[134] Die rgb-Funktion formuliert eine direkte Eingabe der einzelnen Rot, Grün und Blau-Farbanteile auf einer Skala von 0-255: color: rgb(255,0,50) – volles Rot, kein Grün, 50 Blau.
[135] z.B. black, blue, yellow vollständige Liste: [W3 SCHOOLS 001]

```
div.colorbox {
      background-color:rgba(45,55,95,0.7);
      border:8px solid #eee;
      padding:20px;
      -webkit-border-radius:10px;
      -webkit-box-shadow: 10px 10px 5px rgba(55,55,55,0.3);
      width:300px;
      height:150px;
      text-align:center;
      margin:0px auto;
      color:#fff;

}
```

Q27 Der zu Abb. 41 gehörende Quellcode

Die Box verwendet für die Hintergrundfarbe (`background-color`) die funktionelle Notation rgba() mit einer 70% Transparenz. Den Hintergrund lässt die Box durchschimmern, alle anderen Elemente behalten ihre volle Deckkraft. Die Angabe mit rgba(), eignet sich gerade auch für die `box-shadow`-Eigenschaft. So können die Schatten durch Anpassen des Alphawertes nach Belieben dezenter gemacht werden.

### b) HSL / HSLA-Farbwerte

Eine andere funktionelle Notation ist die Angabe mit HSL bzw HSLA. Die Buchstaben des HSL-Farbraums stehen für Farbton (Hue), Sättigung (Saturation) und Helligkeit (Lightness). Der Farbraum stellt eine intuitive Farbwahrnehmung/Farbmischung dar, da er der Farbwahrnehmung des Menschen am ehesten entspricht. Der Farbwert wird anhand des Farbkreises direkt bestimmt und kann anschließend durch Sättigung und der relativen Helligkeit in seiner Wirkung variiert werden.[136]
Die Syntax von hsl /hsla entspricht dabei der Syntax von rgb / rgba.
Die einzelnen Werte werden kommagetrennt hintereinander angegeben, der vierte Wert steht dabei immer für den Alphawert.

[136] [COYIER]

Der zusätzliche Alpha-Wert der neuen funktionellen Notationen ist von großer Bedeutung. War es mit CSS2.1. ausschließlich möglich, kompletten Elementen (inkl. ihrer Rahmen, Hintergründen, Textinhalten) einen einzigen Transparenzwert (`opacity`) zu geben, können nun auch die einzelnen Bestandteile von Elementen beliebig transparent gemacht werden.

In der CSS3-Version in Abb 42 b) erhält nur der Hintergrund eine 50%ige Transparenz, die Schrift bleibt komplett schwarz deckend. Dies erleichtert das Lesen und sorgt für eine dezentere Hervorhebung des Überschriftenelements. Wollte man mit CSS2.1. einen halbtransparenten Background- erstellen, der aber volldeckende Schriften verwendet, mußte entweder ein extra 24Bit PNG mit halber Transparenz als Hintergrundbild verwendet werden oder ein zusätzliches, semantisch nutzloses, HTML-Element in den Container eingebaut werden, in dem die Schrift liegt und das seine `opacity` wieder auf `1.0` setzt – beides sehr umständliche Wege.


```
color: rgb(0, 0, 0);
background-color:
rgb(255, 255, 255);
opacity: 0.5;
```

```
color: rgb(0, 0, 0);
background-color:
rgba(255, 255, 255,
0.5);
```

Abb. 42.: opacity vs. CSS3-Alpha-Transparenz[137]

Ein weiteres Beispiel soll den Nutzwert der neuen Transparenz verdeutlichen. Über ein buntes Foto soll eine weiße Titelleiste gelegt werden. Mit Hilfe des rgba-Alphawertes bleibt die Schrift lesbar und verdeckt wiederum nicht zu viel vom Hintergrundfoto.


Abb. 43 Verbesserung der Lesbarkeit durch einen halbtransparenten, dezenten Hintergrund.

[137] Bildquelle http://24ways.org/2009/working-with-rgba-colour

### 3.2.4.5. Farbverläufe

Farbverläufe[138] sind feste Bestandteile von modernem Grafikdesign und gerade bei der Gestaltung von virtuellen Nutzeroberflächen kaum wegzudenken. Sie suggerieren dem Nutzer eine gewisse Greifbarkeit der Elemente und simulieren Lichteinfall. Wird mit ihnen richtig umgegangen, lassen sie ein Design „echter" und anfassbarer wirken. Gerade im digitalen Einsatzgebiet werden Verläufe verwendet, um der fehlenden Plastizität und der fehlenden Natürlichkeit ein gestalterisches Mittel entgegenzusetzen.


Abb. 44.: Farbverläufe in der Praxis: Apple.com: Farbverlauf der Navigation vs. Poduktfarbverlauf, spiegel.de Farbverläufe im Webdesign

Bisher war es nur mit dem Einsatz von vorher generierten Grafiken, die mittels `background-image` und der `background-repeat` Eigenschaft gesetzt wurden, möglich, Farbverläufe in cssbasierten Oberflächen zu verwenden[139]
Die Definitionen zu CSS-Farbverläufen sind Teil des CSS3 Images-Moduls[140], das den Umgang mit Bildwerten (z.B. die `url()`-Notation bei `background-image`) beinhaltet. Verläufe werden mittels den neuen funktionellen Wertetypen `linear-gradient()` und `radial-gradient()` angegeben. Die beiden Funktionen können also überall eingesetzt werden, wo auch Bilder angegeben werden können, so z.B. bei den Eigenschaften `background` oder `list-style-image`

#### a.) Lineare Farbverläufe

Lineare Farbverläufe werden durch eine Verlaufslinie inkl. ihrer Richtung oder einem Schrägungswinkel sowie mindestens 2 Farbpunkten[141], die auf dieser Linie platziert werden sollen, angegeben

```
linear-gradient(startposition, winkel, farbpunkt1 (farbpunkt1-position), farbpunkt2 (farbpunkt2-position), …)
```

[138] Farbverlauf ist eine stückweise Veränderung von einer Farbe zu einer nächsten Farbe entlang einer Linie (linearer Farbverlauf) oder in Kreisabschnittes (radialer Farbverlauf)
[139] Vgl. 2.6.5.
[140] [W3C 013]
[141] Den sog. Color Stops, Vgl [W3C 013]

```
background: -moz-linear-gradient(45deg,#ccc, #000);
background: -webkit-gradient(linear, left bottom, right top,
            from(#ccc), to(#000));
```

Q28: CSS-Angabe eines einfachen linearen Verlaufes, unterschiedliche Notationen von Mozila (standardkonform) und Webkit (eigene Interpretation)

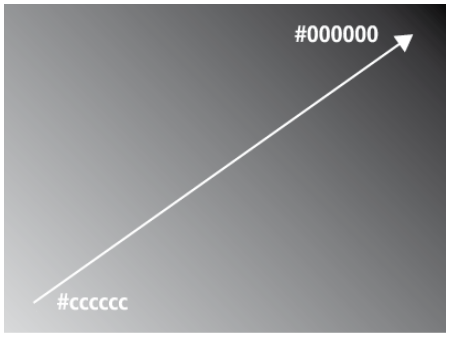



Abb. 45.: Ergebnis von Q27

In Abb. 45. wird ein einfacher Farbverlauf zwischen 2 Farben gesetzt. Auffällig ist die unterschiedliche Notation für die beiden Engines Gecko und Webkit. Im Folgenden wird nur die `-moz-`Notation verwendet, da diese der Syntax des W3C entspricht.

Soll es mehrere Color-Stops geben, die an frei gewählten Punkten der Verlaufslinie sitzen, wird die Position direkt hinter der Farbangabe in relativen und festen Werten definiert.

```
background: -moz-linear-gradient(right top, #ccc 20%, #000 56%,
#eee 80%);
```

Q29: CSS-Farbverlauf mit mehreren individuellen Farbpunkten

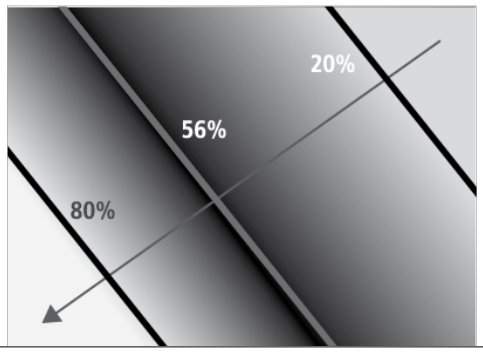



Abb. 46: Ergebnis von Q29

Natürlich ist die Angabe der Farbwerte auch mit den neuen `rgba()` /`hsla()`-Notationen möglich, wodurch die CSS-Verläufe z.B. halbtransparent gemacht werden können.

**b) radiale Verläufe:**

Auch radiale Verläufe, bei denen Farben in Kreisbahnen ineinander übergehen, sind mit der funktionellen Notation `radial-gradient()` möglich. Die Syntax ist dabei ähnlich zu `linear-gradient()` mit dem Unterschied, dass zusätzlich noch die Ausbreitung des radialen Verlaufs sowie dessen Form mit vordefinierten Schlüsselworten angegeben werden kann.

**Formenangaben:**

- ellipse – Der Verlauf besitzt eine Ellipsenform (unterschiedliche Radien)
- circle – Der Verlauf besitzt eine Kreisform (gleicher Radius)

**Ausbreitung des Verlaufs:**

- closest-side: breitet sich bis zur nächstliegenden Seite aus
- closest-corner: breitet sich zur nächstliegenden Ecke aus
- farthest-side: breitet sich zur am weitesten entfernten Seite aus
- farthest-corner: breitet sich zur am weitesten entfernten Ecke aus
- contain: Der komplette Verlauf befindet sich in der Box, u.U. entsteht Leerraum[142]
- cover: Der Verlauf wird über die komplette Box gestreckt.[143]

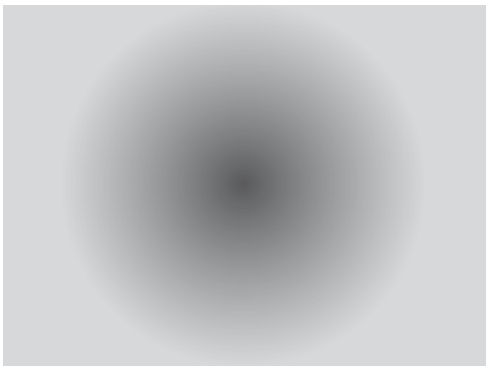


Abb. 47: Radialer CSS - Verlauf

```
background: -moz-radial-gradient(50% 50% 64deg,circle contain,
            #363636, #C9C9C9);
```

[142] Vgl. 3.2.4.1 g)
[143] Dito

## 3.2.5. Text & Typographie

### 3.2.5.1. Text Level 3

Das Text-Modul in CSS3 enthält alle Eigenschaften, die der Manipulation und Darstellung von Text dienen. Es deckt die Bereiche Zeilenumbrüche, Textausrichtung, Leerräume, Textdekoration und Transformation ab und trägt einen großen Teil zur besseren Internationalisierung bei.

**a.) Leerräume**
Mit den neuen Eigenschaften `white-space-collapse` wird definiert, wie Leeräume, die z.B. durch Leerzeichen, Zeilenumbrüche und Tabulator-Einrückungen entstehen, behandelt werden.

`collapse:` Egal wie viel Leerraum im Text-Inhalt vorhanden ist, der komplette Leerraum wird auf ein einzelnes Leerzeichen reduziert. Zeilenumbrüche werden ignoriert.
`preserve:` der komplette Leerraum wird erhalten, auch Zeilenumbrüche werden erhalten[144]
`preserve-breaks:` Zeilenumbrüche werden beibehalten, Leerraum als solcher wird auf ein einzelnes Leerzeichen reduziert.
`discard:` Jeglicher Leerraum im Element wird (visuell) entfernt

**b) Silbentrennung**
Mit der Eigenschaft `hyphenate` wird die Silbentrennung gesteuert. Zulässige Werte sind „`none`“ und „`auto`“. Ist `auto` gesetzt, wird das Wort anhand des Silbentrennungs-Algorithmus des User-Agents sinnvoll mit einem Divis[145] getrennt, wenn es nicht komplett in die Gesamtbreite des Elements passt.

**c) Zeilenumbrüche**

Die Eigenschaft `text-wrap` definiert, wie Zeilenumbrüche im Text vollzogen werden.

`normal` Zeilen werden an allen erlaubten Stellen des Textes umgebrochen
`none` Es gibt keine Zeilenumbrüche. Text, der nicht in eine Box passt, überlappt diese komplett
`unrestricted` Zeilenumbrüche passieren an jeder erdenklichen Stelle.
Zeilenumbruch-Einschränkungen haben keine Auswirkungen und auch die Silbentrennung findet nicht statt.

---

[144] (ohne exta `<br />`-Tags Setzten zu müssen)
[145] Einfacher Trennstrich -

**suppressed** Zeilenumbrüche innerhalb von Elementen[146] werden möglichst unterdrückt, außer wenn es zu einer Überlappung des Textes über seine Box kommen würde. Umbrüche an gültigen Punkten vor und nach dem Element haben allerdings Priorität.

### d) word-wrap

**word-wrap** beschreibt, an welcher Stelle innerhalb eines Wortes ein Zeilenumbruch stattfinden darf, wenn dieses Wort sonst nicht anders umzubrechen ist, und es die Box überlappen würde. Zulässige Werte sind **normal** (Zeilenumbruch nur an erlaubten Punkten) und **break-word** (Umbruch an einem beliebigen Punkt, falls das Wort zu lang ist und nicht anders umgebrochen werden kann)

### e) Ausrichtung und Anordnung

Die Eigenschaft **text-align**, mit der die Ausrichtung des Textes in einem Blockelement definiert wird, wurde zusätzlich um die Werte **start** und **end** erweitert. **start** steht dabei für die Ausrichtung des Textes an der Anfangsgrenze der Text-Zeile.
**end** steht für die Ausrichtung an der Endgrenze Text-Box.

Neu ist die Eigenschaft **text-align-last**, mit der die Ausrichtung der letzten Zeile eines Textblocks oder die letzte Zeile vor einem Zeilenumbruch definiert wird, wenn **text-align** auf **justify** gesetzt ist[147] Zulässige Werte sind dabei: **start,end,left,right,center,justify**.

Ebenfalls neu ist die Eigenschaft **text-justify**, mit der die Methode des Blocksatzes beeinflußt werden kann. Sie ist wichtig für verschiedene Sprachsysteme und deren Buchstabenarten. Mit den zulässigen Werten, können für arabische, asiatische und Lateinische Sprachen die passenden Buchstabenabstände bzw. Wortabstände im Blocksatz gewählt werden.

### e.) Text-Überlappung

Hat eine Box feste Dimensionen und ein zu langer Text geht über die Box hinaus, war es bis dato nur möglich, mittels **overflow:hidden** den Text komplett abzuschneiden. Bei ungünstiger Anwendung wurden dabei u.A. Wörter oder Buchstaben halb „abgeschnitten". Mit der neuen **text-overflow** Eigenschaft kann dieses Verhalten kontrolliert werden.

---

[146] (also zwischen einem öffnenden HTML-und einem schließenden Tag)
[147] Blocksatz

**Zulässige Werte für den Überlappungsmodus sind dabei:**

- `clip` – Schnitt an der Überlappungsstelle (gleiches Verhalten wie bei overflow)

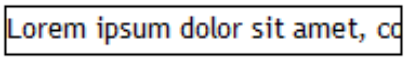



Abb. 47: text-overflow:clip - Text wird an Überlappungsstelle abgeschnitten[148]

- `ellipsis` – Es wird ein mit „text-overflow-ellipsis“ angegebener Auslassungs-String an der Überlappungsstelle eingefügt.
- `ellipsis-word` – gleiches Verhalten wie bei ellipsis, mit dem Unterschied, dass die Auslassungszeichen beim letzten vollständig angezeigten Wort erscheinen

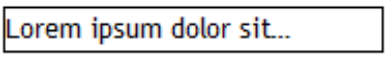



Abb. 48: text-overflow: ellipsis-word

Mit `text-overflow: ellipsis-word` kann ein eigener Auslassungsstring definiert werden[149]

### f) Text-Dekoration

Mit `text-outline` wird dem kompletten Text eine Umrahmung gegeben. Teil der Deklaration sind die Outline-Breite, ein zusätzlicher optionaler Unschärfe-Radius sowie die Farbe der Outline. Mit Hilfe von `text-outline` kann der weiße Text auf dem Skylinefoto aus Abb. 49 sehr klar vom Hintergrund abgehoben werden, was die Lesbarkeit deutlich verbessert:


Abb. 49. Hervorhebung der Schrift mit `text-outline:2px #000000;`

[148] Quelle [CSS3 INFO 002]

[149] (Standard: „...“)

### 3.2.5.2. Webfonts in CSS3

Ein Merkmal, das schon in CSS2.1[150] angedacht war, ist die Implementierung eigener Schriftarten inkl. aller Schriftschnitte über die `@font-face` Regel.
Damit wird die technisch bedingte Eintönigkeit der Schriften im Web endlich aufgebrochen, ohne dass auf JavaScript[151] und Flashlösungen[152] oder auf die semantisch nutzlose und aufwändige Einbindung von Bildern zurückgegriffen werden muß. Über diese @-Regel können verschiedene Schriftdateien im TTF, OTF, dem neuen WOFF, SVG und EOT-Formaten direkt als nutzbare `font-family` verlinkt und in der Gestaltung verwendet werden.

#### a) Deklaration und Einbindung der Schriftdateien

Als Testschrift wird die lizenzfreie Schriftfamilie „Quicksand" verwendet, da diese 7 verschiedene Schriftschnitte verwendet und somit die verschiedenen Möglichkeiten von `@font-face` sehr gut demonstriert.[153]

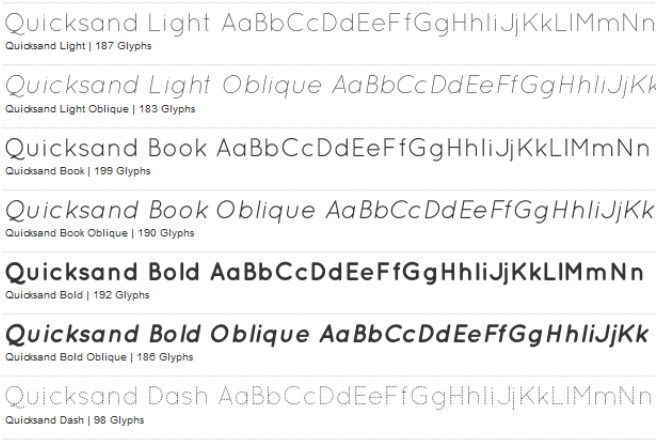



Abb. 50: Die lizenzfreie Schriftart „Quicksand" des Autors Andrew Paglinawan

Zunächst wird in der `@font-face` Regel der Name für die einzubindende Schriftfamilie definiert. Dieser kann beliebig vergeben werden, es empfiehlt sich aber, den in der Schriftdatei vorgesehenen Namen zu verwenden.

[150] [W3C 014]
[151] Cufon http://cufon.shoqolate.com/generate/
[152] sIFR http://www.mikeindustries.com/blog/sifr
[153] „This font comes unlicensed. Feel free to use it commercially as it is license free." [PAGLINAWAN]

```
@font-face {
      font-family: Quicksand;
      src: local("Quicksand"),
      url(font/Quicksand_Book-webfont.ttf) format("truetype");
}
```

Q30: Basis-Definition der @font-face Regel

Mittels `src` werden nun alle Informationen zur Quelle der Schrift angegeben. Die funktionelle Notation `local` ist optional und aus der Praxiserfahrung nicht empfehlenswert.[154] Durch sie wird als erstes überprüft, ob die Schrift schon in der Schriftdatenbank des Nutzers installiert ist.[155] Ist dies der Fall, muß sie nicht heruntergeladen werden, was Bandbreite des Nutzers und des Anbieters spart. Ist dies nicht der Fall, wird über `url` die Schriftdatei heruntergeladen und eingebunden. Mit `format` wird nun noch das Format der Schrift – hier TruetypeFont[156] – angegeben.

Um die Schrift zu verwenden, wird über `font-family` der Schriftfamilien-Name angegeben. Es empfiehlt sich nach wie vor auch Fallback-Schriften bzw. den generische n Schrifttypus-Namen anzugeben, um eventuell auftretende Probleme beim Downloaden abzufedern.

```
h2 {
      font-family:"Quicksand", Arial, Helvetica, sans-serif;
}
```

Q31: Verknüpfung der h2 mit der in Q30 eingebetteten Schriftart


Abb. 51: Ergebnis der Einbettung. Die Überschrift wird in der „Quicksand"-ausgegeben

Das Beispiel-Markup verwendet in der zweiten Zeile das Hervorhebungs-Tag `<strong>`, das mit einem fetterem Schriftschnitt visualisiert werden soll. Da aber für die Schrift-Familie „Quicksand" bisher nur ein Schriftschnitt exisitiert bzw. eingebunden wurde,

[154] [TYPOPHILE]
[155] Hierbei ist zu beachten, dass der Internet Explorer die Notation local() nicht interpretieren kann, und auch die folgende url-Notation ignoriert.
[156] [ADOBE 001]

wird die zweite Zeile im gleichen Schnitt dargestellt. Durch die explizite Angabe, welcher Schnitt verwendet werden soll, wird dieses Problem gelöst:

```
@font-face {
      font-family: Quicksand;
      src: local("Quicksand"),
      url(font/Quicksand_Book-webfont.ttf) format(„truetype");
      font-weight:normal;
}


@font-face {
      font-family: Quicksand;
      src: local("Quicksand Bold"),
      url(font/Quicksand_Bold-webfont.ttf) format("truetype");
      font-weight:bold;
}
```

Q32: Der font-family „Quicksand" wird ein normaler Schriftschnitt und ein dickerer Schnitt zugewiesen

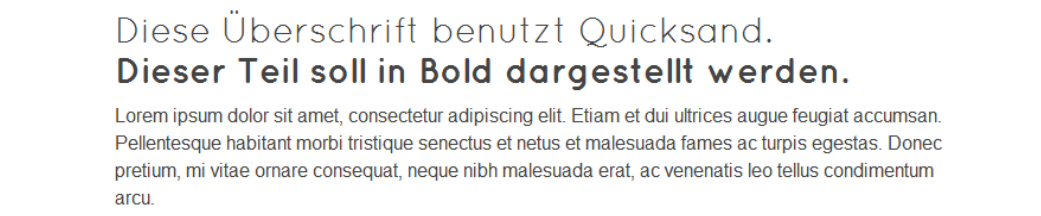



Abb. 52: Die zweite Zeile wird mit dem dickeren Schriftschnitt dargestellt.

Dieser Schritt[157] ist für alle Schriftstile zu wiederholen, die ein CSS-Entwickler im Stylesheet benutzt. Der Kombination sind dabei keine Grenzen gesetzt. Soll ein „Bold-Italic" eingesetzt werden, muß in der entsprechenden `@font-face` Deklaration `font-weight` auf „`bold`" und font-style auf „`italic`" gesetzt werden. In Kombination mit „`font-style`" und der „`font-stretch`"-Eigenschaft, bei der Condensed oder Expanded Schriftstile angegeben werden, können sämtliche Stile, selbst von den größten Schriftfamilien zur Verwendung in CSS freigegeben werden. Mit den Möglichkeiten der Textbeeinflußung sind dadurch typographischen Feinheiten kaum Grenzen gesetzt.

[157] Vgl. Q32 & Abb. 52

### b) Browserübergreifende Kompatibilität

Ein großer (derzeitiger) Nachteil der `@font-face` Variante zur Schrifteinbettung sind die Formatprobleme verschiedener Browser. Insbesondere der Internet Explorer unterstützt bisher nur Schriftdateien im eigenen .eot-Format[158] . Mozilla Browser (Gecko Engine) unterstützen OTF und TTF. Safari und Opera besitzen die größte Formatunterstützung mit den Formaten OTF, TTF und SVG, Google Chrome unterstützt wiederum nur TTF un SVG. Bei Mobile Browsers sieht das Ganze schon wieder anders aus. TTF und OTF werden von Android und Mozilla Browsern unterstützt, während der Mobile Safari nur SVG Support anbietet.[159]

Webfont format support by browser

| | Embedded Open Type (.eot) | TrueType (.ttf) | OpenType (otf) | Web Open Font Format (.woff) | SVG (.svg) |
|---|---|---|---|---|---|
| Mobile Browsers[1] | | | | | |
| Mobile IE (≤ WinMo 6.5) | No | No | No | No | No |
| Opera Mini (≤ v4.2) | No | No | No | No | No |
| Opera Mobile (9.7 Beta WinMo) | No | No | No | No | No |
| Palm WebOS | No | No | No | No | No |
| Mobile Safari (3.1) | No | No | No | No | Yes! |
| Android (≥ 2.2 SDK) | No | Yes | Yes | No | No |
| Desktop Browsers (Incomplete. I'll be filling this in more over time) | | | | | |
| IE (≥ 5.5) | Yes | No | No | No | No |
| Firefox | No | Yes (≥ 3.5) | Yes (≥ 3.5) | Yes (≥ 3.6) | No |
| Safari | No | Yes (≥ 3.1) | Yes (≥ 3.1) | No | Yes (≥ 3.1) |
| Opera | No | Yes | Yes | No | Yes |
| Chrome | No | No | No | No | Yes (≥ 3.0) |
| [1] I haven't had a chance to test Mozilla's portable offerings | | | | | |

Abb. 53 - Browserunterstützung von @font-face[160]

Zurzeit ist also das TTF-Format das am weitesten unterstützte Schriftformat, das – bis auf den Internet Explorer – von nahezu allen (Desktop)-Browsern verwendet werden kann. Große Umstände bereitet Microsofts eot-Format, da es ein extra für den Zweck der Schrifteinbettung geschaffenes Dateiformat ist und auch nur vom Internet Explorer unterstützt wird.

Der IE war einer der ersten Browser, die Font-Embedding überhaupt verwenden konnten.[161] Das EOT-Format wurde aus dem Grund geschaffen, lizenzierte Schriften auch

[158] [W3C 015]
[159] Vgl Abb. 53 und [WEBFONTS]
[160] Quelle: [HENRY]
[161] schon mit dem IE4 im Jahre 1997

im Web zu verwenden, ohne den EULA Lizenzvertrag[162] zu verletzen.[163]
Der Lizenzvertrag regelt die Benutzung und Verbreitung der Schriftarten.
Für die meisten kommerziellen Fonts gilt: Schriften dürfen nur in einem sicheren Format, das ausschließlich das Betrachten und Drucken des Textes erlaubt,verwendet werden.[164] Da Fonts bei der Einbettung auf einem öffentlichen Server liegen, besteht die Möglichkeit, Schriftarten kostenlos herunterzuladen und sie zu benutzen oder zu verändern. Das EOT-Format verhindert dies, da es nur in der Schrifteinbettung funktioniert, nicht aber in Textverarbeitungs- oder Grafik/Satzprogrammen und auch nicht zu TTF oder OTF rückkonvertiert werden kann. Auf technischer Seite hat dies allerdings den Nachteil, dass jede `@font-face` Regel mehrfach deklariert werden muss, damit die Schrifteinbettung browserübergreifend funktioniert. Hinzu kommt die Konvertierung der Schriften in die entsprechenden Formate, was insbesondere bei eot externe Software und zusätzlichen Aufwand in Anspruch nimmt.[165]

Eine Lösung, die zumindest den Aufwand der Deklaration vereinfacht, ist die sogenannte „Bulletproof font-face-Syntax" des Webentwicklers Paul Irish[166]. Eine browserübergreifende @font-face-Deklaration nach dieser idee gestaltet sich wie folgt:

```
@font-face {
      font-family: ,QuicksandBook';
      src: url(,Quicksand_Book-webfont.eot'); /* für den IE */

      src: local(,☺'),
      url(,Quicksand_Book-webfont.woff') format(,woff'),
      url(,Quicksand_Book-webfont.ttf') format(,truetype'),
      url(,Quicksand_Book-webfont.svg') format(,svg');
      font-weight: normal;
      font-style: normal;
}
```

Q33: „Bulletproof @font-face Syntax" von Paul Irish - Browserübergreifend @font-face einbetten

Die Lösung von Paul Irish macht sich die fehlende IE-Unterstützung von local() zunutze, und nimmt diese als eine Art Browserweiche. Dadurch werden alle Font-Definition, die nach local() folgen, vom IE ignoriert und der Browser implementiert folgerichtig die erste eot-Anweisung. Alle anderen Datei-Implementationen sind für andere Browser angedacht, die sich einfach das passende Format heraus suchen. Der Smiley in der local()-Anweisung ist Platzhalter für einen Schriftnamen, der so definitiv nicht auf ei-

[162] EULA – Enduser License Agreement
[163] [W3C 015]
[164] Fonts.com EULA [...]You may electronically distribute Font Software embedded [...] only when the Font Software embedded in such document (i) is in a static graphic image [...]or an embedded electronic document, and is distributed in a secure format that permits only the viewing and Printing [...]
[165] http://www.kirsle.net/wizards/ttf2eot.cgi Kommandozeilentool ttf2eot
[166] [IRISH}

nem lokalen Rechner existiert, um Praxisrisiken mit lokalen Schriften zu umgehen.
Das Problem der Konvertierung in 5 Dateiformate bleibt allerdings nach wie vor bestehen.

### c) WOFF Webfonts & Lizenzen

Was die browserübergreifende Schrifteinbettung braucht, ist ein spezielles, lizenzrechtlich einsehbares, hochwertiges, weboptimiertes und leicht zu konvertierendes Format, das von allen Browsern unterstützt wird. Ein solches Format ist WOFF (Web Open Font Format), ein von Mozilla entwickeltes Webfont-Format, das erstmalig im Firefox 3.6. unterstützt wird.
Es erlaubt die Komprimierung von Truetype, OpenType und OpenFont Formaten in eine gepackte Form, so dass es auch wieder ohne Qualitätsverluste rückkonvertiert werden kann – es ist eine Art Reisekoffer für OTF- und TTF-Schriften.
Durch die zusätzlichen XML-Metadaten ist es möglich, Lizenzbestimmungen, Namen der Font-Designer, und Fontvertriebe direkt in die Schrift einzubetten.[167] Allerdings gibt es hier keinen DRM-ähnlichen Ansatz[168] , der die Schriften in bestimmten Situationen sperrt und mehr Kontrolle erlaubt. WOFF ist das Webfont-Format der Zukunft, so hat sich neben den Chrome-Entwicklern auch Microsoft für den Support von WOFF eingesetzt.
Das eigentliche Umdenken für einen neuen Umgang mit Schriftarten im Web muß aber von den Font-Vertrieben selbst kommen. Wie der digitale Musikmarkt gezeigt hat, wird es kaum auf lange Sicht zu verhindern sein, dass Nutzer , durch einen Blick in die CSS-Datei, Schriftdateien downloaden und kostenlos verwenden können.
Obwohl mit der Verschlüsselung von font-face-URLs (Obfuscation) bei typekit.com [169] eine Kontrolle geschaffen wurde, setzen einige größere Fontvertriebe wie FSI Fontshop lieber auf das offene Format WOFF und öffnen sich so dem neuen Markt.
Welche Lösung Erfolg hat, wird die Zukunft zeigen.
**Zum Zeitpunkt der Arbeit entstehen einige Webfont-Lösungen, so z.B.**

- Font Face-Generatoren / lizenzfreie kostenlose Webfonts inkl CSS-Code
  fontsquirrel.com
- kostenloses externes Fonthosting mit bequemen Einbetten
  http://code.google.com/webfonts
- kommerzielles verschlüsseltes Fonthosting
  typekit.com
- Erweiterte EULA, offene WOFF-Formate zum Kauf
  fontspring.com, fontshop.com, fonts.com

---

[167] [WOFF]
[168] DRM - Digital Rights Management [GOLEM]
[169] Für ein jährliches Abonnement können hunderte kommerzielle Schriftarten in den richtigen Formaten von typekit.com verlinkt/eingebettet werden. Die Einbettung unterliegt einer 64bit Verschlüsselung, um Fontklau zu unterbinden.

## 3.2.6. Transformationen

### 3.2.6.1. 2D Transformationen

Mit dem 2D Transformations Modul lässt sich die Form von Elementen auf verschiedene Arten im zweidimensionalen Raum verändern. Das Modul besteht aus den 2 Eigenschaften `transform` und `transform-origin` sowie einer Vielzahl funktioneller Notationen, die auch mehrfach angewandt bzw miteinander kombiniert werden können.[170]

`transform-origin` definiert dabei, wo der Ausgangspunkt der Transformation liegen soll, was insbesondere bei Rotationen ausschlaggebend ist. Der standardmäßige Ursprung ist die Mitte des Objekts (50% 50%). Ähnlich wie bei `background-position` werden zwei Prozentzahlen, Längen oder Schlüsselwerte als Ursprungsangabe angegeben.

**a) rotate**

Mit der `rotate()`-Funktion wird eine Drehung des Elements ausgehend vom mit `transform-origin` gesetzten Punkt um einen Winkel vollführt.
Positive Werte stellen dabei eine Drehung im Uhrzeigersinn dar.

```
-webkit-transform:rotate(10deg);
```

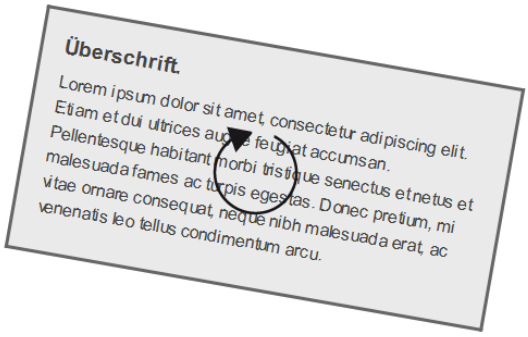



Abb. 55: rotate();

**b) skew (stauchen)**

Mit der Funktion skew() wird das komplette Element um einen bestimmten Winkel gestaucht

```
-webkit-transform:skew(30deg);
```

[170] [W3C 016]

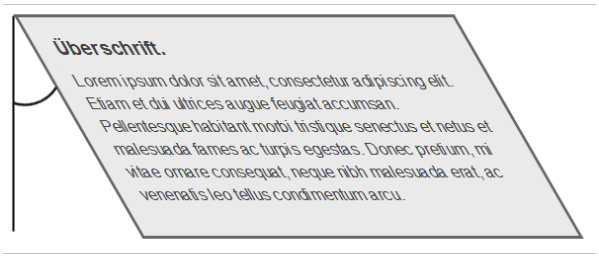



Abb. 56: skew();

### c) scale (Skalieren)

Mit der Skalierungsfunktion kann ein komplettes Element vergrößert oder verkleinert werden. Während die Funktion `scale()` das Element verhältnisbeibehaltend skaliert, kann durch die Funktionen `scaleX` und `scaleY` eine Skalierung nur einer Achse vorgenommen werden (x Breite und y Höhe). Nur die Eingabe von Fließkommawerten, die das 1:1.0 Verhältnis ausdrücken, möglich.

```
-webkit-transform:scalex(0.5);
```

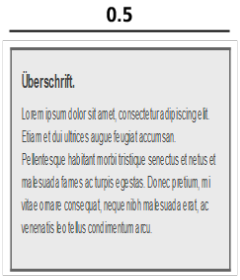



Abb. 57: scale();

### d) translate

Mit der `translate`-Funktion kann eine Verschiebung um einen x-und y-Wert erzielt werden, ausgehend vom Ursprung. Mit `translateX` und `translateY` kann eine Verschiebung auf einer Achse vorgenommen werden.

```
-webkit-transform:translate(50px, 50px);
```

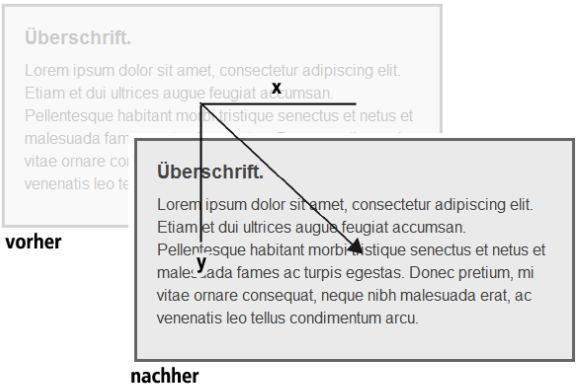



Abb. 58: translate();

### f) Kombination

Die Funktionen können beliebig miteinander kombiniert werden, indem sie hintereinander notiert werden.

```
transform: translate(80px, 80px) scale(1.5, 1.5) rotate(45deg);
```

Die wohl größte Aufälligkeit bei dem Modul ist die rein visuelle Transformation. Die Verschiebungen und Drehungen haben keine Auswirkungen auf andere Elemente oder den Viewport, sie werden von anderen Elementen genauso „behandelt“ wie in ihrer Ursprungslage.

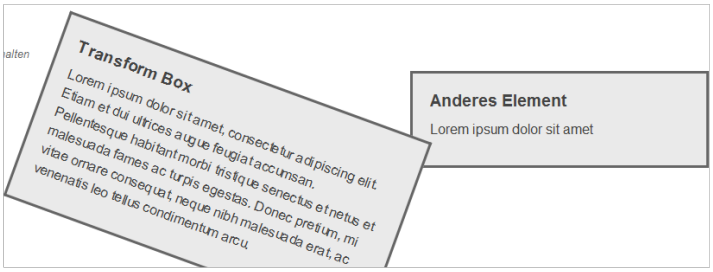



Abb. 59: Transformierte Elemente beeinflussen andere Elemente nicht. Sie werden wie in ihrer Ursprungs-Position behandel

### 3.2.6.2. 3D Transformationen

Diese Erweiterung der 2D-Transformationen ergänzt CSS erstmalig um einen dritte Raumachse Z, womit dreidimensionale Transformationen ermöglicht werden.[171]
Zusätzlich kommt eine Eigenschaft `perspective` zum Einsatz, mit der eine Tiefenwirkung der Elemente erzielt wird.[172] Jedem der im vorherigen Abschnitt vorgestellten Elemente wird zusätzlich eine Z-Funktion zugeteilt. So kann z.B: mit `rotateZ` eine Rotation um die Z-Achse erfolgen, oder mit `translateZ` eine Verschiebung auf der z-Achse.
Mit der Eigenschaft `transform-style`, die die Werte `flat` oder `preserve-3d` annehmen kann, wird angegeben, wie sich verschachtelte Elemente im 3D-Raum verhalten. Mit `flat` wird die Z-Achse auf die 2dimensionale Ebene projiziert.
Elemente, die sich auf unterschiedlichen Punkten der z-Achse befinden, sonst aber die gleichen x- und y-Koordinaten besitzen, werden aufeinander positioniert.
Mit `preserve-3d` wird eine dreidimensionale Darstellung erzielt, die Elemente behalten ihre Position im Raum. Mit der Eigenschaft `backface-visibility` wird angegeben, ob die Rückseite eines Elements angezeigt werden soll, wenn es sich z.B. um mindestens 180° gedreht hat. Der Standardwert ist `visible`, das Element wird angezeigt – Simulation einer Rückansicht. Mit `hidden` verschwindet es

#### b) Funktionen der 3D Transformationen

Die Funktionen der 2D-Transformationen wurden zusätzlich um eine `functionZ()` erweitert, um die dritte Raumachse einzubinden. Außerdem gibt es für jede Funktion noch eine `function3d()`-Variante, mit der eine gleichzeitige Transformation auf allen Achsen erzielt werden kann.

```
a)  -webkit-transform: rotateY(-30deg);
b)  -webkit-transform: rotateY(30deg);
```

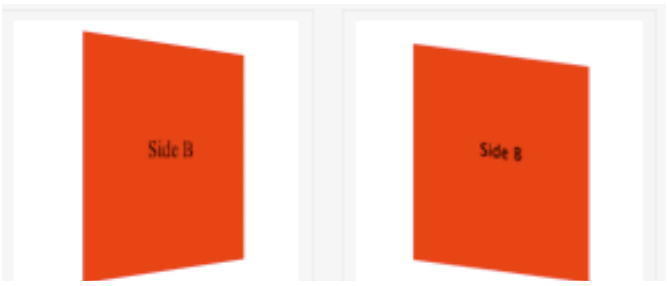



Abb. 60: 3D-Transformationen mit CSS - Ein `<div>`–Container wird um die y-Achse gedreht

[171] [W3C 016]
[172] Mit perspective-origin kann eine Koordinate angegeben werden, die den Betrachtungspunkt darstellen.

## 3.2.7. Ausgabeprofile

### 3.2.7.1. Media Queries – Medien Abfragen

Mit den„Media-Queries“ lassen sich Styleanweisungen direkt im CSS-Dokument auf bestimmte Medien (z.B. Screen) oder auch auf bestimmte Medieneigenschaften (z.B. Viewportgröße, Orientation) begrenzen. So können Darstellungen perfekt auf das Ausgabemedium angepasst werden, ohne den Inhalt ändern zu müssen.
Mit dem Media Queries wird eine Alternative zu „Liquid Layouts“ geschaffen.

Ab CSS 2 war es durch die @media-Regel möglich, die darin eingeschlossenen Stilanweisungen auf bestimmte Ausgabegeräte zu beschränken1[173]

```
@media print {
    body { font-size: 10pt }
  }
  @media screen {
    body { font-size: 13px }
  }
  @media screen, print {
    body { line-height: 1.2 }
  }
```

Q34: Die @media-Regel in CSS2.1 - Stilanweisungen auf Medientypen beschränken

Die zweite Methode für medienspezifische Stylesheets ist die Verwendung des media-Attributes in dem `<link>`-Tag im Kopf der HTML-Datei. Auch die Verwendung der `@import` Regel, kann eine CSS-Datei nur bei Verwendung eines bestimmten Ausgabegerät laden.

```
<link rel="stylesheet" type="text/css" href="handheld.css"
media="handheld" />

HTML-Version: <style type="text/css" media="handheld">@import
„handheld.css"</style>

CSS-Version: @import „handheld.css" handheld
```

Q35: Weitere Möglichkeiten der Medienspezifizierung mittels @import und dem „media“-Attribut

Diese Optionen werden in CSS3 nun um einige komplexere Abfragen erweitert, die eine exakte Kontrolle nach Ausgabemedium oder Ausgabeeigenschaften ermöglichen.

[173] [W3C 018]

Diese Abfragen – die sog. Media Queries – sind logische Ausdrücke, die entweder wahr oder falsch sind.[174] Das assozierte CSS wird nur angewandt, wenn der Ausdruck wahr ist. In allen der vorhin vorgestellten Methoden können Media-Queries platziert werden.

Ein Media Query besteht im Allgemeinen aus der ***Angabe eines Medientyps*** und optionalen Ausdrücken.

```
@media all and (min-width:500px) { … }
```

Nur für Medientypen mit einer Mindestbreite von 500px

**Logische Schlüsselwörter sind**

- „not" (nicht)
- „and" (und)
- „only" (nur)

Medieneigenschaften werden in einer css-ähnlichen Syntax angegeben, stellen aber nicht immer eine CSS-Eigenschaft dar. Die meisten Medieneigenschaften akzeptieren min- oder max- Prefixe, um „gößer-oder-gleich" oder „kleiner-oder-gleich" Bedingungen auszudrücken

Durch die logischen Ausdrücken können relativ komplexe, aber dennoch intuitive und leicht zu lesende Medienabfragen erstellt werden:

```
@media not print and (min-color:1) {... }
```
Alle Ausgabemedien, die nicht druckbasiert sind und mindestens eine Farbe verwenden.

```
@media only screen and (min-color:1) {... }
```
Nur auf Farb-Bildschirmen

```
<link rel="stylesheet" type="text/css" href="16to9.css"
media="screen and (device-aspect-ratio:16/9)" />
```

Laden der 16to9.css bei Bildschirmausgabemedien mit einem Verhältnis von 16:9.

[174] [W3C 019]

Außerdem sind Kombinationen von logischen Ausdrücken möglich, um z.B. eine „Zwischen x und y"-Situation abzugrenzen. Desweiteren kann eine weiterer Media-Query ergänzt werden, der eine andere Situation für das anzuwendende Stylesheet beschreibt.

```
<link rel="stylesheet"
media="screen and (min-width: 380px) and (max-width:1200px),
print and (orientation:portrait)" href="example.css" />
```

Die „example.css" wird geladen, bei:

1. Query: Bildschirmausgabemedien mit einer Mindestbreit von 380px und einer Maximalbreite von 1200px.
2. Query Allen Druckmedien, die im Hochkantformat gedruckt werden

Durch die Media Queries lassen sich also sämtliche Medientypen mit den entsprechenden CSS-Anweisungen ausstatten. Für Webentwickler ist es damit um einiges leichter, ein einziges CSS Layout auf Mobile Geräte, Monitore mit geringer Auflösung, Monitore mit großer Auslösung sowie verschiedener Printversion anzupassen. Die Media Queries leisten damit auch einen großen Beitrag zu barrierefreien Gestaltung.

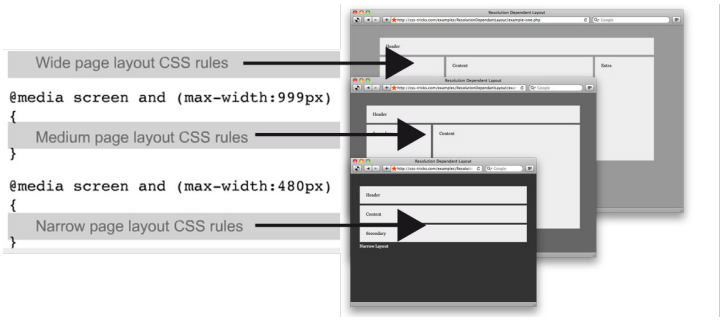



Abb. 61: unterschiedliche Stylesheets für unterschiedliche Anwendungsfälle

### 3.2.7.2. Paged Media

Das Paged Media Modul liefert mit Hilfe der `@page` Regel ein spezielles Modell, um seitenbasierte Dokumente zu layouten. Es ist eine gemeinsame Entwicklung von Opera und Hewlett Packard.[175] Seitenbasierte Medien haben im Vergleich zu Weblayouts, die fortlaufend sind, andere Erfordernisse an die Ausgabe:
Es muß eine Seitenummerierung möglich sein, es muß definiert werden können, wie sich das Layout auf unterschiedliche Seitenformate[176] anpasst, es müssen wiederholende Elemente definiert werden können[177], es muß das Verhalten von Elementen definiert werden, wenn sie mehrere Seiten überspannen. Weiterhin bedarf es einer Lösung um Doppelseiten definieren zu können sowie linke und rechte Seiten eines buchähnlichen Formates.

Das Page Modell definiert eine sog. rechteckige Page Box mit festen Abmessungen (je nach Größe des Seiten-Formats), die als Container für alle Elemente eines Dokuments fungiert.[178]
In dieser Page Box lässt sich der Inhalt auf 16 verschiedene Positionen – den sogenannten Page Sections – verteilen.[179]

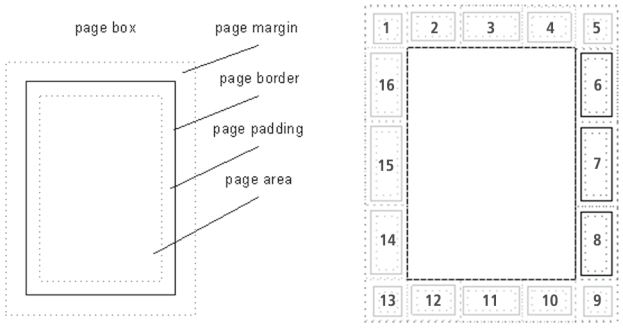



Abb. 62: Die Page Box und die 16 Layoutsektionen

Mit der `size` Eigenschaft kann nun flexibel festgelegt werden, wie groß das Medium ist. Außerdem wird angegeben, ob es sich um ein Hoch- oder Querformat handelt.

```
@page {
  size: A4 landscape;
}
```

[175] [W3C 020]
[176] A4 A5, Querformat, Hochformat, nichtdruckbare Bereiche, etc...
[177] (ähnlich eines Briefkopfs)
[178] [W3C 020]
[179] [W3C 021]

Dazu können verschiedene Seiten-Typen definiert werden, die andere Eigenschaften aufweisen können. Seiten, die unterschiedliche Typen benötigen sind z.B. die erste Seite eines Buches[180] oder die linken und rechten Seiten eines Buches.
Seitentypen werden mit folgender (strukturbasierter) Pseudoklasse[181] angesprochen.
Jeder so definierte Seitentyp hat Zugriff auf die 16 Sektionen.[182]
Die Sektionen können ebenfalls mit einer @-Regel spezifisch angesprochen werden.

```
@page :first {
  color: green;

  @top-left {
    color: blue;
  }
@page :left {
  margin-left: 3cm;
  margin-right: 4cm;
}

@page :right {
  margin-left: 4cm;
  margin-right: 3cm;
}
```

**Q36: Page Media - Definieren der Seitentypen und Seitensektionen**

Die erste Seite (`:first`) erhält die Farbe Grün. Die oben linke Sektion (`@top-left`) auf dieser ersten Seite erhält die Farbe Blau. Die linke Seite erhält einen Linksaußen-Abstand von 3cm und einen rechten Abstand von 4cm, bei der rechten Seite ist es genau umgedreht.

Mit den `page-break`-Eigenschaften kann nun definiert werden, wie sich bestimmte Elemente verhalten, wenn ein Seitenumbruch vor dem Element (`page-break-before`), nach dem Element (`page-break-after`) oder innerhalb des Elements (`page-break-inside`) geschieht.

**Zulässige Werte für page-break-inside sind:**

[180] Das Cover
[181] Vgl. 3.2.1.2.
[182] Vgl Abb 53

`auto` – ein Seitenumbruch innerhalb des Elements wird weder verboten noch erzwungen
`avoid` – ein Seitenumbruch innerhalb des Elements wird vermieden.

**Zulässige Werte für page-break-before und after sind:**

`auto` – ein Seitenumbruch vor (nach) dem Element wird weder verboten noch erzwungen
`avoid` – ein Seitenumbruch vor (nach) dem Element wird vermieden.
`always` – es wird immer ein Seitenumbruch vor (nach) dem Element erzwungen.
`left` – es werden ein oder zwei Seitenumbrüche vor (nach) dem Element erzwungen, damit die nächste Seite eine linke Seite ist.
`right`– es werden ein oder zwei Seitenumbrüche vor (nach) dem Element erzwungen, damit die nächste Seite eine rechte Seite ist.

```
h1 { page-break-before : always }
h2 { page-break-after : avoid }
```

Q37: Page-Break-Eigenschaften

Für die Hauptüberschrift wird ein vorheriger Seitenumbruch erzwungen[183] , die `h1` befindet sich also immer auf einer neuen Seite. Ein nachfolgender Seitenumbruch für `h2` wird vermieden. Nach ihr erscheint also immer der folgende Inhalt

Alle CSS3-Eigenschaften, die Größe, Hintergrund, Abstände, Schriften, Rahmen und Innenabstände beeinflussen, können in einem Seitenkontext angewandt werden.
So ergeben sich flexible Möglichkeiten exakte Vorlagen für eine Druckversion einer Website oder sogar eines rein auf HTML/CSS basierendes Seitendokument – ein PDF-Ersatz – zu definieren.

[183] Vgl. Q37

## 3.2.8. Übergänge & Animationen

### 3.2.8.1. Transitions

Mit dem Transitions-Modul[184] wird eine völlig neue Philosophie in den Standard implementiert. Mit den Styling- und Layoutmöglichkeiten von CSS war es bisher nur möglich, einen statischen Ist-Zustand zu gestalten.[185] Wenn sich normalerweise eine CSS-Eigenschaft ändert, beispielsweise durch veränderte Werte bei `:hover`, wird das Resultat sofort angezeigt. Mit Transitions wird Entwicklern nun ein Werkzeug gegeben, mit dem sie Eigenschaftsänderungen über einen gewissen Zeitraum, Übergänge von Zustand A zu Zustand B definieren können. Diese Philosophie des verhaltensgesteuerten CSS ist eine Neuerung – war es doch bisher die Aufgabe von Java-Script, statische Seiten mit erweiterter Frontend-Funktionalität wie Echtzeit-Anzeigeeffekten und Animationen auszustatten.[186]

Grundlage für das Ausführen einer CSS-Transition ist eine Nutzerinteraktion – Übergange werden erst initialisiert, wenn sich etwas ändert, beispielsweise durch Klicken eines Elements oder Darüber Hovern. Sie treten also erst in Kraft, wenn die Zustands-Pseudoklassen aktiviert werden.

Das Modul gliedert sich in die 3 Eigenschaften, die dem zu animierenden Element übergeben werden

- `transition-property`: Welche CSS-Eigenschaft soll animiert werden
- `transition-duration`: Wie lange soll der Übergang von Zustand A zu Zustand B dauern
- `transition-timing-function`: Angabe einer sog. Easing-Funktion, Hiermit kann die Anfangs- und Endgeschwindigkeit des Übergangs gesteuert werden (z.B. Abbremsen)[187]
- transition-delay: Angabe einer Zeit-Verzögerung des Übergangs, sie erlaubt ein verspätetes Starten der Transition.

Alle 4 Eigenschaften können in der Kurzschreibweise `transition` vereint werden. Alle transition-Eigenschaften erlauben die Eingabe einer kommagetrennten Liste von Werten, wodurch mehrere Übergänge auf verschiedene Eigenschaften zu verschiedenen Zeitpunkten angewandt werden können.

---

[184] Transitions – engl. Übergänge
[185] Abgesehen von Pseudoklassen wie `hover`
[186] Vgl. Abb. 4 „Das 3-Ebenen-Modell"
[187] Genaueres siehe [W3C 022]

```
.transitionbox {
width:120px; 1
height:120px; 2
background:#666; 3
                    4          5          6
-webkit-transition: width 2s, height 2s, background-color 1s;
}
```

Q38: CSS3-Transition Initialisierung

Die `transition`-Eigenschaften werden immer auf den Ist-Zustand des Elements deklariert – also innerhalb des normalen Selektors[188] ohne Zustands-Pseudoklasse. Dieser stellt den Startpunkt des Übergangs dar, somit sind seine Werteangaben 1 2 3 auch immer die Startwerte.

Der hier aufgeführten Box werden 3 mögliche Übergänge zugewiesen:
Einmal eine Breitenveränderung innerhalb 2s 4, sowie eine Höhenänderung innerhalb 2s 5. Außerdem kommt eine Änderung der Hintergrundfarbe innerhalb 1s hinzu 6. Alle anderen Eigenschaften der Box werden sich nicht ändern.
Diese Art der Notation hat den Vorteil, dass im Folgenden mehrere Endpunkte durch CSS-Anweisungen mit verschiedenen Zustands-Pseudoklassen angegeben werden können, die von den vorhin festgelegten Transition-Eigenschaften in die neuen Werte ihrer Eigenschaften übergehen.

Für die Hover-Transition soll die Breite auf 300px und die Höhe auf 300px übergehen:

```
.transitionbox:hover {
      width:300px;
      height:300px;
}
```

Q39: Transition für die Hover-Pseudoklasse: Übergang von 120px/120px (Vgl. Q38) auf 300px/300px

[188] im Beispiel Q38: `.transitionbox`

Das Ergebnis gestaltet sich wie folgt:

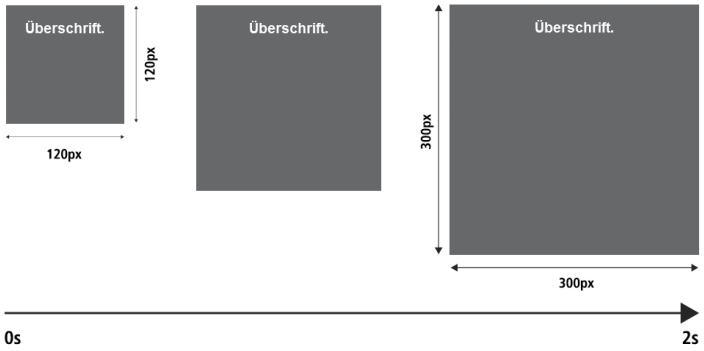



Abb. 63: Das Ergebnis der Transition aus Q39

Nachdem der durch die Pseudoklasse definierter Zustand aufhört oder er unterbrochen wird, wechselt das Element innerhalb der gleichen Dauer in den Anfangszustand zurück.

**Kombination mit den Transformations-Eigenschaften:**

Insbesondere die Kombination mit den 2D- und 3D-Transformationen[189] lässt die Stärken von „Transitions" erkennen.

```
#transformation {
      -webkit-transition: -webkit-transform 1s ease-in;
}

#transformation:hover {
      -webkit-transform: rotate(180deg);
}
```

Q40: Transition in Kombination mit den CSS-Transformationen

Einem Element wird als Übergangs-Eigenschaft `transform` übergen.
Die Transition soll 1s dauern und mittels der `ease-in` Funktion langsamer beginnen.[190]

Die Transition soll bei Hover über dem Element aktiv werden, wodurch sich das Element um 180° im Uhrzeigersinn dreht.

[189] Vgl. 3.2.6.
[190] Vgl. [W3C 022]

Das Ergebnis gestaltet sich wie folgt:

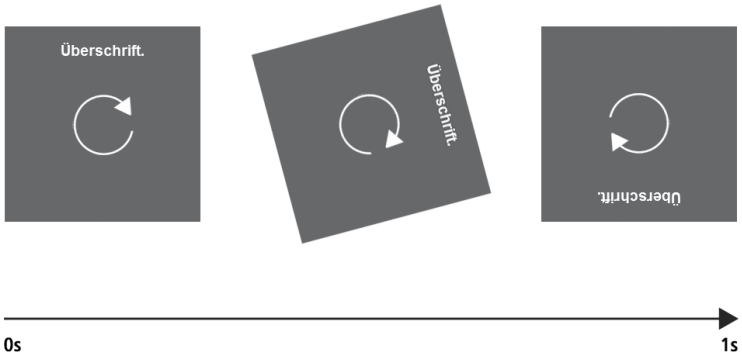



Abb. 64: 180° Drehung eines Elements mit Transition in Kombination mit Transform

Übergänge im dreidimensionalen Raum geben Freiheiten für interessante Nutzererfahrungen, jenseits von flachen, rechteckigen „Boxen".
So können greifbar wirkende Navigationsstrukturen oder die Visualisierung von gezeichneten Karten oder interaktiven Übersichts-Plänen als informationstragende Mittel eingesetzt werden. Mit Transitions können auch gegenwärtige User-Interface-Lösungen, wie Klick-Galerien[191] oder UI-Akkordions mit Slide-Effekten erstmals ohne den Einsatz von Java-Script oder Flash realisiert werden.

Die Kombination liefert die Werkzeuge um wirklich innovative und einfache zu handhabende Designlösungen zu entwickeln, ohne auf die Vorteile eines CSS-Layouts (Graceful Degradation[192], Skalierbarkeit, Textmarkierung) verzichten zu müssen.
Allerdings bergen sie auch die Gefahr des unsinnigen, übermäßigen und störenden Einsatzes, der dem Nutzer keinen Mehrwert bringt, sondern eher von der eigentlichen Information ablenkt. Auf eine ausführliche Demonstration wird der Autor hier verzichten. Einige experimentelle und innovative Demonstrationen befinden sich unter folgendes Links

- 3D Animation nur mit CSS - Film Poster:
  http://demo.marcofolio.net/3d_animation_css3/

- Poster Circle - ein komplexes 3D Gebilde
  http://webkit.org/blog-files/3d-transforms/poster-circle.html

[191] Beispiel siehe http://www.nealgrosskopf.com/tech/thread.php?pid=45
[192] Vgl. 4.3.

# 4. CSS3 Praxis

Das folgende Kapitel wird näher beleuchten, inwiefern die CSS3 Module schon praxistauglich sind und welche Kriterien für den Einsatz des Standards von Nöten sind. Er liefert einen Einblick darüber, welchen Problemen der Webstandard momentan noch gegenübersteht.

## 4.1. Der Stand der Module

Blickt man auf den W3C-Status[193] der in Abschnitt 3.2. vorgestellten Module und Eigenschaften, fällt auf, dass zum jetzigen Zeitpunkt der größte Teil von ihnen den Working Draft-Status[194] besitzt. Das ist gleichbedeutend mit einem Arbeitsentwurf, an dem sich nach dem Review durch die W3C-Mitglieder noch einiges ändern kann, ehe es zu einer Candidate Recommendation[195] kommt.

| Name des Moduls | Letztes Änderungsdatum | Status |
|---|---|---|
| Paged Media | 10.10.06 | Last Call |
| Values and Units | 19.09.06 | Working Draft |
| CSS Text | 06.03.07 | Working Draft |
| Backgrounds & Borders | 17.12.09 | Last Call |
| Fonts | 18.06.09 | Working Draft |
| Box Model | 09.08.07 | Working Draft |
| Multi-Column Layout | 17.12.09 | Candidate Recommendation |
| Template Layout | 29.04.10 | Working Draft |
| Media Queries | 15.09.09 | Candidate Recommendation |
| Basic User Interface | 11.05.04 | Candidate Recommendation |
| Grid Positioning | 05.09.07 | Working Draft |
| Flexible Box Layout | 23.07.09 | Working Draft |
| CSS Image Values | 23.07.09 | Working Draft |
| 2D Transformations | 01.12.09 | Working Draft |
| 3D Transformations | 20.03.09 | Working Draft |
| Transitions | 01.12.09 | Working Draft |
| Animations | 20.03.09 | Working Draft |

Tabelle 6: Die CSS3-Module und ihr derzeitiger Status

[193] [W3C 001]
[194] siehe A1 „The Maturity Levels"
[195] Kandidat Empfehlung - siehe A1

Die Kandidat-Empfehlung bzw. den „Last Call“ zur Kandidat-Empfehlung besitzen bisher nur wenige der in dieser Bachelorarbeit vorgestellten Module. Diese stehen also kurz vor ihrer Empfehlung und sind dementsprechend ausgereift.
Alle anderen (hier vorgestellten) Module und Eigenschaften sind noch mehr oder weniger stark im Bestehen begriffen.[196]
Darunter fallen fast alle für Layoutzwecke benötigten Module wie das Template Layout, das Flexible Box Layout, das Box Model, die Fonts und vor allem das Grid Positioning Modul.

Einzig das Multi-Column-Layout, die Backgrounds & Borders, die Media Queries und Paged Media besitzen schon den stabilen CR-Status und können in den Browsern, die sie untersützen, schon relativ sorglos verwendet werden.
Blickt man auf die Daten der letzten Änderungen der Working Drafts, fallen teilweise Zeiträume von über einem Jahr auf, in denen nichts an der Modulspezifikation geändert wurde.[197]

Gründe für das zähe Fortschreiten der Modulentwicklungen sind manigfaltig und nicht immer technischer Natur:
Langsame Reviews, noch nicht 100% stabile Konzepte[198], fehlende Interoperabilität mit anderen Modulen, Organisationsprobleme / Uneinigkeiten innerhalb der jeweiligen Arbeitsgruppe[199] eigene Interessenpolitik der beteiligten Unternehmen oder harsche Kritik von außen[200] sind einige der Gründe, die die jetzige Situation verursacht haben könnten. Als außenstehender Beobachter ist diese Lage schwierig einzuschätzen und vieles ist spekulativer und subjektiver Natur und damit nicht Teil dieser Arbeit.

Fakt ist, dass CSS3 als kompletter Standard noch einige Zeit brauchen wird bis er seine vollständige Ausgereiftheit erlangt. Bert Bos, einer der führenden Entwickler des CSS-Teams spricht 2007 von 4-5 Jahren, bis eine vollständige Implementierung erreicht sein könnte. Alle Module, die bis dahin nicht implementiert werden, werden laut ihm radikal aufgegeben.[201]
Dies ist ein sehr radikaler Schnitt und könnte gerade den innovativen Layoutmechanismen, die mit Grid Positioning und dem Template Layout entwickelt wurden, das Genick brechen. Doch andererseits nützt ein gutes Konzept nichts, so lange es von keinem Browser implementiert werden kann.

---

[196] Vgl. Tabelle - dunkelgraue Hinterlegung
[197] Oder zumindest nicht veröffentlicht wurde
[198] [CSS3 TEAM 001]
[199] [HIXIE] „The CSS Working Group is irrelevant“
[200] [WILCOX] [RUSSELL] [BUDD]
[201] [CSS3 TEAM 001] „[...] Over the next four or five years most of the modules will become implemented, I think, and the rest we will drop definitively. I can’t tell you which ones, though…“

## 4.2. Der Stand der Browser

An dieser Stelle zeigt die Eingangs beschriebene Modularisierung des Standards ihre Vorteile[202]. Stabile (CR) CSS3-Merkmale, wie das Backgrounds & Borders Modul oder das Multicolumn-Modul sind über die Browser Prefixes schon in allen modernen A-Grade Browsern[203], bis auf den Internet Explorer 8 nutzbar. Und wie in den Praxisbeispielen im Kapitel 3 aufgezeigt, gibt es auch einige experimentelle Working-Draft-Eigenschaften, die ihren Weg in die Browser gefunden haben.
Den Browserherstellern wächst, wie auch bei den Vorgängerversionen des Standards, eine immense Bedeutung für das Fortschreiten von CSS zu. So sind sie es, die mit der CSS3 Test Suite und einer zügigen standardkonformen Implementation die CSS3-Merkmale auf Herz und Nieren testen und es damit den Webentwicklern als anwendbares Werkzeug bereit stellen.

Wie im Kapitel „CSS-Probleme"[204] erläutert, hatte CSS 2.1. erst Ende 2009, mit der Veröffentlichung des IE8, eine flächendeckende Nutzung erfahren. Auch hinsichtlich der CSS3-Spezifikation wird dem Internet Explorer und seiner kommenden Version 9 die größte Bedeutung zu teil. Der Internet Explorer 8 unterstützt bis auf die `@font-face`-Regel kein einziges CSS3-Merkmal und auch keinen der CSS3-Selektoren.[205]
Apple's Safari-Browser bzw. sämtliche auf die Webkit-Engine aufsetzende Browser[206] weisen zum Zeitpunkt der Arbeit die umfassendste CSS3-Unterstützung auf. Alle neuen CSS3-Selektoren, alle grafischen und dekorativen Eigenschaften, das Multi-Column-Layout, 2D-Transformationen und Transitions, `@font-face` und die funktionalen Farbangaben werden unterstützt.[207] Der Safari-Browser ist momentan auch der einzige Browser, der 3D-Transformationen implementiert hat.
Die Gecko-Engine (Firefox) beweist ebenfalls guten Support, es fehlen nur Animationen, Reflektionen und Transformationen. Opera liefert eine Unterstützung für alle als CR freigegebene Module und ignoriert zur Zeit noch alle Merkmale, die den Status „Working Draft" besitzen, da sich diese noch deutlich ändern könnten. Anzumerken ist hierbei, dass sämtliche Eigenschaften noch mit dem jeweiligen Browser-Prefix angegeben werden müssen.[208]

[202] Vgl. 3.1.
[203] [YAHOO DEVELOPERS]
[204] Vgl. 2.6.
[205] Vgl. Abb66-67 [FINDMEBYIP] [MICROSOFT 001]
[206] [WEBKIT] u.A. Chrome, Safari,
[207] Vgl. Abb. 66-67
[208] Vgl. Q-Anwendungsbeispiele 3.2.

| | MAC | | | | | WIN | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | CHROME | FIREFOX | OPERA | SAFARI | | CHROME | | FIREFOX | | OPERA | SAFARI | IE | | |
| | 5 | 3.6 | 10.6 | 5 | 4 | 5 | 6 | 3.6 | 4.02 | 10.6 | 5 | 6 | 7 | 8 |
| RGBA | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✗ | ✗ | ✗ | 78% |
| HSLA | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✗ | ✗ | ✗ | 78% |
| Background Size | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✗ | ✗ | ✗ | 73% |
| Multiple Backgrounds | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✗ | ✗ | ✗ | 73% |
| Border Image | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✗ | ✗ | ✗ | 77% |
| Border Radius | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✗ | ✗ | ✗ | 78% |
| Box Shadow | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✗ | ✗ | ✗ | 77% |
| Opacity | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✗ | ✗ | ✗ | 78% |
| CSS Animations | ✓ | ✗ | ✗ | ✓ | ✓ | ✓ | ✓ | ✗ | ✗ | ✗ | ✓ | ✗ | ✗ | ✗ | 33% |
| CSS Columns | ✓ | ✓ | ✗ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✗ | ✓ | ✗ | ✗ | ✗ | 74% |
| CSS Gradients | ✓ | ✓ | ✗ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✗ | ✓ | ✗ | ✗ | ✗ | 69% |
| CSS Reflections | ✓ | ✗ | ✗ | ✓ | ✓ | ✓ | ✓ | ✗ | ✗ | ✗ | ✓ | ✗ | ✗ | ✗ | 33% |
| CSS Transforms | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✗ | ✗ | ✗ | 76% |
| CSS Transforms 3D | ✗ | ✗ | ✗ | ✓ | ✗ | ✗ | ✗ | ✗ | ✗ | ✗ | ✓ | ✗ | ✗ | ✗ | 9% |
| CSS Transitions | ✓ | ✗ | ✓ | ✓ | ✓ | ✓ | ✓ | ✗ | ✓ | ✓ | ✓ | ✗ | ✗ | ✗ | 39% |
| CSS FontFace | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | 94% |

Abb. 66: CSS3-Browserunterstützung[209] IE ist weit abgeschlagen

| | MAC | | | | | WIN | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | CHROME | FIREFOX | OPERA | SAFARI | | CHROME | | FIREFOX | | OPERA | SAFARI | IE | | |
| | 5 | 3.6 | 10.6 | 5 | 4 | 5 | 6 | 3.6 | 4.02 | 10.6 | 5 | 6 | 7 | 8 |
| CSS3: Begins with | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✗ | ✓ | ✓ | 97% |
| CSS3: Ends with | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✗ | ✓ | ✓ | 97% |
| CSS3: Matches | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✗ | ✓ | ✓ | 97% |
| CSS3: Root | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✗ | ✗ | ✗ | 80% |
| CSS3: nth-child | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✗ | ✗ | ✗ | 78% |
| CSS3: nth-last-child | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✗ | ✗ | ✗ | 78% |
| CSS3: nth-of-type | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✗ | ✗ | ✗ | 78% |
| CSS3: nth-last-of-type | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✗ | ✗ | ✗ | 78% |
| CSS3: last-child | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✗ | ✗ | ✗ | 80% |
| CSS3: first-of-type | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✗ | ✗ | ✗ | 78% |
| CSS3: last-of-type | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✗ | ✗ | ✗ | 78% |
| CSS3: only-child | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✗ | ✗ | ✗ | 80% |
| CSS3: only-of-type | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✗ | ✗ | ✗ | 78% |
| CSS3: empty | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✗ | ✗ | ✗ | 80% |
| CSS3: target | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✗ | ✗ | ✗ | 79% |
| CSS3: enabled | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✗ | ✗ | ✗ | 80% |
| CSS3: disabled | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✗ | ✗ | ✗ | 80% |
| CSS3: checked | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✗ | ✗ | ✗ | 79% |
| CSS3: not | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✗ | ✗ | ✗ | 80% |
| CSS3: General Sibling | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✗ | ✓ | ✓ | 97% |

Abb. 67: CSS3-Sektoren Browserunterstützung[210] IE ist weit abgeschlagen

[209] Quelle: [FINDMEBYIP] übernommen
[210] Quelle: [FINDMEBYIP] übernommen

Auch Microsoft hat mit der ersten Testversion des Internet Explorer 9 eine HTML5/ CSS3-Unterstützung angekündigt und tut im Hinblick auf die schrumpfenden Marktanteile gut daran. [211]
Der Platzhirsch[212] positioniert sich wieder als Browser, der Webstandards ernst nimmt. Mit der öffentlich zugänglichen Testversion und einer Website[213], die die Entwicklung des Browsers und seine Testergebnisse transparent dokumentiert, scheint Microsoft aus den Fehlern der Vergangenheit[214] gelernt zu haben. Die Ergebnisse des ersten Platform Previews fallen allerdings noch eher ernüchternd aus[215]. Die Betaversion des IE9 unterstützt zwar alle CSS3 Selektoren, aber bis auf `rgba()` und `@font-face` keine weiteren CSS3-Merkmale. Der IE9 scheint also nach bisherigem Kenntnisstand wieder deutlich der Konkurrenz hinterher zu hängen. Hoffnung gibt die neu erschienene dritte Platform Preview[216], die zumindest einige der CR-Module von CSS3 unterstützen soll[217] Für die Veröffentlichung bleibt zu hoffen, dass wenigstens die CSS3-Merkmale mit CR-Status komplett implementiert werden. Passiert dies nicht, wird sich die Zeitverzögerung einer wirklich browserübergreifenden Implementation der CSS-Eigenschaften noch weiter erhöhen und Javascript-Übergangslösungen, die im nächsten Abschnitt beschrieben werden, werden die einzige Möglichkeit sein, zumindest einen Teil der Module browserübergreifend benutzen zu können.

Der Wettlauf der Browser um die schnellste CSS3-Implementation hat begonnen und Webkit hat in punkto Quantität und Qualität die Nase vorn[218]. Eine zügige Nachrüstung mit neuen CSS3-Merkmalen bringt die gesamte CSS3-Entwicklung voran, sie holt aktuelle Merkmale sofort in die Praxis und sie setzt die Browser-Konkurrenz, wie auch die CSS-Arbeitsgruppe des W3C unter Druck – ein Druck, der aber nicht nur positiv ausfallen kann. Es entsteht das Risiko eines zweiten „Browserkrieges“[219] um die scheinbar beste CSS3-Rendering-Engine. Durch die Browser-Prefixes haben die Hersteller sämtliche Freiheiten, um auch nicht-standardkonforme Lösungen zu implementieren, falls ihnen bestimmte Spezifikationen nicht schnell genug nachrücken oder einfach nicht gefallen. So ähnlich ist es schon bei der Implementation der „Gradients“ in Safari und Firefox geschehen.[220] Während Firefox die standardkonforme Lösung verwendet, setzt Safari auf eine eigene Interpretation.
Da aber ein reger Austausch zwischen W3C und den Browser-Entwicklern stattfindet, bleibt dieses Szenario mit Sicherheit Fiktion.

---

[211] Vgl. 2.5.1.
[212] 60& Marktanteil, Vgl. Abb 5
[213] [MICROSOFT 002]
[214] Vgl. 2.5.1.
[215] Vgl. Abb 68
[216] 04.08.2010 [MICROSOFT 003]
[217] Fonts, Media Queries, Background & Borders, Namespaces, Selectors, Vgl. [MICROSOFT 003]
[218] Vgl. Abb 66, 67
[219] Vgl. 2.5.1
[220] Vgl. Q28

## 4.3. CSS3 in der Praxis

Aus den vorhergehenden Abschnitten wird ersichtlich, dass CSS3 zum jetzigen Zeitpunkt nicht komplett übergreifend gleich stabil angewandt werden kann. Während einige Browser, wie Safari 5, schon eine recht beachtliche Zahl von Merkmalen unterstützen, fallen andere, wie der aktuelle IE8, komplett heraus. In dieser Zeit des Übergangs stellt sich die Frage, ob eine Website wirklich in jedem Browser exakt gleich aussehen muß bzw. kann[221]. Oder ob es nicht sinnvoller ist, ein gewisses Basis-Layout zu erstellen, das die Informationszugänglichkeit, den Grundaufbau und die grundlegende Nutzbarkeit in schwächeren Browser-Engines erhält, aber moderne Browser nicht daran hindert, moderne und innovative Features zu nutzen.
Die Antwort liefern die praktischen Entwicklungsmethoden des Progressive Enhancement[222] und der Graceful Degradation[223], mit denen das Nutzen von Neuerungen ermöglicht wird, ohne aber ältere Browser zu vernachlässigen.

### 4.3.1. Progressive Enhancement

Das Design fokussiert sich auf den Inhalt von Webseiten und seiner uneingeschränkten Zugänglichkeit mit älteren Browsern. Inhalt und Grundaufbau einer Website werden mit semantisch wertvollem HTML-Markup und nur den wesentlichen browserübergreifend funktionierenden CSS-Eigenschaften versehen, und dann Stück für Stück mit fortgeschrittenen Techniken ausgestattet, die nur von moderneren Browsern genutzt werden. So wird garantiert, dass das Wesentliche einer Webseite – ihr Inhalt – immer zugänglich ist, auch wenn das Aussehen und der „Style" in schwächeren Engines weniger präsent ist.[224]

PE bietet sich für den Praxiseinsatz von teilweise implementierten CSS3-Eigenschaften an. Dekorative Eigenschaften, wie multiple Hintergrundbilder, abgerundete Ecken oder Schlagschatten sind nicht die wesentlichen Faktoren einer Website.
Sie bereichern die Nutzererfahrung und helfen Informationen hervorzuheben und sie besser und angenehmer zu visualisieren. Mit dieser Methode des PE wird auf aufwändige und zeitraubende „Browser-Hacks", Filter und JS-Plugins für CSS3-Unterstützung, die älteren Browsern ein Verhalten einzuimpfen versuchen, bewusst verzichtet[225].

Die Charakteristik eines jeden Browsers wird in die Entwicklung einbezogen – und wenn eine bestimmte Eigenschaft nicht unterstützt wird, wird sie eben einfach weg-

[221] [WEBKRAUTS] [DOWEBSITES...]
[222] engl.: Fortschreitende Erweiterung
[223] engl.: Kontrollierter Rückfall - Teilausfall
[224] [ALISTAPART 002]
[225] Natürlich darf Java-Script verwendet werden, allerdings nicht für die versuchte Implementation von CSS3, sondern nur für optionale Funktionalität

gelassen und auf das Basis-CSS zurückgegriffen. Die Erweiterung der technischen Rafinesse von unten nach oben erfolgt rein cssbasiert und nur unter Zuhilfenahme der „Conditional Comments“[226], die die zuerst erstellten Basis-CSS-Anweisungen für ältere IE-Versionen liefern und die CSS3-Anweisungen für diese Versionen überschreiben.[227]

**Vorteile:**

- einfache Wartbarkeit (Layout wird Stück für Stück weiterentwickelt)
- schlanker, valider Quellcode
- weniger Server-Abfragen, geringere Ladezeiten
- Barrierefreiheit und Informationszugänglichkeit
- Zukunftssicher, da sich in der Weiterentwicklung nur noch auf neue Implementationen konzentriert werden muß (Alte Browser verwenden immer das angedachte Basis-Layout)

**Nachteile:**

- unkonsistente Wirkung der Webseite auf verschiedenen Browsern – u.U. bis hin zu einem verwässerten Wiedererkennungswert
- scheinbar „billige“ Anmutung auf alten Browsern


Abb. 69: Progressive Enhancement in der Anwendung. Erweiterung der technischen Raffinesse von unten nach oben.[228]

[226] Browserweichen [SELFHTML 001]
[227] [STARR]
[228] Quelle [DEV OPERA]

### 4.3.2. Graceful Degradation

Einen etwas anderen Ansatz zur Anpassung von CSS-Layouts an verschiedene Versionen verfolgt die Methode des „kontrollierten Rückfalls“ bzw. der Teilausfälle.
Hierbei wird ein CSS3-Layout für moderne Browser standardgemäß umgesetzt.
Älteren Browser wird mit Hilfskonstrukten wie Java-Script-Lösungen[229] , Direct-X-Filtern oder der Einbindung von HTML-Komponenten[230] (htc-Dateien) die gewünschte Funktionalität eingebaut. Dabei ist zu beachten, dass bei Teilausfällen der gewählten „Workarounds“ (z.B. deaktiviertes Java-Script), die Seite immer noch nutzbar bleibt – die Webseite fährt sich gewissermaßen stückweise zurück.
Im Gegensatz zu PE verwendet GD eine Entwicklung von oben nach unten. Die Charakteristik bestimmter Browser wird mit den beschriebenen Hilfskonstrukten verändert.

m Gegensatz zu PE verwendet GD eine Entwicklung von oben nach unten.
Die Charakteristik bestimmter Browser wird mit den beschriebenen Hilfskonstrukten verändert.

**Vorteile dieser Methode:**
- CSS3-Merkmale können in älteren Browsern verwendet werden
- browserübergreifend relativ gleich aussehende Webseiten
- Bei Teilausfällen bleibt die Seite weiterhin nutzbar.

**Nachteile:**
- höherer Zeit- und Code-Aufwand bei der Erstellung
- schwieriges Überblicken des Verhaltens aller Browserversionen
- teils schwierige Wartbarkeit / Zukunftssicherheit (neue Implementation verlangen auch wieder neue Hilfskontrukte für alte Browser)
- aufgeblähter – teilweise invalider - Quellcode

[229] [MODERNIZR]
[230] [W3C 022]

### 4.3.3. Anwendung

Als überblickendes Beispiel für die Praxis-Anwendung dieser beiden Konzepte soll eine rein CSS3-basierte Umsetzung der apple.com Haupt-Navigation dienen. Sie lässt sich mit den bereits implementieren CSS3-Eigenschaften nahezu perfekt und im Sinne des Progressive Enhancement umsetzen.


Abb. 70: Die apple.com Hauptnavigation - Im Folgenden wird diese rein css-basiert umgesetzt

Zur Zeit verwendet die Hauptnavigation auf apple.com ein typisches CSS2.1 Horizontal-Layout, das auf den Einsatz von Hintergrundgrafiken basiert. Es werden sog. Image-Sprites verwendet, die allen Stadien der Navigation innehaben und deren Teile über die `background-position` Eigenschaften den jeweiligen Navigationspunkten und ihren Hover und Aktiv-Stadien zugewiesen werden.


Abb. 71: Die Image-Sprites der Navigationshintergrundgrafik

Ziel ist es nun, die Navigation mit den in 4.3. erläuterten Konzepten und ohne Hintergrundgrafiken[231] umzusetzen.

Zum Einsatz kommen die bereits in Webkit-Engine implementierten CSS-Merkmale[232]:

- Border-Radius
- CSS Gradients
- @font-face / Webfonts
- Box-Shadow und Text-Shadow

Das Markup wird unter der HTML5[233] Spezifikation erfolgen. Als Minimum unterstützenswerter Browser wird der IE6 festgelegt.

[231] bis auf as Apple Logo und das Such-Icon
[232] Vgl. Abb 66
[233] siehe A4 „HTML5 - ein Überblick"

Zunächst wird das HTML5-Markup standardkonform aufgezogen:

```
<!doctype html>
<html>
<head>

<!-- Eric Meyers Reset CSS --> 1
<link rel="stylesheet" type="text/css" href="reset.css" />
<link rel="stylesheet" type="text/css" href="basic_css.css" />
                                                    2
<link rel="stylesheet" type="text/css" href="advanced_css.css" />
                                                    3

<title>apple.com mit HTML5 und CSS3</title>
</head>

<body>

<!-- Header Element -->
<header>
      <!-- Navigations-Element -->
      <nav>
          <ul>
           <li id="apple"><a href="#">Home</a></li>
            <li><a href="#">Store</a></li>
            <li><a href="#">Mac</a></li>
            <li><a href="#">iPhone</a></li>
            <li><a href="#">iPad</a></li>
            <li><a href="#">iTunes</a></li>
            <li><a href="#">Support</a></li>

          </ul>


      </nav>

    <!-- Suchmaske -->
      <form id="search" action="index.php">
            <input type="text" value="Suchen..." />
      </form>

</header>


</body>
</html>
```

Q41: Das HTML5-Markup des Anwendungsbeispiels

Als Erstes wird das sog. „Reset-Stylesheet" von Eric A. Meyer[234] 1 initialisiert. Es sorgt dafür, dass die browserspezifisch unterschiedliche Standard-CSS-Interpretation von Elementen in allen Browsern gleichgesetzt bzw. auf 0 gesetzt wird. Dies läuft der PE-Philosophie nicht entgegen, eher im Gegenteil. Es ist eine reine CSS-Lösung, um alle Browser an den gleichen Ausgangspunkt zu setzen.

Dann wird das Basis-Stylesheet 2 für alle Browser entworfen. Dies sollte so grundlegend wie möglich gehalten werden, damit alle Browser es ähnlich gut implementieren. Die Navigationsliste erhält floatende Listenpunkte mit festen Dimensionen, die rein über die Farben ihren aktiven/Hover Status anzeigen. Das Suchformular wird ebenfalls links daneben gefloatet. Das Basis-CSS ist fertig, wenn es in allen anfangs abgesteckten Browserversionen funktioniert:[235]


Abb. 72: Die Anwendung des Basis Stylesheets in verschiedenen Browsern

Mit einem weiteren verlinkten Stylesheet, das die fortschrittlichen Stilanweisungen enthält (advanced_css.css) 3, werden nun die grundlegenen Deklarationen erweitert und teils überschrieben.[236] Es werden spezifische Angaben für abgerundete Ecken, Schrifteinbettung, Text- und Box Shadows und natürlich die unterschiedlichen Verlaufsfüllungen für die Hover/Aktiv-Stadien der Navigationselemente angelegt.[237]

Alle älteren IE Versionen parsen zwar ebenfalls dieses Stylesheet, ignorieren aber die neuen Anweisungen komplett. Bei diesem einfachen Beispiel ist es nicht von Nöten, mittels Conditional Comments das Basis-Stylesheet nur für bestimmte IE-Versionen zugänglich zu machen. Es ist aber natürlich sinnvoll mit CC zu arbeiten, wenn es um komplexere Implementationen geht, wie die CSS3 Layoutsysteme[238] .

Dadurch können älteren IE-Versionen eine vereinfachte Float-Variante nutzen, während moderne Browser z.B. auf das Template Layout Modul zurückgreifen.

[234] [MEYER 001]
[235] siehe A5 für das Basis-Stylesheet
[236] Das „Advanced Stylesheet" befindet sich in A6
[237] Im Detail: siehe A6
[238] wenn sie denn irgendwann einmal funktionieren Vgl. 4.1.

Das Ergebnis der Anwendung des „Advanced Stylesheets" gestaltet sich nun wie folgt:

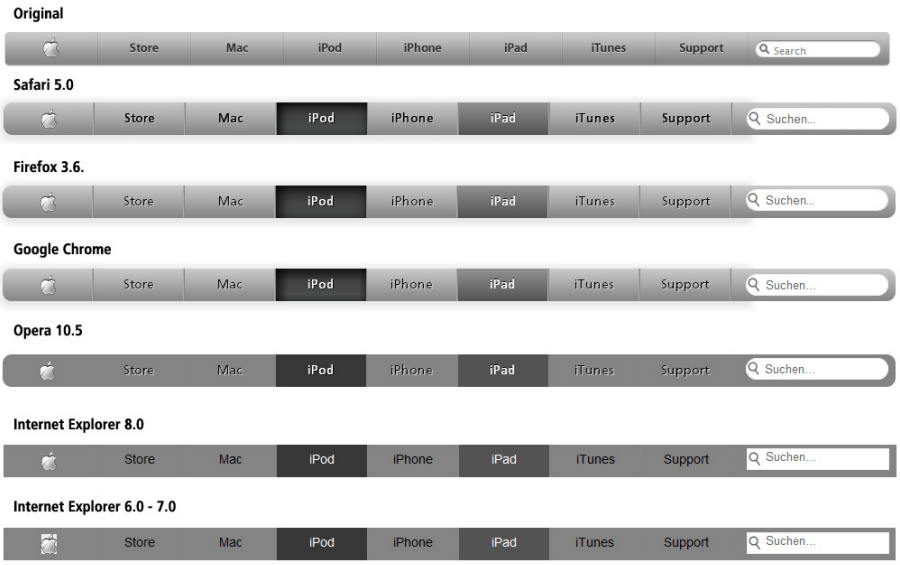



Abb. 73: Ergebnis des Beispiels aus 4.3. auf verschiedenen Browsern

Es zeigt sich, dass sämtliche Browser, die schon einige CSS3-Merkmale implementieren, eine deutlich erweiterte Version darstellen. Opera verwendet zumindest die als CR freigegebenen Eigenschaften `border-radius` und `text-shadow`. IE6-8 verwenden das Basis-Stylesheet. Trotzdem ist durch das Verwenden des Basis-Stylesheets die Navigation in IE6-8 ohne Probleme nutzbar, übersichtlich und hat zumindest einen grundlegenden Wiedererkennungswert.

Als weiteres Resultat zeigt sich, dass eine moderne Navigation, die sonst nur über die Einbindung von Grafiken erstellt werden konnte, komplett auf CSS3 basierend umgesetzt werden kann. Dies hat den Vorteil, dass bei Änderung oder Ergänzung eines Navigationpunktes, keine Grafiken aufwändig neu gestaltet oder ausgetauscht werden müssen. Die Änderung erfolgt im HTML und mit minimalen Anpassungen[239] in der CSS-Datei.

Die Frage ist bloß, ob es sinnvoll ist, ca. 60% der Nutzer eine solch abgespeckte, relativ unästhetisch[240] wirkende Version zu präsentieren.[241] Dies kann sich deutlich auf die Außenwirkung des Internetauftrittes auswirken und u.U. Nutzer vergraulen. Die Lösung heißt: Graceful Degradation.[242] In der Praxis bietet es sich an, beide Konzep-

[239] bei neuen Menüpunkten mit SIcherheit Änderung der `width`
[240] für Apple-Verhältnisse
[241] Vgl. Abb5, IE = 60% Marktanteil
[242] Vgl. 4.3.2

te[243] zu kombinieren, um ein bestmögliches Ergebnis zu erhalten und die Vorteile beider Konzepte zu nutzen.

Erst wird eine Basis-Version einer Website für ältere Browser entworfen, die anschließend Stück für Stück mit CSS3-Neuerungen erweitert wird. (PE)[244]

Um dann trotzdem auch in den eigentlich nicht unterstützten Browsern auf CSS3-Merkmale zurückgreifen zu können, kann auf zwei Javascript-Plugins zurückgegriffen werden – „Modernizr"[245] und „IE-7.js2"[246] Beide parsen das Stylesheet nach CSS3-Anweisungen und rendern diese entsprechend.
Damit können auch den IE-Versionen grafische und dekorative Eigenschaften, wie box-shadow, text-shadow, linear-gradients und border-radius relativ problemlos übergeben werden, womit sie ähnliche Ergebnisse erzielen, wie die Webkit/Gecko-Engines.

Der zweite Vorteil der Kombination beider Philosophien wird an dem Zeitpunkt verständlich, an dem aus Teilausfällen Totalausfälle werden. Sollten die eingebundenen Java-Scripte nicht funktionieren[247] steht dann immer noch das mit PE entworfene Basis-Stylesheet zur Verfügung, federt das Schlimmste ab und liefert eine nutzbare und zugängliche Variante.

[243] Progressive Enhancement und Graceful Degradation
[244] Vgl. Abb 73
[245] [MODERNIZR]
[246] [IE-/ JS]
[247] beispielsweise durch deaktiviertes JS beim Nutzer

# 5. Fazit

CSS Level3 gestaltet sich mit seiner Vielzahl an durchdachten Modulen als ernstzunehmende Weiterentwicklung der mittlerweile 12 Jahre alten Vorgängerversion. Sämtlichen Unzulänglichkeiten[248] wurden ausgereifte Konzepte und Lösungen entgegengesetzt, die diese Probleme eliminieren und das Webdesign auf Basis von HTML/CSS zeitsparender und erleichternder gestalten. Insbesondere die neuen Konzepte für das globale Layouting[249] können den Webstandard zu einer wirklichen Layoutsprache mit perfekt miteinander kombinierbaren Layoutmechanismen entwickeln, die bekannte Float-Layouts und auch die veralteten Tabellen-Layouts in den Schatten stellen. Doch die eigentlichen Layout-Module sind derzeit noch theoretischer Natur. Eine Implementierung scheint sich schwierig zu gestalten.[250]
Zusammen mit den Media Queries und den erweiterten Selektionsmöglichkeiten, die zumindest in fortschrittlichen Browsern funktionieren, stellen die CSS3-Layoutsysteme nach Ansicht des Autors, den wichtigsten Punkt der Spezifikation dar.
Sie könnten wichtige Werkzeuge für die Zukunft liefern, um an verschiedene Medien angepasste Designs zu erstellen, die den Entwicklern einen großen gestalterischen Spielraum hinsichtlich der globalen visuellen Struktur von Nutzerinterfaces überlassen. Gerade der Boom-Markt für mobile Geräte und die damit einhergehenden Forderungen nach mobil angepassten Websites ließen sich mit den CSS3 Layoutmechanismen erfüllen.

Die grafischen und dekorativen Eigenschaften (Multiple Hintergrundbilder, Schlagschatten, Border-Images), die neuen funktionellen Wertetypen, die Alpha-Transparenz, die Webfonts und die Transformationen stellen ebenfalls wichtiges Rüstzeug, um CSS3 auf zeitgemäße Designaufgaben auszurichten. Sie holen CSS wieder in die Gegenwart und machen viele Aufgaben, die professionelle Screendesign-Software erforderten, direkt mit CSS möglich.
Weitere innovative Merkmale, wie die Transitions, mit denen sämtliche CSS-Eigenschaften animiert werden können, lassen Raum für kreative Einsatzmöglichkeiten, die ein Design außergewöhnlich machen. SIe bergen allerdings die Gefahr des mißbräuchlichen und übermäßigen Einsatzes, so wie es in den Anfangszeiten der Flash-Technologie zu sehen war.

Gleichzeitig entstehen zur Zeit eine Vielzahl kreativer und innovativer Anwendungsbeispiele von CSS3. Es herrscht eine selten da gewesene Aufbruchsstimmung in den Fachmedien und eine spielerische Auseinandersetzung mit den neuen Merkmalen.

Die Frontend-Webentwicklung mit CSS3 (und HTML5) befindet sich in einer typischen

---

[248] Vgl. 2.6.

[249] das Template Layout Modul, Flexbox sowie Multi-Column und Grid Positioning

[250] Vgl. 4.2.

Übergangssituation. Wie lange diese andauern wird, und wie diese ausgehen wird, hängt von unterschiedlichen Faktoren ab.

Die im dritten Kapitel vorgestellten Eigenschaften und Merkmale der Spezifikation sind zu einem annehmbaren Teil schon in den technisch fortschrittlichsten Browser-Engines Webkit und Gecko auf experimenteller Basis[251] einsetzbar.

Dazu gehören

- die neuen strukturbasierten Selektoren, die semantisch fragwürdige Klassenvergaben ad acta legen, Elemente zählbar machen, die Außer-Selektion und Nachbars-Selektion ermöglichen sowie allgemein interessante Usability-Möglichkeiten bewirken können.
- das alternative Box-Modell, das kompliziertes Errechnen der Dimensionen bei Boxen mit Padding und Borders überflüssig macht und exakt 50% breite Boxen in „Liquid Layouts" ermöglicht.
- Das Flexible Box Layout, mit dem sich flexibel aneinander orientierenden Boxen45 geschaffen werden, die relativ unabhängig vom HTML-Markup agieren und damit einen großen Schritt in Richtung wirklicher Trennung von CSS und Markup ermöglichen. Durch sie werden typische Probleme wie die vertikale Zentrierung, gleiche Spaltenhöhen, und die Ausrichtung an Ober-/Unter-/Mittelkanten von nebenstehenden Elementen elegant beseitigt
- Das Multi-Column-Layout, mit dem Inhalte in einen dynamischen Spaltenaufbau gebracht werden können, in denen die Inhalte ohne Einsatz von anderen Techniken von einer Spalte zur anderen fließen können
- alle grafischen und dekorativen Eigenschaften, wie die multiplen Hintergrundbilder, Border-Radien und Border-Images, Schlagschatten in jeglicher Form und Farbverläufe, die allesamt ein Design hochwertiger und detailreicher erscheinen lassen (können)
- die Farbangaben mit spezifischer Alpha-Transparenz und rgba und hsla Werten, mit denen ansprechende Transparenzeffekte für Elementteile ermöglicht werden
- die Einbettung von eigenen Schriften, die die typographische Monotonie im Web aufbrechen wird
- die 2D und 3D Transformationen, die in Verbindung mit den animierten Übergängen, ganze neue Ansätze im CSS-Design liefern
- und zu guter letzt die Media Queries, mit dem ein CSS-LAyout flexibel auf bestimmte Ausgabegeräte, insbesondere Mobiltelefone, abgestimmt werden kann

Die Realität – in Form der Browserstatistiken – spricht allerdings eine andere Sprache. Firefox 3.6., Safari 5 und Google Chrome 5.0 kommen zusammen auf einen Marktanteil von 30-40% - noch lange nicht genug um, alles, was CSS3 bis jetzt hergibt, in

[251] Browser-Prefixes

Live-Sites zu implementieren.
Wie im Praxisbeispiel in 4.3. aufgezeigt, spricht nichts dagegen, die grafisch/dekorativen Eigenschaften, die moderne Browser schon unterstützen, als trendgemäßes Design-„Addon" für diese zu verwenden, da diese nicht das wesentlichen Layout beeinflussen, sondern nur zusätzliche „Effekte" ins Spiel bringen.

Auch das Multi-Column ist durchaus für Textspalten einsetzbar, da bei Nichtunterstützung, der Text einfach gewohnt vertikal (über die komplette Breite) fließt. Den Unterschied werden die meisten Nutzer nicht feststellen können[252]

Insbesondere der Internet Explorer ist missionskritisch, da er einen erheblichen Anteil am Browser-Markt [253] besitzt.
Das Umdenken bei Microsoft hat definitiv angefangen, doch leider scheint auch der kommende IE9 bislang sehr wenige CSS3-Merkmale zu unterstützen und gehört mit dem IE8 zu den aktuellen Browsern, die ohne Java-Script oder Filterlösungen nicht wirklich CSS3-kompatibel sind. Der Abstand zwischen der CSS3-Unterstützung in fortschrittlichen Engines und dem IE-Versionen ist derzeit so groß, dass der Aufwand, dem IE8-9 bestimmte CSS3-Funktionalität „beizubringen" (div. JS-PLugins[254] ) oder die Seite auf ein annehmbares CSS2.1-Layout zurückzufahren, dem Nutzen noch nicht gerecht wird. Selbst mit den Methoden Progressive Enhancement und/oder Graceful Degradation[255] ist es noch zu zeitaufwändig und damit für Kunden zu budgetintensiv, mehrere angepasste Versionen eines CSS-Layouts zu entwerfen.

Die Modularisierung des Standards ist ein gutes Konzept, doch wenn nicht alle Browser mitziehen, leider auch ein Erfolgloses.
Hinzu kommt, dass nur ein kleiner Teil der so eben aufgeführten Module den relativ stabilen CR-Status aufweisen, d.h. Im schlimmsten Fall, ändern sich alle anderen Module im Working-Draft-Status komplett und sämtliche erstellten Webseiten müssen entsprechend angepasst werden.
Das W3C gerät durch diverse interne Probleme mit der Entwicklung des Standards in Zeitverzug, einige Teilspezifikationen haben sich seit 1-2 Jahren[256] kaum geändert. Dazu wird das Bedürfnis nach weiteren CSS-Merkmalen größer – so z.B. den CSS3 Variablen/Konstanten, für die es bis jetzt noch keine Lösung gibt.

So besteht derzeit eine stagnierende Abwartehaltung, die aus der sichergehenden Implementierung von nur wenigen – zwar schön anzuschauenden – aber harmlosen Eigenschaften wie Schlagschatten oder abgerundeten Ecken besteht. Das Risiko einer

[252] Wer surft schon gleichzeitig mit 2 Browsern)
[253] (40-60% je nach Browserstatistik und Zielgruppe) Vgl. Abb5 und [BROWSERSTATISTIKEN]
[254] Vgl. 4.3.2.
[255] Vgl. 4.3.1. und 4.3.2.
[256] Vgl. Tabelle 7

stark heruntergefahrenen Website für über 50% der Nutzer[257] zu liefern ist unakzeptabel für die meisten Live-Einsätze und Kunden.
CSS3 hat das Potential für die Weblayoutsprache der Zukunft, allerdings gilt es noch die ein oder andere aufgeführte Hürde zu überwinden. Es bleibt zu hoffen, dass sich alle angerissenen Probleme in den nächsten Monaten lösen und sich die zähe Unterstützung, die die Vorgängerversion erfahren hat, nicht wiederholt. Mit der gerade aktualisierten Platform Preview des IE9, die einen deutlichen Fortschritt gegenüber der ersten Version markiert, scheint sich positiver Trend abzuzeichnen.

Für Webentwickler ist es wie bei jeder Technologie, ratsam, CSS3-Merkmale einzusetzen, wenn es Sinn macht und es entsprechende „Fallback-Lösungen" für schwächere Engines gibt. Je mehr Webseiten in Zukunft auf CSS3 zurückgreifen und auch kommende Eigenschaften nachfragen, desto höher wird der Druck auf Browserhersteller und das W3C, diese anzubieten.

[257] Vgl. Abb. 5

# VII. Anhang

## A1 Die Reifungs-Stufen eines W3C Standards[1]

**Working Draft (WD)**
A Working Draft is a document that W3C has published for review by the community, including W3C Members, the public, and other technical organizations. Some, but not all, Working Drafts are meant to advance to Recommendation; see the document status section of a Working Draft for the group's expectations.
Working Draft (WD)
A Working Draft is a document that W3C has published for review by the community, including W3C Members, the public, and other technical organizations. Some, but not all, Working Drafts are meant to advance to Recommendation; see the document status section of a Working Draft for the group's expectations.

In addition to Working Drafts that are meant to advance to Recommendation, the other maturity levels of the Recommendation Track are:

**Candidate Recommendation (CR)**
A Candidate Recommendation is a document that W3C believes has been widely reviewed and satisfies the Working Group's technical requirements. W3C publishes a Candidate Recommendation to gather implementation experience.

**Proposed Recommendation (PR)**
A Proposed Recommendation is a mature technical report that, after wide review for technical soundness and implementability, W3C has sent to the W3C Advisory Committee for final endorsement.

**W3C Recommendation (REC)**
A W3C Recommendation is a specification or set of guidelines that, after extensive consensus-building, has received the endorsement of W3C Members and the Director. W3C recommends the wide deployment of its Recommendations. Note: W3C Recommendations are similar to the standards published by other organizations.

[1] [W3C 003]

## A2 Tabellen-Layout vs. CSS-Layout

Das Beispiel beschreibt eine typisches HTML-Dokument, so wie es vor der größerwerdenden Verbreitung von Stylesheets üblich war. Mithilfe des `table` Elements und den dazugehörigen Tabellen-reihen- und -spalten-Tags wurden mehrspaltige, komplexe Layouts realisiert. Das ging einher mit der visuellen Formatierung über HTML-Attribute, den präsentationellen Elementen font (für Veränderungen am Schriftbild) b und i sowie den Einsatz sogenannter „Spacer-GIFs", die einen 1px Rahmen kreieren.
Beide Methoden funktionieren zwar, allerdings stellen sich bei der Layout-Tabellen-Variante einige Nachteile ein.

| Kriterien | Tabellenlayout | HTML + internes CSS | HTML + externes CSS |
|---|---|---|---|
| Dateigröße | 12,7 kB | 5.5 kB | 3,2 kB (HTML)<br>2,5 kB (CSS) |
| Codezeilen | 237 | 170 | 48 (HTML) |
| Text-zu-Code-Verhältnis | 15.85 % | 38.02 % | 64.3 % |
| Übersichtlichkeit | teilweise 3fach verschachtelte Tabellen. Doppelte Stildefinitionen Präsentationelles Markup | Gegeben, kein unnötiger Codebalast, durch semantisches Markup und Klassennamen Erleichterung des Findens von Elementen | Gegeben, kein unnötiger Codebalast, durch semantisches Markup und Klassennamen Erleichterung des Findens von Elementen |
| Semantik | Markup gibt keine Auskunft über die Wichtung und Bedeutung des Inhaltes. | Deutliche Strukturierung des Inhalts durch entsprechende Tags und die umschließenden Container | Deutliche Strukturierung des Inhalts durch entsprechende Tags und die umschließenden Container |
| Wartbarkeit / Redesign | Sämtliche Layoutveränderungen müssen mühsam im Dokument mehrfach geändert werden. | Über die entsprechende CSS-Deklaration können Formatierungen leicht angepasst werden, allerdings bei mehreren Dokumenten sollte CSS ausgelagert werden | Über die entsprechende CSS-Deklaration können Formatierungen leicht angepasst werden |

## A3 Wertetypen in CSS

**1. Schlüsselwörter z.B. text-align:left; -**
Schlüsselwörte sind festgelegte Textwerte für diese Eigenschaft (aber auch für weitere Eigenschaften z.b. float:left;), die ein bestimmtes Aussehen hervorrufen.
Zu diesen Schlüsselwörtern gehören außerdem „inherit" (Wert vom Elternelement wird übernommen) sowie „initial" (Der Initialwert der Eigenschaft), die wiederum für alle CSS-Eigenschaften gültig sind.

**2. Reine Zahlen-Wertetypen z.B. z-index: 99 –**
Dies Wertetypen von CSS-Eigenschaften, die ohne Maßeinheiten angegeben werden. Abhängig von der Eigenschaft können entweder Integerwerte (ganzzahlig) oder reelle Zahlenwerte verwendet werden. Bei Nicht-negativen Zahlenwerten (z.B. orphans 3) darf der Wert nur größer 0 sein.

**3. Wertetypen mit Maßeinheit (relativ oder fest) z.B. margin-left: -10px)** – Werte mit Maßeinheiten dienen zur Dimensionierung und zur „Messung" von Abständen, Längen, Höhen oder Radien.

**Absolute Maßeinheiten** sind exakt angegebene Größen, z.B. cm

**Relative Maßeinheiten** beschreiben einen Wert im Verhältnis zu dem Wert einer anderen Eigenschaft oder dem Standardwert einer Eigenschaft des Ausgabemediums. (z.B. „em" orientiert sich an der Schriftgröße des Elements oder an der Schriftgröße des Elternelements)

**4. String-Wertetypen** z.B. content: "Absatz: " - beliebiger Text kann der Wert von bestimmten Eigenschaften sein, Wert muß in Anführungszeichen notiert sein.

**5. Funktionale Werteypen.** Die Wert-Zuweisung erfolgt über eine der festgelegten CSS Funktionen z.B. background-image: url(http://www.example.org/image.png) Werte, die aus einer CSS-Funktion und ihrem dazugehörigen Wert in Klammern bestehen und so eine URI3 oder Farbangaben definieren (z.B. url(...) oder rgb();)

**6. Sonderfälle** z.B. color: #FF0000" and „font-family: Helvetica, sans-serif – Kombination aus verschiedenen Wertetypen, oder Werte, die nicht eindeutig als Zahl oder String definiert werden können (z.B. Hex-Codes) Insbesondere bei den Farbwerten gibt es durch das CSS3-Colormodul wesentliche Neuerungen.

# A4 HTML5 - ein Überblick

HTML5 ist die Weiterentwicklung der Auszeichnungssprache HTML4 und XHTML1.1 (X)HTML ist für die semantische Strukturierung von Inhalten innerhalb der Dokumente des World Wide Web verantwortlich. HTML5 führt einige neue semantische Elemente ein, die typischen Inhalts-Bereichen – sog. Sektionen - von Webseiten entsprechen. Damit soll der übermäßige Gebrauch von `<div>`-Tags[1] eingedämmt werden und die Möglichkeiten der Interpretierbarkeit von Webseiten verbessert werden – für Mensch und Maschine.

**Außerdem implementiert HTML5 noch**

- direkte Einbindung von Medieninhalten wie Audio- oder Videodateien
- verbesserte Interaktivität (Webforms 2.0)
- clientseitige Datenspeicherung (Web Storage, z.b. Speichern von Formularinhalten)
- das scriptbare Zeichenfläche `canvas`
- neue Schnittstellen für die Erstellung von Webanwendungen unter Einbeziehung von JS (Drag & Drop Funktionialität, Geolocation)

**Neue strukturierende Elemente sind:**

`section` - kennzeichnet einen allgemeinen zusammengehörigen Bereich, z.B. Gruppierung von Artikeln des Hauptinhalts einer Seite. Sektionen müssen Titelbereiche (header) besitzen

`article` – Kennzeichnet einen inhaltlich unabhängigen Bereich einer Seite, wie z.B. einen Blogbeitrag, Forumsbeitrag, Kommentar. (Artikel)1

`header` – Kennzeichnet Titelbereiche, so z.B. den globalen Kopfbereich wie auch einzelne Titel von Sektionen oder Artikeln

`nav` – kennzeichnet Navigationsbereiche
aside – kennzeichnet Randinhalt, der nur bedingt thematisch zugehörig zum Rest der Seite ist.

`footer` – kennzeichnet Fußbereiche mit Zusatzinformationen

`figure` – kennzeichnet einen Bereich mit in sich geschlossenem Inhalt, wie ein Video oder ein Bild, das wiederum mit `figurecaption` mit einer Bildunterschrift gekennzeichnet werden kann.

---

[1] `<div>` ist ein allgemeinen Block-Element, das per se keine semantische Bedeutung besitzt, und erst durch die Vergabe von ids und Klassen dem umschlossenen Inhalt einen Sinn gibt.

Weitere neuen Elemente betreffen die Auszeichnung von Meta-Informationen, wie z.B.
`<time>` mit dem `datetime` Attribut zur machinenlesbaren Angabe von Daten und Zeiten

### Medien-Einbettung

Mit den Elementen `<video>` und `<audio>` können Video und Audiodateien direkt in ein Dokument eingebunden werden, ohne auf einen flashbasierten Video-Player zurückgreifen zu müssen. Wenn das `controls`-Attribut gesetzt ist, können Browser selbst alle nötigen Kontroll-Elemente zur Steuerung anzeigen, wie etwa Fortschrittsbalken, Start/Pause-Schaltfläche oder Lautstärkenregelung.

### Canvas

Mit dem `<canvas>`-Element kann eine Zeichenfläche im Dokument angelegt werden, auf der per Java-Script geometrische Formen und Pfade gezeichnet werden können Die Anwendungsbeispiele reichen von interaktiven Diagrammen bis hin zu Spielen.[2]

### Web Forms 2.0[3]

HTML5 erweitert die Möglichkeiten des `<input>`-Elements, indem es dem `type`-Attribut einige neue Werte ermöglicht. So kann beispielsweise mit `type="email"` angeben werdem, dass eine E-Mail-Adresse erwartet wird. Die Überprüfung, ob es sich bei einer Eingabe um eine gültige Adresse handeln kann, findet dann bereits direkt im Browser statt. Weitere neue `type`-Werte sind `date, time` und `datetime` (bzw. datetime-local) für echte Datums- und/oder Zeitangaben, `number` für numerische Werte, `color` für RGB-Farbwerte in Hex-Notation oder `url` für Internetadressen. Für komplexere Fälle kann man über das `pattern`-Attribut ein Suchmuster in Form eines regulären Ausdrucks angeben, auf welches der Eingabewert zutreffen muss. Sehr praktisch ist in dieser Hinsicht auch das neue Attribut `required`. Wenn es gesetzt ist, muss das betreffende Feld vom Benutzer ausgefüllt werden, unabhängig von möglichen weiteren Werteinschränkungen.
Weiterhin gibt es ein `autocomplete`-Attribut. Ist dieses auf on gesetzt, werden Eingaben gespeichert und bei zukünftigen Seitenaufrufen zur automatischen vervollständigung angeboten.
Weitere wichtige Attribute sind die `min-` und das `max-` Attribute, die einen einzugebenden Wert eingrenzen.
Interessant ist in dieser Hinsicht auch das `<datalist>`-Element, mit dem eine Datenliste angelegt werden, über die über ein `<input>`-Element zugegriffen werden kann.

---

[2] http://www.peterkroener.de/eine-kleine-canvas-einfuehrung/
[3] teilweise übernommen aus [DRWEB 001]

### Dokumenttypdeklaration & Syntax

Mit HTML5 wird eine einheitliche und leichter verständliche DocType-Deklaration eingeführt:

```
<!doctype html>
```

Außerdem wurde die XML-konforme Syntax des Ende 2009 eingestellten XHTML-Standards ebenfalls in Teilen übernommen[4] . D.h. Tags können auch in XHTML Kleinbuchstaben-Schreibweise definiert werden. Elemente ohne Endtag können nach wie vor mit einem „/“ beendet werden.

[4] [DEV W3C 002]

**Ein typisches HTML 5 Markup:**

```
<!doctype html>
<html>
<head>
      <title>Page title</title>
</head>
<body>
      <header>
            <h1>Page title</h1>
      </header>
      <nav>
            <ul>
                  <li><a href="#">Startseite</a></li>
                  <li><a href="#">Blog</a></li>
                  <li><a href="#">Kontakt</a></li>
            </ul>
      </nav>
      <section id="intro">
            <header><h2>Willkommen auf html5.de</h2></header>

      </section>
      <section id ="content">
            <h3>Neues aus dem Blog</h2>
            <article>
                  <h2>Blogbeitrag 1</h2>
                  <section>Einleitungstext1...</section>
            </article>
            <article>
                  <h2>Blogbeitrag 2</h2>
                  <section>Einleitungstext2...</section>
            </article>
      </section>
      <aside>
            <nav>
                  <ul>
                  <li><a href="#">Archiv 2010</a></li>
                  <li><a href="#">Archiv 2009</a></li>
                  <li><a href="#">Archiv 2008</a></li>
                  </ul>

            </nav>
      </aside>
      <footer>
            <!-- Footer -->
      </footer>

</body>
</html>
```

## A5 - Basis Stylesheet aus 4.3

```
/* Basic CSS Document */

body {
        font-family:Arial, Helvetica, sans-serif;
        color:#000;
        font-size:1.0em;
}
```

```
#mainNavigation {
float:left;
background-color:#848484;
width:824px; height:36px;
padding:0; margin:0;
}


        #mainNavigation li {
        float:left; display:block; list-style:none;
        }


        #mainNavigation li a {
        float:left; display:block;
        width:100px; height:27px;
        padding-top:9px; text-align:center; text-decoration:none; color:#000000;
        list-style:none; border-right:1px solid #878787;
        }

                #mainNavigation li a:hover, #mainNavigation li a.hover {
                background-color:#535353; color:#ffffff;
                }

                #mainNavigation li a.active {
                background-color:#373837; color:#ffffff;
                }


                #mainNavigation li#apple a {
                background:url(apple-logo.png) no-repeat center center;
                text-indent:-300px;
                }
```

```
#search {
        float:left;
        margin:0;
        padding:0;
        display:block;
        width:160px;
        padding-top:4px;
        padding-right:6px;
        background-color:#848484;
        height:32px;

}
```

```
#search input { width:133px; padding:3px 3px 3px 20px; margin-right:12px; line-
height:10px; display:block; color:#666; background:url(search.png) no-repeat
left center #ffffff; border:1px solid #f2f2f2;
```

## A6 Advanced Stylesheet aus 4.3.

```
@font-face {
       font-family: ‚Frutiger-Roman‘;
       src: url(‚Frutiger-Roman.eot‘);
       src: local(‚Frutiger Roman‘), url(‚Frutiger-Roman.ttf‘)
format(‚truetype‘);
       font-weight: normal;
       font-style: normal;
}

body {
       font-family:“Frutiger-Roman“, Arial, Helvetica, sans-serif;
       padding:50px 0 0 50px;

}
```

```
ul#mainNavigation {
       -webkit-border-radius:10px 0 0 10px;
       -moz-border-radius:10px 0 0 10px;
       background: -webkit-gradient(linear, left top, left bottom,
       from(#cacaca), to(#848484));
       background: -moz-linear-gradient(top, #cacaca, #848484);
       -moz-box-shadow:2px 2px 10px 5px rgba(0,0,0,0.1);
       -webkit-box-shadow:2px 2px 10px 5px rgba(0,0,0,0.1);
}
```

```
ul#mainNavigation li a  {

       text-shadow:1px 1px #ccc;
       -moz-box-shadow:-1px 0 #aaaaaa inset;
       -webkit-box-shadow:-1px 0 #aaaaaa inset;
       font-size:0.9em;
       font-weight:normal;
}
```

```
       #mainNavigation li a:hover, #mainNavigation li a.hover {
              background: -webkit-gradient(linear, left top,
              left bottom, from(#919191), to(#535353));
              background: -moz-linear-gradient(top, #919191, #535353);
              text-shadow:0px -1px #000000;
              }

       #mainNavigation li a.active {
              background: -webkit-gradient(linear, left top, left bot-
tom,                  from(#373837), to(#505251));
              background: -moz-linear-gradient(top, #373837, #505251);
              text-shadow:1px 1px #000000;
              -moz-box-shadow:3px 3px 20px #000000 inset;
```

```
            -webkit-box-shadow:3px 3px 20px #000000 inset;
            }
```

```
#search {
      background: -webkit-gradient(linear, left top, left bottom,
from(#cacaca), to(#848484));
      background: -moz-linear-gradient(top, #cacaca, #848484);
      -webkit-border-radius:0 10px 10px 0;
      -moz-border-radius:0 10px 10px 0;
}

#search input {
-moz-border-radius:20px;
-webkit-border-radius:20px;

}
```

# VIII. Literaturverzeichnis

## b.) Bücher, Studien & Fachzeitschriften


| | |
|---|---|
| DAVID | Mathew David, HTML5 Designing Rich Internet Applications, Focal Press ,2010 |
| GRANNELL | .CRAIG GRANNELL, NET Article „get the best out of CSS3“, 2008 |
| INTERNET INTERN | 02/10 - Perfektes Duo im Praxiseinsatz HTML5 & CSS3, 2010 |
| LABORENZ | CSS PRAXIS E-Book Edition, Galileo Computing; Auflage: 3 2005 |
| MEYER | Eric A. Meyer, The Definitive CSS GUide, 3rd Edition, O'Reilly & Associates, 2006 |
| POWELL | Thomas Powell, HTML & CSS: The Complete Reference, Fifth Edition, McGraw-Hill Osborne Media; 2010 |
| SCHMITT | CSS Cookbook, Christopher Schmitt, 2nd Edition, O'Reilly Media; 2006 |
| ANDREW & YANK | RACHEL ANDREW & KEVIN YANK, EVERYTHING YOU KNOW ABOUT CSS IS WRONG!, Sitepoint, 2008 |
| WIUM LIE & BOS 1999 | Håkon Wium Lie and Bert Bos, 1999, „The CSS Saga“ |
| WIUM LIE 1994 | PHD „Cascading Stylesheets“ 1.1.2 Presentational HTML (S. 29) |


## b.) Internetquellen


| | |
|---|---|
| 24 WAYS | http://24ways.org/2009/working-with-rgba-colour, Drew McLellan, 2009, „Working with RGBA Colour“ |
| ADOBE 001 | http://www.adobe.com/de/type/topics/info9.html, Adobe, letzte Prüfung 20.08.2010 |
| ALISTAPART 001 | http://www.alistapart.com/articles/fauxcolumns/, Dan Cederholm, „2004, Faux Columns“ |
| ALISTAPART 002 | http://www.alistapart.com/articles/understandingprogressiveenhancement/, Aaron Gustafson, 2008, „Understanding Progressive Enhancement“ |
| BROWSER-STATISTIKEN | http://www.browser-statistik.de/statistiken/, letzte Prüfung 15.08.2010 |
| BUDD | http://www.andybudd.com/archives/2007/05/css22/, Andy Budd, 2007, „CSS2.2“ |
| COYIER | http://css-tricks.com/yay-for-hsla/, Chris Coyier , 2010, Yay for HSLA |

| | |
|---|---|
| CSS3 INFO | http://www.css3.info/preview/, 2010, CSS3 PREVIEW |
| CSS3 INFO 002 | http://www.css3.info/preview/text-overflow/, css3.info ( Hrsg.) 2007 |
| CSS3 TEAM | http://xhtml.com/en/css/conversation-with-css-3-team/, 2007, Interview mit Bert Box über den Stand von CSS3 |
| DEV OPERA | http://dev.opera.com/articles/view/progressive-enhancement-with-css-3-a-be/, Peter Gasston, 2008 „PE & CSS3“ |
| DEV W3C | http://dev.w3.org/html5/html4-differences/#syntax, Anne van Kesteren, 2010, „HTML4 HTML5 Differencies“ |
| DOWEBSITES... | http://dowebsitesneedtobeexperiencedexactlythesameineverybrowser.com/ |
| DRWEB 001 | http://www.drweb.de/magazin/html5-ueberblick/, Clemens Kaposi, 10.11.09, „HTML5 - Überblick“ |
| FIND ME BY IP | http://www.findmebyip.com/#target-selector, letzte Überprüfung 20.08.2010, „Browser Support CSS3“ |
| GOLEM | http://www.golem.de/specials/drm/, 2008, Digital Rights Management |
| GOOGLE 2010 | http://www.google.com/support/webmasters/bin/answer.py?hl=de&answer=35769, Google (Hrsg.), 2010, „Richtlinien zur Gestaltung und zum Content“ |
| HAY | http://www.slideshare.net/stephenhay/the-future-state-of-layout, Stephen Hay, 05.11.2009, The Future State of Layout |
| HENRY | http://mwhenry.com/blog/2009/12/font-face-support-table/, Matt Henry, 2009, „Font-Face Support Table“ |
| HITSLINK 2010 | http://marketshare.hitslink.com/browser-market-share.aspx?qprid=0&qpcal=1&qptimeframe=M&qpsp=139, letzte Prüfung 15.08.2010, „Browser Market Share“ |
| HIXIE | http://ln.hixie.ch/?start=1181118077&count=1, Ian Hixie, 2007 |
| IRISH | http://paulirish.com/2009/bulletproof-font-face-implementation-syntax/, Paul Irish, 2009, „Bulletproof @font-face Implementaion Syntax |
| MEYER 001 | http://meyerweb.com/eric/tools/css/reset/, Eric A. Meyer, 2008, The Reset Stylesheet |
| MICROSOFT 001 | http://msdn.microsoft.com/en-us/library/cc351024%28VS.85%29.aspx, MSDN, o.J., „IE Compatibility“ |
| MICROSOFT 002 | http://ie.microsoft.com/testdrive/Default.htmll, 2010, Microsoft Corp., CSS3 Unterstützung IE9 |
| MICROSOFT 003 | http://samples.msdn.microsoft.com/ietestcenter/#css3, 2009, Microsoft, IE9 Test Center CSS3 |
| MODERNIZR | http://www.modernizr.com/, Modernizr is a small and simple JavaScript library that helps you take advantage of emerging web technologies |

| | |
|---|---|
| PAGLINAWAN | http://www.fontspace.com/andrew-paglinawan/quicksand, Andrew Paglinawan, 2008, Autor der Schriftart Quicksand |
| RUSSELL | http://infrequently.org/2007/09/css-3-a-giant-serving-of-fail/, Alex Russell, 2007, CSS3 is a giant serving of fail |
| SELFHTML 001 | http://de.selfhtml.org/css/layouts/browserweichen.htm, SELFHTML (Hrsg.), 2007, „Veraltete Browser ausschließen“ |
| SITEPOINT 2007 | http://reference.sitepoint.com/css/css, Sitepoint (hrsg), „What Is CSS?“ |
| SMASHING | http://www.smashingmagazine.com/2010/06/17/start-using-css3-today-techniques-and-tutorials/, Vitaly Friedman, 2010 |
| STARR | http://perishablepress.com/press/2010/01/11/css3-progressive-enhancement-smart-design/, Jeff Starr, 2010, „CSS3 + Progressive Enhancement = Smart Design“ |
| TYPOPHILE | http://typophile.com/node/63992, Richard Fink, 2009, Different Fonts Named The Same: Does This Happen Frequently? |
| W3C 001 | http://www.w3.org/Style/CSS/current-work, o.S., letzte Prüfung 15.08.2010 |
| W3C 002 | http://www.w3.org/People/howcome/p/cascade.html, Håkon W Lie, 1994, „The Original Proposal“ |
| W3C 003 | http://www.w3.org/2003/06/Process-20030618/tr.html, W3C (Hrsg.), 2003, „The Maturity Levels of a Recommandation“ |
| W3C 004 | http://www.w3.org/TR/2000/WD-css3-roadmap-20000414, Bert Box, 2000, „CSS3 Roadmap“ |
| W3C 005 | http://www.w3.org/Style/CSS/members.php3, W3C (Hrsg.), „CSS Members“ |
| W3C 006 | http://www.w3.org/TR/css3-namespace/, W3C, 2000, Terminologie und Syntax der CSS-Namensräume |
| W3C 007 | http://www.w3.org/TR/css3-ui/#box-model, W3C, 2004, „box-sizing-property“ |
| W3C 008 | http://www.w3.org/TR/css3-flexbox/#multiple, W3C, 2007 |
| W3C 009 | http://www.w3.org/TR/css3-layout/#declaring-templates-the-display-property, W3C, 2007 |
| W3C 010 | http://www.w3.org/TR/css3-grid/, W3C, 2007, Grid Positioning Module |
| W3C 011 | http://www.w3.org/TR/css3-background/, W3C, 2010, Backgrounds & Borders |
| W3C012 | http://www.w3.org/TR/css3-background/#the-box-shadow, W3C, 2010, „The Box shadow“ |
| W3C013 | http://dev.w3.org/csswg/css3-images/#linear-gradients |
| W3C014 | http://www.w3.org/TR/REC-CSS2, Bert Bos; Håkon Wium Lie; Chris Lilley; Ian Jacobs. Cascading Style Sheets, level 2. 1998. |

| | |
|---|---|
| W3C015 | http://www.w3.org/Submission/EOT/, Paul Nelson (Microsoft Corporation), 2008, „EOT Format“ |
| W3C 016 | http://www.w3.org/TR/css3-2d-transforms/, W3C & Apple, 2009, „2D Tranformations“ |
| W3C 017 | http://www.w3.org/TR/css3-3d-transforms/#transform-style-property, W3C, 2009 |
| W3C 018 | The @media rule http://www.w3.org/TR/CSS2/media.html |
| W3C 019 | http://www.w3.org/TR/css3-mediaqueries/#media0, 2008, Media Queries |
| W3C 020 | http://www.w3.org/TR/css3-page/, Opera, HP, W3C, 2006, The Page Model |
| W3C 021 | http://www.w3.org/TR/css3-page/#margin-box-def, Opera, HP, W3C, 2006, Margin Box Definitions |
| W3C 022 | http://www.w3.org/TR/css3-transitions/#transition-timing-function_tag, W3C, Apple Inc, 2009, „The Transition Timing Function“ |
| W3C 023 | http://www.w3.org/TR/NOTE-HTMLComponents, W3C, 1998, HTML Components (HTC) |
| W3C 024 | http://www.w3.org/Consortium/, W3C, 2010, „The Consortium“ |
| W3C NEWS | http://www.w3.org/News/2009#item119, 2009, W3C, „XHTML 2 Working Group Expected to Stop Work End of 2009“ |
| W3SCHOOLS 001 | http://www.w3schools.com/css/css_colornames.asp, W3schools, o.J., „Colornames“ |
| WEBKIT | http://webkit.org/, The Webkit Rendering Engine |
| WEBSTANDARDS | http://archive.webstandards.org/css/winie/, 1998, „IE‘s Top 10 Problems“ |
| WASP FEEDBACK | http://fantasai.inkedblade.net/style/discuss/wasp-feedback-2008, WaSP (Hrsg.), 2008, „CSS Feedback“ |
| WOFF | http://people.mozilla.com/~jkew/woff/woff-spec-latest.html, Mozilla Corp., 2010, „WOFF Specification“ |
| WEBFONTS | http://www.webfonts.info/wiki/index.php?title=@font-face_browser_support, Webfonts.info (hrsg), o.J. @font-face browser support |
| WEBKRAUTS | http://www.webkrauts.de/2006/09/01/webseiten-sind-keine-gemaelde/, Jens Grochtdreis, 2006, „Webseiten sind keine Gemälde“ |
| WEBMASTERPRO 001 | http://standards.webmasterpro.de/index-article-vertikal+Zentrieren.html, Paul Kröning, 2005, „Vertikales Zentrieren“ |
| WEBMASTERPRO 002 | http://www.webmasterpro.de/design/news/2009/02/02/verbreitung-von-schriftarten-im-web.html, Karin Wehle, 2009 „Verbreitung von Schriftarten im Web“ |

| | |
|---|---|
| WEBTOOLKIT 001 | http://www.webtoolkit.info/css-clearfix.html Die Clearfix-Methode zum Auslöschen störender Floats |
| WILCOX | http://mattwilcox.net/archive/entry/id/1031/, Matt Wilcox, 2007, „The fundamental Problems of CSS3“ |
| YAHOO DEV | http://developer.yahoo.com/yui/articles/gbs/, letzte Überprüfung August 2010, Yahoo Developer Network, „A-Grad-Browsers“ |
| WIKIPEDIA | http://en.wikipedia.org/wiki/Usage_share_of_web_browsers#GVU_WWW_User_Survey_.281994_to_1998.29, Market Share from 2000 to Present, 2009, „Usage share of web browsers“ |
| WIKIPEDIA 002 | |
| WSSEXPERT | http://www.wssexpert.de/Style/Examples/007/center.html#vertical, Bert Box, 2001, „Vertical Center“ |
| ZDNET 2006 | http://www.zdnet.com/blog/web2explorer/web-technology-penetration-report-css-continues-its-rise-and-frames-still-survive-in-16-of-websites/156, Richard MacManus, 2006, „Web Technology Penetration Report“ |

# IX. Erklärung zur selbstständigen Anfertigung der Arbeit

Ich erkläre, dass ich die vorliegende Arbeit selbstständig und nur unter Verwendung der angegebenen Literatur und Hilfsmittel angefertigt habe.
Chemnitz, 31.08.2010

_____

Sandro Tanneberger