­­

2018

Karl Steinam

Klara-Oppenheimer-Schule

28.11.2018

HTML und CSS



Inhalt

[Historie von HTML und css 4](#_Toc1419591)

[Zusammenhang HTML / Webserver 4](#_Toc1419592)

[5](#_Toc1419593)

[Eigenschaften 6](#_Toc1419594)

[Elemente 7](#_Toc1419595)

[7](#_Toc1419596)

[HTML-Tags 7](#_Toc1419597)

[HTML-Attribute 9](#_Toc1419598)

[Das HTML-Grundgerüst 10](#_Toc1419599)

[Die Grundlagen von CSS 15](#_Toc1419600)

[Aufbau von CSS-Regeln 17](#_Toc1419601)

[Einbinden von CSS 18](#_Toc1419602)

[CSS und Selektoren 20](#_Toc1419603)

[TypSelektoren 20](#_Toc1419604)

[Klassenselektor 20](#_Toc1419605)

[Übung: 21](#_Toc1419606)

[ID-Selektor 22](#_Toc1419607)

[Kontextselektor 23](#_Toc1419608)

[Lösungen 29](#_Toc1419609)

[Verdeutlichung am Strukturbaum 29](#_Toc1419610)

[Dein bester Freund DIV 32](#_Toc1419611)

[Das BOX-Modell von CSS 33](#_Toc1419612)

[Gesamthöhe und Gesamtbreite 34](#_Toc1419613)

[Aufgabe 34](#_Toc1419614)

[box-sizing: border-box 35](#_Toc1419615)

[Aufgabe 36](#_Toc1419616)

[DISPLAY-Eigenschaft 39](#_Toc1419617)

[inline: Inline-Box(en) 39](#_Toc1419618)

[block: Block-Box 39](#_Toc1419619)

[inline-block: Außen Inline-Box, innen Block-Box 39](#_Toc1419620)

[Aufgabe 41](#_Toc1419621)

[Float und Clear 42](#_Toc1419622)

[Einsatzgebiet 43](#_Toc1419623)

[Clearing the Float 44](#_Toc1419624)

[The Great Collapse 45](#_Toc1419625)

[#Video 45](#_Toc1419626)

[Aufgabe 46](#_Toc1419627)

[CSS/Eigenschaften/Positionierung/position 51](#_Toc1419628)

[Größenangaben 56](#_Toc1419629)

[Arten von Farben 57](#_Toc1419630)

[Schriftarten 58](#_Toc1419631)

[Glossar 59](#_Toc1419632)

[Die wichtigsten Definitionen in CSS 59](#_Toc1419633)

sf

# Historie von HTML und css

(Quell: Selfhtml, <https://wiki.selfhtml.org/wiki/HTML/Entstehung_und_Entwicklung>)

<https://wiki.selfhtml.org/wiki/Grundlagen/Einstieg/Entstehung_des_Internet#1993_-_Das_World_Wide_Web_.28WWW.29>

HTML (HyperText Markup Language) wurde 1990 vom Web-Gründer Tim Berners-Lee als Auszeichnungssprache (Markup Language) entwickelt, die auf SGML basierte.

Eine solche Auszeichnungssprache hat die Aufgabe, die logischen Bestandteile eines textorientierten Dokuments zu beschreiben. Als Auszeichnungssprache bietet HTML daher die Möglichkeit an, typische Elemente eines textorientierten Dokuments, wie Überschriften, Textabsätze, Listen, Tabellen oder Grafikreferenzen, als solche auszuzeichnen.

Das Auszeichnungsschema von HTML geht von einer hierarchischen Gliederung aus. HTML zeichnet Inhalte von Dokumenten aus. Dokumente haben globale Eigenschaften wie zum Beispiel Kopfdaten. Der eigentliche Inhalt besteht aus Elementen, zum Beispiel einer Überschrift 1. Ordnung, Textabsätzen, Tabellen und Grafiken. Einige dieser Elemente haben wiederum Unterelemente. So enthält ein Textabsatz zum Beispiel eine als betont markierte Textstelle, eine Aufzählungsliste besteht aus einzelnen Listenpunkten, und eine Tabelle gliedert sich in einzelne Tabellenzellen.

In der ursprünglichen Version von HTML gab es 18 Elemente, von denen TITLE und die Textstrukturierungselemente (H1-H6, P, A, ADDRESS, UL,OL und LI, sowie DL,DT und DD) immer noch verwendet werden, während Elemente wie PLAINTEXT, ISINDEX und LISTING heute nicht mehr gültig, sondern als obsolet angesehen werden. [1] [2]

Eine der wichtigsten Eigenschaften von HTML ist die Möglichkeit, Verweise zu definieren. Verweise („Hyperlinks“) können zu anderen Stellen im eigenen Projekt führen, aber auch zu beliebigen anderen Adressen im World Wide Web und sogar zu Internet-Adressen, die nicht Teil des Webs sind.[3]. Durch diese einfache Grundeigenschaft eröffnet HTML völlig neue Welten. Das Bewegen zwischen räumlich weit entfernten Rechnern wird bei modernen grafischen Web-Browsern auf einen Mausklick reduziert. In Ihren eigenen HTML-Dateien können Sie Verweise notieren und dadurch inhaltliche Verknüpfungen zwischen Ihren eigenen Inhalten und denen anderer Anbieter herstellen. Auf dieser Grundidee beruht letztlich das gesamte World Wide Web, und dieser Grundidee verdankt es seinen Namen

Wichtigste Erkenntnis:   
**HTML soll Dokumente strukturieren, nicht formatieren! Dafür setzte sich mit den Cascading Style Sheets (CSS) gegenüber angedachten Alternativen eine Erweiterung durch, die die Präsentation festlegen konnte**

* 1990 von Tim Berners-Lee als Auszeichnungssprache (Markup Language) entwickelt, die auf SGML basierte.
* Beschreibt die logischen Elemente eines Textdokumentes, z.B. Überschriften, Textabsätze, Listen, Tabellen, Bilder
* Dokument ist hierarchisch gegliedert
* Besteht aus Kopfdaten (Metadaten) und den eigentlichen Dokumenten

# Zusammenhang HTML / Webserver

Die Kommunikation zwischen Browser und serverseitigen Komponenten kann wie folgt aussehen.

## 

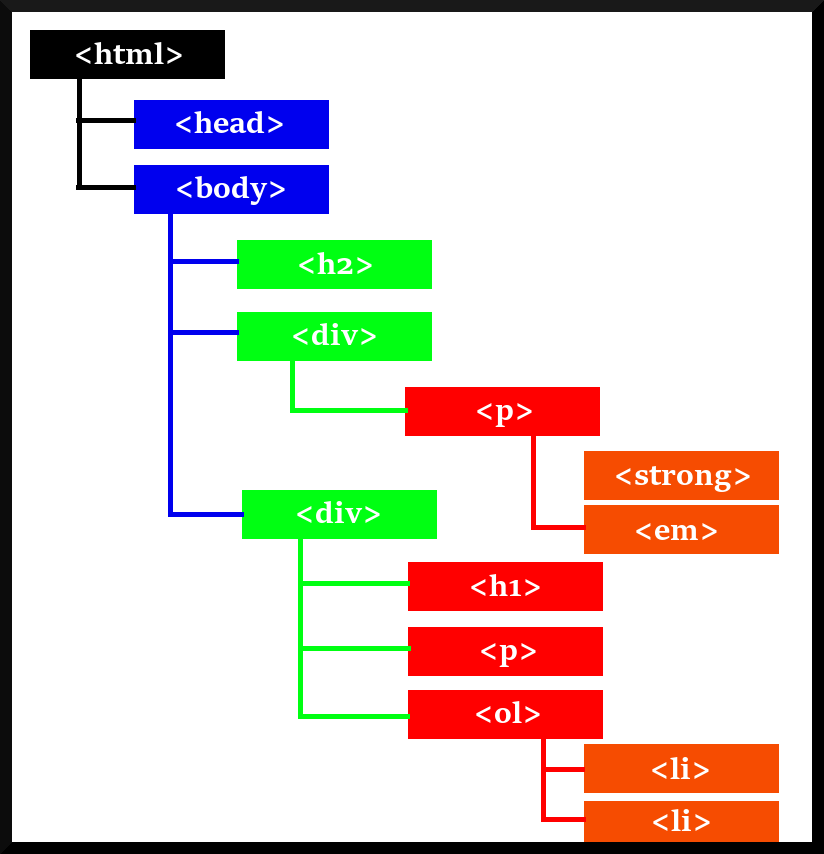
## Eigenschaften

Die Hypertext Markup Language (englisch für Hypertext-Auszeichnungssprache), abgekürzt HTML, ist eine textbasierte Auszeichnungssprache zur Strukturierung digitaler Dokumente wie Texte mit Hyperlinks, Bildern und anderen Inhalten. HTML-Dokumente sind die Grundlage des World Wide Web und werden von Webbrowsern dargestellt. Neben den vom Browser angezeigten Inhalten können HTML-Dateien zusätzliche Angaben in Form von Metainformationen enthalten, z. B. über die im Text verwendeten Sprachen, den Autor oder den zusammengefassten Inhalt des Textes.

HTML5 ist die fünfte Fassung der Hypertext Markup Language (engl. für Hypertext-Auszeichnungssprache), einer Computersprache zur Auszeichnung und Vernetzung von Texten und anderen Inhalten elektronischer Dokumente, vorwiegend im World Wide Web.

HTML dient als Auszeichnungssprache dazu, einen Text semantisch zu strukturieren, nicht aber zu formatieren. Die visuelle Darstellung ist nicht Teil der HTML-Spezifikationen und wird durch den Webbrowser und Gestaltungsvorlagen wie CSS bestimmt. Ausnahme sind die als veraltet (englisch deprecated) markierten präsentationsbezogenen Elemente.

Das World Wide Web Consortium (W3C) hat am 28. Oktober 2014 die fertige HTML5-Spezifikation („W3C Recommendation“) vorgelegt. HTML5 wird damit als Nachfolger von HTML4 die Kernsprache („core language“) des Webs. Sie ersetzt die Standards HTML 4.01, XHTML 1.0 und DOM HTML Level 2. Sie bietet neue Funktionen wie Video, Audio, lokalen Speicher und dynamische 2D- und 3D-Grafiken, die von HTML4 nicht direkt unterstützt wurden und sich ohne HTML5 nur mit zusätzlichen Plugins (z. B. Adobe Flash) umsetzen ließen. Zukunftsweisend sind weiterhin neue Elemente, die eine verbesserte semantische Struktur ermöglichen.



## Elemente

HTML-Elemente können hinsichtlich des Inhalts, den sie enthalten dürfen, kategorisiert werden.

Seit HTML5 existieren hierzu sieben grundsätzliche Kategorien:

* Metadaten: Beeinflussen das gesamte Dokument, beispielsweise das Einbinden externer CSS-Dateien
* Fluss-Elemente: Enthalten in der Regel Text oder Aussagen-Elemente
* Überschriften: Beschreiben einen Abschnitt
* Elemente zur Aufteilung: Teieln ein Dolkument in verschiedene logische Bereiche (article, aside, nav, section)
* Stil-Elemente: differenzieren Formulierung und Darstellungsebene ihres Inhalts, s.B. sup, sub, em, b
* Elemente zur Einbindung von Ressourcen und Interaktive Elemente, z.B. audio, canvas, iframe, img, svg und video

## 

### HTML-Tags

Das grundlegende Element von HTML ist das sog. **TAG**. Es definiert eine semantischr Aussage ohne die optische Presentation festzulegen. Alle Browser haben allerdings eine Statndardinterpretation für jeden Tag.

Die Struktur eines Tags ist wie folgt:



* In HTML gibt es fest definierte Tags, die alle eine bestimmte Aufgabe haben, z.B. <b>Inhalt</b> für fettgedruckten Text.
* Tags sind gekennzeichnet durch die <…>-Zeichen.
* Es gibt immer (zumindest fast) ein öffnendes und ein schließendes Tag, wobei das schließende Tag noch einen Schrägstrich vor dem Namen des Tags hat.
* Tags können geschachtelt seien, wichtig ist jedoch die korrekte Schachtelung
  + Tags müssen in der umgekehrten Reihenfolge geschlossen werden
  + Das Tag, welches als letztes geöffnet wurde, muss erstes wieder geschlossen werden
  + Fehler: <b><u>Inhalt</b></u>
* Es gibt einige Tags, die kein schließendes Tag besitzen (z.B. <br> oder <hr>)  
  Das sind sogenannte selbstschließende Tags (engl. self-closing tags)  
  Selbstschließende Tags sind Inhaltsleer (was nicht bedeutet, dass sie keine Informationen besitzen)
* <br> ist ein erzwungener Zeilenumbruch (markiert nur eine Stelle)
* <hr> ist eine horizontale Linie (Linie hat keinen Inhalt)

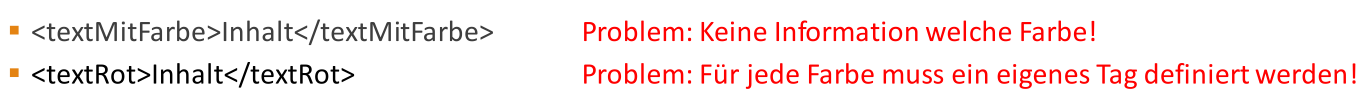
### 

### HTML-Attribute



Attribute können die Tags eines HTML-Elementes erweitern. Unterschiedliche Tags können verschiedene Attribute haben. Das Konzept von Attributen ist auch in anderen Textverarbeitungen / Programmen vorhanden

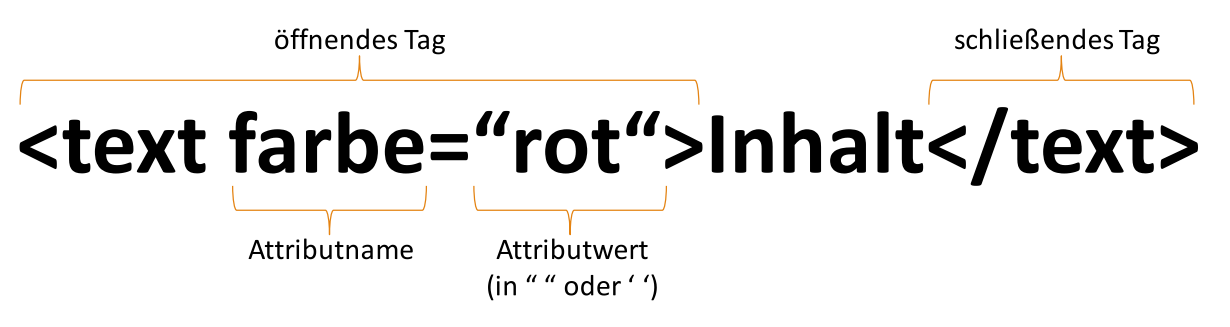
Im konkreten Fall einer Farbe wäre es schwierig, für jedes Element verschiedene Versionen des Tags vorzuhalten.



Die Lösung sind Attribute:



Attribute haben folgenden Aufbau:



## Das HTML-Grundgerüst

Eine minimale HTML-Datei hat folgenden Aufbau

<!DOCTYPE html>

<html lang="de">

<head>

<meta charset="utf-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Titel</title>

</head>

<body>

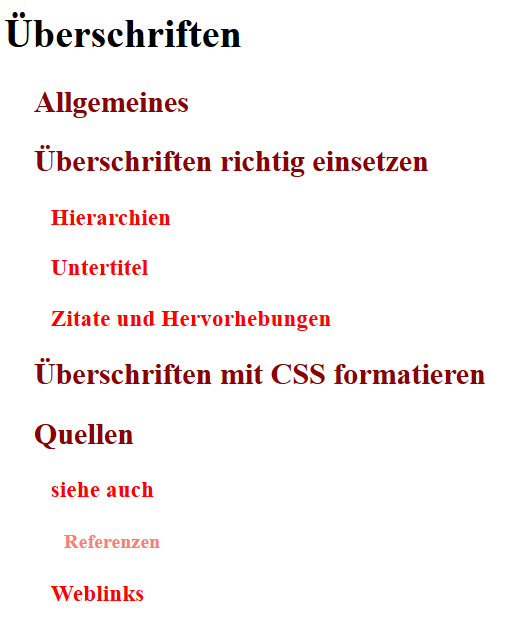
</body>

</html>

Übungsaufgabe:

A1.

Erzeugen Sie eine HTML-Datei, die folgende Struktur abbildet. Ignorieren Sie die unterschiedlichen Farben.



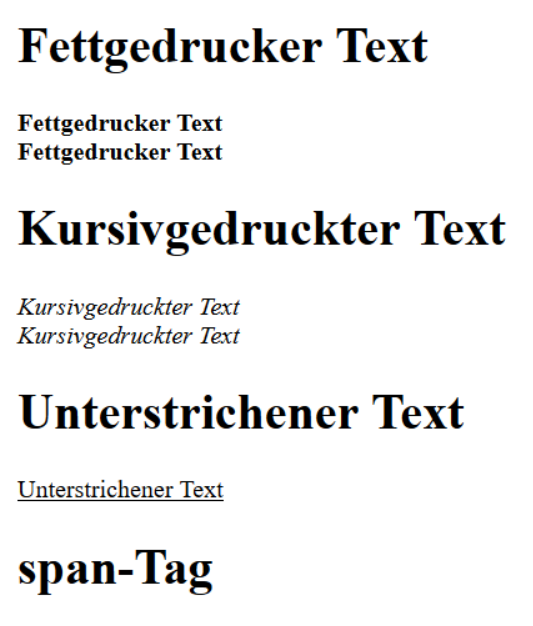
Erweiterung:

Welche Änderungen müssen Sie vornehmen, um die Überschriften in farbiger Schrift erscheinen zu lassen.

A2:

Erzeugen sie eine HTML-Datei mit folgendem Aussehen

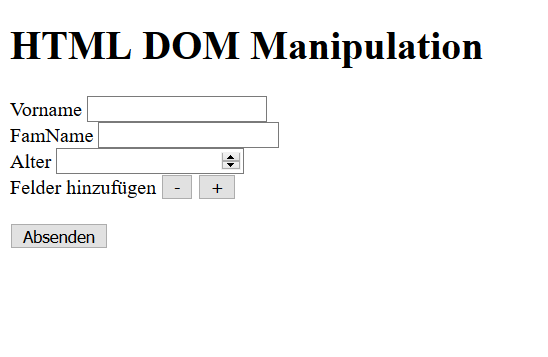
Hilfe gibt es unter https://wiki.selfhtml.org/wiki/HTML/Textauszeichnung



Aufgabe 3

Folgende HTML-Formular zeigt beispielhaft die Manipulation des DOM mit Javascript.

*http://bioub.github.io/d3.DOMVisualizer/*



Aufgabe 4:

Erstellen Sie ein DOM (Document Object Model) des folgenden HTML-Dokumentes

<!doctype html>

<html>

<head>

<meta charset="utf-8"/>

<title>Steckbrief von Charly</title>

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

</head>

<body>

<h1>Mein Hund</h1>

<p><img alt="Foto von Charly" src="charly.jpg"/></p>

<p>

Ich heiße <em>Charly</em> und wohne mit meinem Frauchen in der Nähe von Würzburg

</p>

<p>

Ich mag ganz gerne <em>Agility-Sport</em>.

Meine Lieblingsdisziplinen sind:

</p>

<ul>

<li><em>Sprung</em> durch einen Reifen</li>

<li><em>Lauf</em> über eine Wippe</li>

<li><em>Slalomlauf</em> zwischen Stangen</li>

</ul>

<p>

Ich gehe jedes Wochenende zum

<a href="https://www.hundesportverein-giebelstadt.de/">Hundesportverein Giebelstadt</a>

.

</p>

</body>

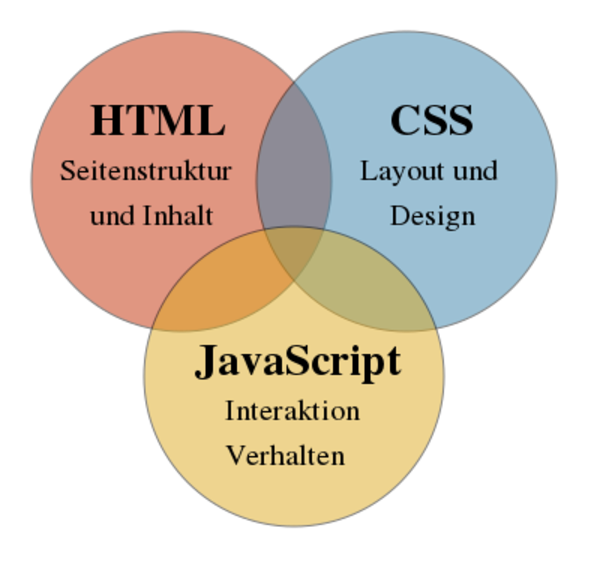
</html>

Lösung:

|  |  |
| --- | --- |
| https://www.inf-schule.de/content/1-information/1-informationsdarstellunginternet/2-strukturbeschreibunghtml/5-konzept_elemente/strukturbeschreibung_mit_elementen_3_v80.png | Baum der Elemente |
| Baum der Elemente | |

# Die Grundlagen von CSS

Im modernen Webdesign kommt den Webtechniken HTML, CSS und JavaScript jeweils eine bestimmte Rolle zu.



* + HTML legt fest, was auf der Seite stehen soll (struktureller Aufbau einer Webseite)
  + CSS legt fest, wie es dargestellt werden soll (Formatierung & Gestaltung)
* JavaScript legt fest, was passieren soll. (interaktive Elemente)

CSS (Cascading Style Sheets, zu deutsch „Mehrstufige Formatvorlagen“) ist eine Formatierungssprache für HTML-, SVG- und XML-Dokumente. Der „große Sinn von CSS“ besteht in der Trennung von Inhalt und Design. Das hört man oft, stellt sich nichts drunter vor und bastelt dann doch eine besondere Überschrift mit einem inline-style, einen neuen div-container mit einer ganz speziellen id oder Klasse, die später nirgends wieder auftaucht. Das funktioniert auch mehr oder weniger - man wird ohne viel Mühe eine fixe Seite zusammenstellen, die ihren Zweck erfüllt - und vergisst das ganze.

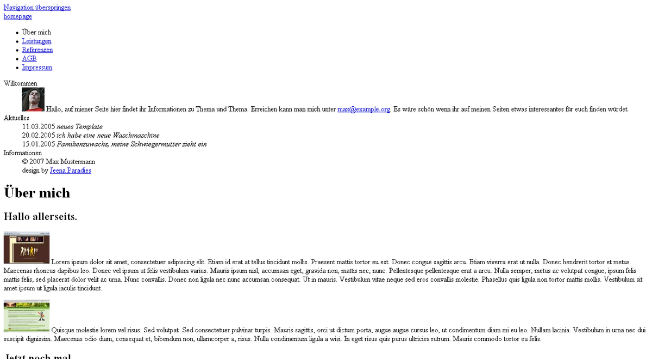
Es ist aber beinahe unmöglich, einen so über die Zeit gewucherten Internetauftritt umzugestalten - man müsste dutzende oder mehr Einzeldateien umschreiben und im Gewusel der Klassen und Elemente wird man sich schnell verlieren. Letztlich dauert die Änderung beinahe länger als die Neuerstellung. Und genau darin liegt die Stärke von CSS: uneingeschränkte Flexibilität, wenn z.B. das Layout nicht mehr zeitgemäß ist oder wenn neue Strukturen, vor allem bei dynamischen Seiten, Änderungen erfordern. Dabei wäre man ohne CSS buchstäblich „verloren im Quelltext“.

**Das erste Gebot von CSS lautet: „am Anfang ohne.“**

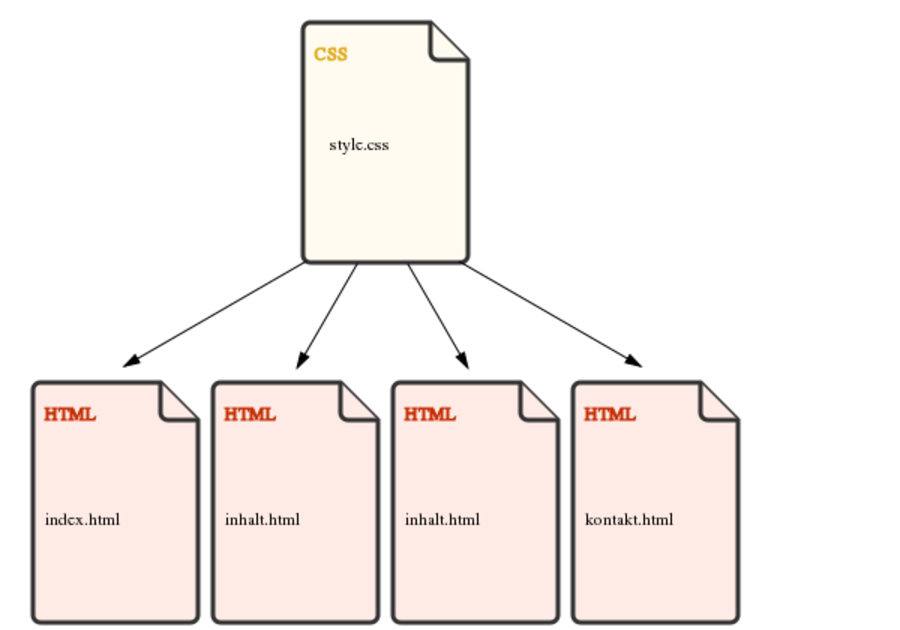
Soll heißen - man gestaltet die Seite erst komplett ohne CSS, nur die Seitenstrukturierung in Kapitel und Absätze, bei diesen dann die Textauszeichnung von Überschriften und Ähnlichem. Es stellt auch die „allgemeine Gültigkeit“ des Layouts sicher - ist die Grundlage an sich stimmig, kann man die Seite beliebig umgestalten, jeder Fehler schränkt dies wieder ein.

Eine solche Webseite sieht nicht gut aus, ist aber uneingeschränkt nutzbar. Die einzelnen Elemente werden vom Browser einfach untereinander auf dem Bildschirm angeordnet, und zwar in der Reihenfolge, in der sie in der HTML-Datei aufgeschrieben sind. Die Elemente selbst werden gemäß den Voreinstellungen des Browsers dargestellt; so ist z.B. festgelegt, dass der Hintergrund des Dokuments weiß oder grau darzustellen ist, dass Überschriften in fett und einer Größe von soundsoviel Punkten darzustellen sind.

An diesem Punkt setzen die Cascading Stylesheets ein. Es handelt sich dabei um eine unmittelbare Ergänzungssprache, die vorwiegend für HTML (aber auch für SVG) entwickelt wurde. Sie klinkt sich nahtlos in HTML ein und erlaubt das beliebige Formatieren einzelner HTML-Elemente. Mit Hilfe von Stylesheets können Sie beispielsweise festlegen, dass alle Überschriften 24 Punkt groß sind und mit einem Nachabstand von 16 Punkt und mit einer grünen doppelten Rahmenlinie oberhalb dargestellt werden. Schematisch würde dies etwa so aussehen:



CSS erlaubt es, Stile, Farben und Formen zu definieren, beispielsweise für alle Überschriften, oder für alle Textabsätze mit einem bestimmten Klassennamen, oder für kursiv ausgezeichneten Text, der innerhalb einer Tabellenzelle vorkommt.

Die zentralen Formate können sich auf eine HTML-Datei beziehen, aber auch in eine externe Style-Datei ausgelagert werden, die man in beliebig viele Seiten einbinden kann. So werden einheitliche Formatvorgaben möglich, und der HTML-Code wird von unnötigem Ballast befreit. Spätere Änderungen am Design können so leicht durchgeführt werden.

In der folgenden Abbildung dient eine einzige CSS-Datei beispielsweise vier HTML-Dateien als Formatvorlage:

## 

## Aufbau von CSS-Regeln

Eine CSS-Regel besteht aus

* der Bezeichnung für das Element, auf das die Regel zielt (**CSS-Selektor**),
* aus Eigenschaften (Properties), die dem Element zugewiesen werden.
* CSS-Eigenschaften stehen in geschweiften Klammern und sind durch Semikolons voneinander getrennt.
* Klassen- und ID-Namen sind case-sensitiv



selector { Eigenschaft: Wert;   
 Eigenschaft: Wert;   
 Eigenschaft: Wert   
}

h1 {   
 font-family: Helvetica;  
 font-size: 1.4em;  
 color: red  
}

## Einbinden von CSS

Styleinformationen können auf verschiedene Arten in ein HTML-Dokument eingebunden werden

**INLINE**

Mit dem Universalattribut **style** können Sie Eigenschaften einem Element direkt zuweisen. Dabei sind nur Deklarationen, also Eigenschaft-Wert-Zuweisungen möglich.

Beispiel:

<p style="text-align: center; color: green;">   
 Dieser Absatz wird über ein style-Attribut formatiert.  
</p>

**Bewertung:**

* Direktes Festlegen von Formaten pro Element
* Verlust vieler Vorteile
* Hoher Wartungsaufwand
* verringerte Flexibilität
* keine zentrale Bearbeitung

**INTERNAL**

Das HTML-Element **style** legt Formate zentral im Head des HTML-Dokuments fest.

<html lang="de">  
<head>  
 <style>  
 h1 {  
 background-color: green;  
 color: blue;  
 }  
  
 p {  
 text-align: center;  
 color: red;  
 }  
 </style>  
</head>  
<body>

<h1>Überschrift</h1>  
<p>Ein text halt</p>  
</body>  
</html>

**EXTERNAL**

Das CSS wird einem eigenen externen Stylesheet mit der Endung .css abgespeichert und mit dem HTML-Element link direkt im Head eingebunden.

<!--Einbinden eines externen Stylesheets -->

<!doctype html>  
<html lang="de">  
<head>  
 <link rel="stylesheet" href="stylesheet.css">

## CSS und Selektoren

Damit Formateigenschaften auf ein Element angewendet werden können, muss definiert werden, welche Elemente angesprochen werden. Dies geschieht über Selektoren. Als Selektoren bezeichnet man die Teile einer CSS-Regel, die vor dem Abschnitt aus geschweiften Klammern stehen. Voneinander unabhängige Selektoren, denen dieselben Eigenschaften zugewiesen werden, können mit Kommata getrennt werden. Es gibt verschiedene Arten von Selektoren.

### TypSelektoren

Der Element- bzw. Typselektor besteht aus dem Namen des Elements, das angesprochen werden soll. Mit diesem Selektor werden alle Elemente eines Typs angesprochen.

<!doctype html>

<html>

<head>

<meta charset="utf-8">

<title>CSS-Beispiel: Typselektor</title>

<style> p {border: 3px solid green;} </style>

</head>

<body>

<p>Dieses Beispiel demonstriert die Wirkung des Typselektors auf <abbr>HTML</abbr>-Dokumente.</p>

</body>

</html>

Typselektoren geben manche Eigenschaften(z.B. Schriftgestaltung) an ihre Kinder weiter und manche nicht (z.B. background-color, border). Dies kann erwünscht bzw. unerwünscht sein. Man kann dies innerhalb von CSS durch den Attributwert **inherit** verändern.

### Klassenselektor

* spricht Elemente an, die einer bestimmten Klasse zugehörend sind
* mehrere HTML-Elemente können die gleichen Klasse zugeordnet bekommen
* Klassenselektoren können mit anderen Selektoren verbunden werden
* Klassenselektor kann auch mit dem Typselektor verbunden sein
* **Ein Klassenselektor wird gebildet, wenn vor dem Klassennamen ein Punkt notiert wird**

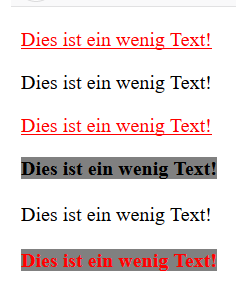
### Übung:

Erzeugen Sie aus dem folgenden HTML das nebenstehende Aussehen mit Hilfe von Klassenselektoren

<!DOCTYPE html>  
<html lang="de">

<head>  
<meta charset="utf-8">  
<title>Klassenselektoren</title>  
<link rel="stylesheet" type="text/css" href="klassenselektoren.css">  
</head>

<body>  
<p>Dies ist ein wenig Text!</p>  
<p>Dies ist ein wenig Text!</p>  
<p>Dies ist ein wenig Text!</p>  
<span>Dies ist ein wenig Text!</span><br><br>  
<span>Dies ist ein wenig Text!</span><br><br>  
<span>Dies ist ein wenig Text!</span>  
</body>  
</html>



### ID-Selektor

* Spricht ein Element an, dem eine ID zugeordnet wurde
* Gebildet durch Voranstellen des Gatterzeichens vor den ID-Namen
* ID-Selektoren können mit anderen Selektoren verbunden werden
  + Mit Elementselektoren: elementname#id
  + Mit Klassenselektoren: .klassenname#id bzw. #id.klassenname

Beispiel:

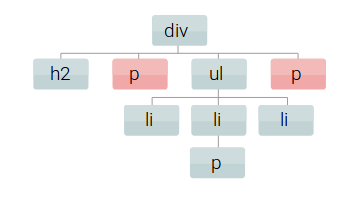
<!doctype html>  
<html>  
 <head>  
 <meta charset="utf-8">  
 <title>CSS-Beispiel: IDSelektor</title>  
 <style>  
   
 your code here.......  
   
 </style>  
 </head>  
 <body>  
 <div id="beispiel">  
 <h1>ID-Selektoren</h1>  
 <p>Dieses Beispiel demonstriert die Wirkung des ID-Selektors.</p>  
 </div>  
 </body>  
</html>

### Kontextselektor

Ein Nachfahren-Selektor oder Nachbar-Selektor wählt Elemente aus ihrer Position innerhalb von anderen Elementen aus – aus dem **Kontext** der HTML-Struktur. Sie sind also abhängig von der Struktur, in der sie auftreten: von ihren Nachbarn, Vorfahren und Nachfahren.

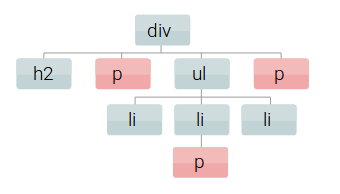
* Ein Nachfahren-Selektor (**Descendant Selector)** filtert Elemente unterhalb bestimmter Elemente.
* Ein Nachbar-Selektor (**Adjacent Selector)** filtert aufeindnerfolgende Elemente der gleichen Ebene,

#### Kindselektor

* ">" zwischen den beiden Selektoren
* Trifft nur auf Elemente zu, die direkt innerhalb des div-  
   Elementes liegen

div > p { color: black; background-color: red; }

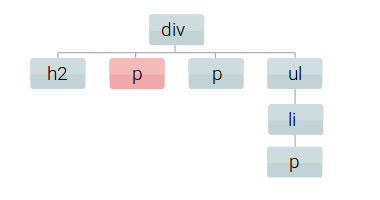
#### Nachfahrenselektor

* Leerzeichen " " zwischen den Selektoren

Alle p-Elemente, die innerhalb von div-Elementen liegen, werden durch die Regel angesprochen

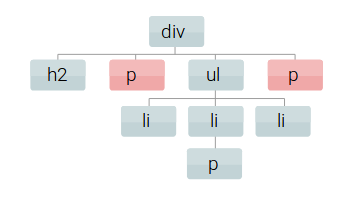
div p { color: black; background-color: red; }

#### Direkte Nachbar-Selektoren (Adjacent)

* Absteigender Selektor, angewendet auf ein p-Element,   
   das dem h2-Element direkt folgt.
* h2 und p haben dabei immer dieselben Eltern

h2 + p { color: black; background-color: red; }

#### Indirekte Nachbar-Selektoren

* Angewendet auf alle p-Elemente, die dem h2-  
   Element folgen.
* h2 und p haben dabei immer dieselben Eltern –   
   liegen also in derselben Ebene.

h2 ~ p { color: white; background-color: red; }

#### Aufgaben zu Selektoren

* 1. Folgende HTML-Datei ist vorhanden:

<!DOCTYPE html>

<html lang="de">

<head>

<meta charset="utf-8">

<title>Typselektoren</title>

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

</head>

<body>

<h1>Leggings kickstarter</h1>

<p>

Coloring book vaporware heirloom cloud bread man braid, ramps organic umami distillery.

<span>

Kale chips retro literally locavore migas raw denim.

<b>Live-edge austin taiyaki four dollar toast, cray shoreditch.</b>

DIY meh wayfarers retro banh mi selfies food truck.

</span>

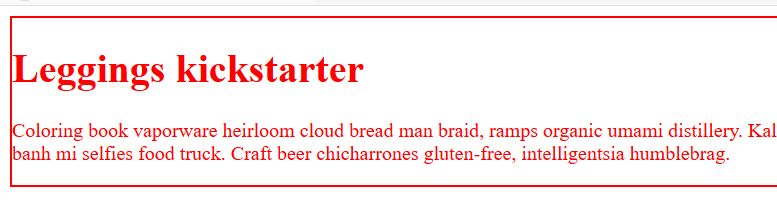
Craft beer chicharrones gluten-free, intelligentsia humblebrag.

</p>

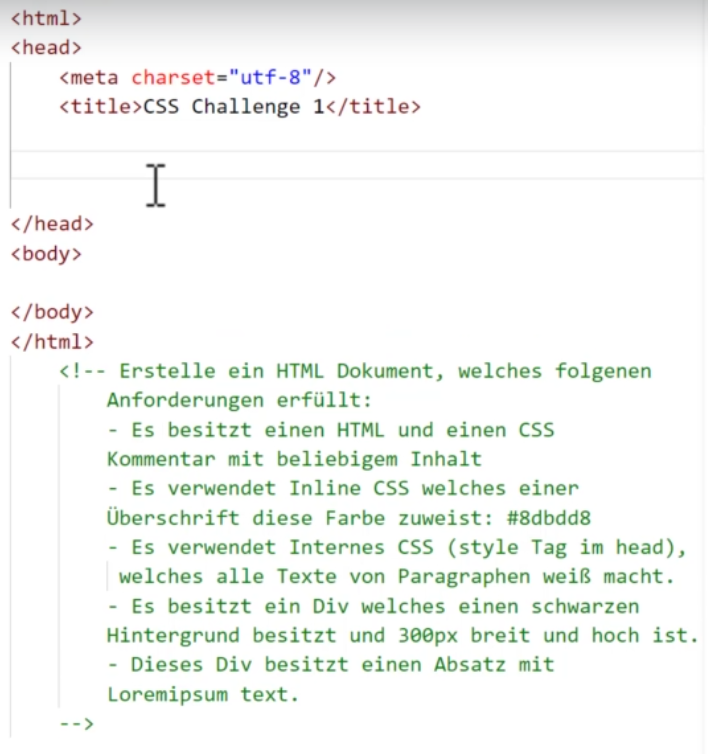
</body>

</html>

Erstellen Sie untenstehende HTML-Datei mit Hilfe von Typselektoren  
Arbeiten Sie die Aufzählungspunkte hintereinander ab.

* Die Farbe des body-Tags soll einheitlich rot sein  
  
* Die Farbe des span-tags soll grau sein
* Die Farbe des h1-Tags soll ausnahmsweise blau sein
* Der body-Tag soll einen roten Rahmen erhalten  
   Warum erhalten die Kindelemente des body-Tags keinen Rahmen ?
* Geben Sie dem h1-Tag ebenfalls einen Rahmen, indem Sie von body den Rahmen erben.

Aufgabe 2:



Aufgabe 3

Erstelle eine HTML Datei, welche folgenden Bedingungen erfüllt:

* Grundstruktur einer HTML Datei
* Verwendet einen Tag Selektor (z.B. p Tag) der die Farbe auf Dunkelgrau setzt
* Verwendet einen ID Selektor "viereck" welcher ein div 300px breit und 300px hoch macht und ihm einen #b2d3e6 Hintergrund gibt.
* Verwende einen Class Selektor "monogross" der mindestens von zwei Elementen genutzt wird und die Schriftart zu monospace ändert und die Schriftgröße auf 150% setzt.
* Besitzt einen Link, der den Nutzer auf Google weiterleitet und beim Hovern die Farbe Rot annimmt.

Aufgabe 4:

Formatieren Sie die untenstehende HTML-Datei mit Hilfe von CSS.

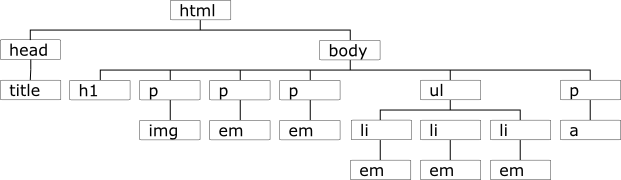
body

h1

<https://www.inf-schule.de/information/informationsdarstellunginternet/formatierungcss/exkurs_css/selektoren>

Aufgabe 5:

Gegeben ist folgende DOM-Struktur



Markieren Sie sich die Stellen, die durch den jeweiligen CSS-Selektor ausgewählt werden.

ul em  
{  
 color: red;  
}

Em  
{  
 color: blue;  
 font-weight: bold;  
}

h1, em  
{  
 font-weight: bold;  
 font-style: normal;  
}

Body  
{‘  
 background-color: orange;   
 color: black;  
 font-size: small;  
 font-family: Georgia, "Trebuchet MS", Verdana, sans-serif;  
}

Wie wird sich folgende CSS-Anweisung auf den DOM auswirken

body

{

background-color: orange;

color: black;

font-size: small;

font-family: Georgia, "Trebuchet MS", Verdana, sans-serif;

}

em

{

color: blue;

}

Was müssen Sie im HTML-Dokument ändern, damit folgende Regel wirken kann.

p.wichtig

{

background-color: white;

}

Aufgabe 6:

Lösen Sie das CSS-Rätsel unter

https://steinam.rigel.uberspace.de/css-diner/

# Lösungen

Aufgabe 5:

### Verdeutlichung am Strukturbaum

em

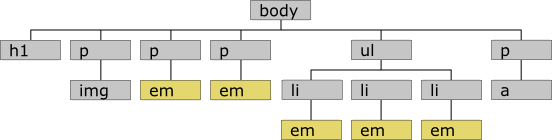
{

color: blue;

font-weight: bold;

}

Dieser Selektor wählt im Strukturbaum die farbig markierten Elemente aus. Für diese Elemente legt die Regel dann die Eigenschaften fest.



Das zweite Beispiel betrifft den Selektor h1, em in der folgenden Regel, der zwei Elemente betrifft.

h1, em

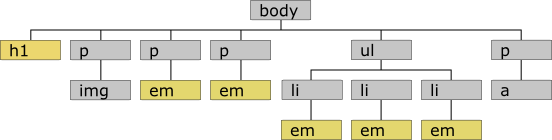
{

font-weight: bold;

font-style: normal;

}

Dieser mit einem Komma zusammengesetzte Selektor wählt folgende Elemente aus:



Im Style-Sheet stil.css kommt die folgende Regel mit einem zusamengesetzten Selektor ul em vor (Achtung: hier ohne ein Komma).

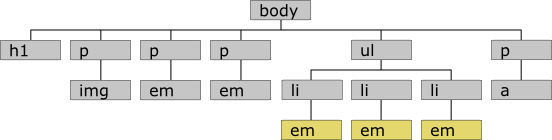
ul em

{

color: red;

}

Hiermit werden alle <em></em>-Elemente ausgewählt, die Nachkommen eines <ul></ul>-Elementes sind.



Eigenschaften vererben sich im Strukturbaum auf die Nachkommen - das sind die Elemente unterhalb des Ausgangselements. Die folgende Regel für das <body></body>-Element legt implizit fest, dass auch alle Nachfahren über dieselben Eigenschaften verfügen.

body

{

background-color: orange;

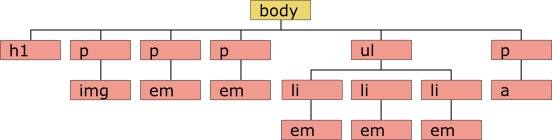
color: black;

font-size: small;

font-family: Georgia, "Trebuchet MS", Verdana, sans-serif;

}

Im Strukturbaum sind das ausgewählte Element (gelb) und alle seine Nachkommen (rot) jetzt farbig markiert.



Im vorliegenden Fall bewirkt diese Regel, dass u. a. die Schriftfarbe für alle Elemente schwarz ist.

Kommt jetzt aber unterhalb einer allgemeinen Regel eine speziellere, so überschreibt diese die allgemeine, wie das folgende Beispiel zeigt.

body

{

background-color: orange;

color: black;

font-size: small;

font-family: Georgia, "Trebuchet MS", Verdana, sans-serif;

}

em

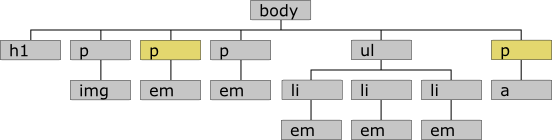
{

color: blue;

}

Durch diese beiden Regeln wird festgelegt, dass zunächst einmal alle Elemente die Schriftfarbe schwarz haben. Für die <em></em>-Elemente wird durch die zweite Regel eine Ausnahme gesetzt. Diese Elemente sollen die Schriftfarbe blau erhalten.

Es gibt auch die Möglichkeit, nur ganz bestimmte Elemente auszuwählen. Im folgenden Strukturbaum sind nur einige der <p></p>-Elemente markiert. Nur diese sollen z.B. die Hintergrundfarbe weiß erhalten.



Damit das möglich wird, müssen zunächst die betreffenden <p></p>-Elemente im HTML-Quelltext markiert werden. Hierzu dient das Attribut class, dem man einen selbst gewählten Wert (hier: "wichtig") zuweisen kann.

<!doctype html>

<html>

<html lang="de">

<meta charset="utf-8">

<link href='styles/style.css' rel='stylesheet' type='text/css'>

<title>Steckbrief von Bailey</title>

</head>

<body>

<h1>Steckbrief</h1>

<p><img src="img/bailey.jpg" alt="Foto von Bailey"></p>

<p class="wichtig">

Ich heiße <em>Bailey</em>. Ich lebe in der Nähe

von Kaiserslautern.

</p>

<p>

Ich interessiere mich für <em>Agility-Sport</em>.

Meine Lieblingsdisziplinen sind:

</p>

<ul>

<li><em>Sprung</em> durch einen Reifen</li>

<li><em>Lauf</em> über eine Wippe</li>

<li><em>Slalomlauf</em> zwischen Stangen</li>

</ul>

<p class="wichtig">

Ich gehe jedes Wochenende zur

<a href="http://www.welpenschule-kl.de/index.html">Welpenschule</a>

.

</p>

</body>

</html>

Jetzt kann für die Klasse der wichtigen Absätze eine neue Regel definiert werden.

p.wichtig

{

background-color: white;

}

CSS

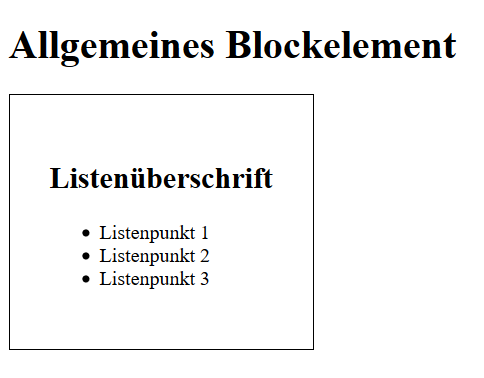
Alles recht kompliziert, oder? Es gibt noch weitere Möglichkeiten, bestimmte Elementgruppen auszuwählen. Das kannst du dann in einem Nachschlagewerk wie [SELFHTML](https://www.inf-schule.de/information/informationsdarstellunginternet/hinweise) nachlesen.

# Dein bester Freund DIV

Das **div**-Element ist dazu gedacht, mehrere Elemente wie Text, Grafiken, Tabellen usw., in einen gemeinsamen Bereich einzuschließen. Dieses allgemeine Element bewirkt nichts weiter als dass es in einer neuen Zeile des Fließtextes beginnt. Ansonsten hat es keine Eigenschaften. Es ist dazu gedacht, Bereiche zu erzeugen, die mit Hilfe von CSS formatiert werden können. div bedeutet *division*, etwa Abteilung oder Bereich.

Häufig werden div-Elemente als **wrapper** (englisch für Verpackung) oder Container-Element verwendet. Dabei wird die [semantische](https://wiki.selfhtml.org/wiki/Semantisch) Struktur um ein weiteres, semantisch leeres Element ergänzt, dessen einziger Zweck beispielsweise die Zentrierung oder die Formatierung mit einer Randlinie ist.

Beispiel: Erzeugen Sie folgende Ausgabe

Lösung:

<!doctype html>   
<html>   
<head> <meta charset="utf-8">   
<title>Allgemeines Block-Element</title>   
<style> div { border: 1px solid; float: left; } </style>   
</head>   
<body>   
 <h1>Allgemeines Blockelement</h1>   
 **<div>** <h2>Listenüberschrift</h2>  
 <ul>   
 <li>Listenpunkt 1</li>   
 <li>Listenpunkt 2</li>   
 <li>Listenpunkt 3</li>   
 </ul>   
 **</div>**   
</body>   
</html>

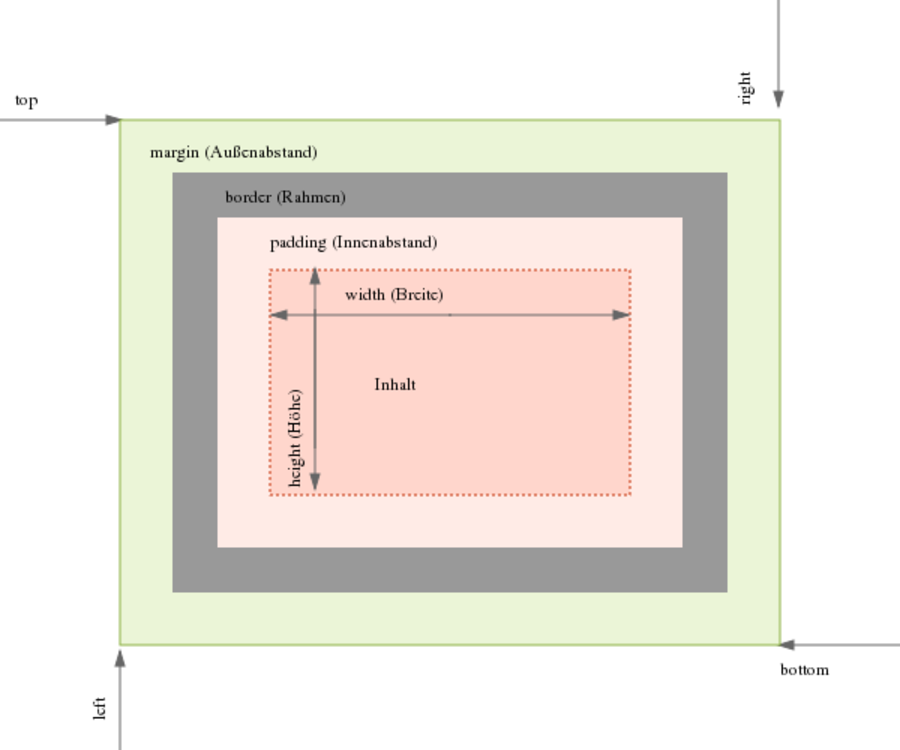
# Das BOX-Modell von CSS

<https://codepen.io/carolineartz/full/ogVXZj>

https://de.wikibooks.org/wiki/Websiteentwicklung:\_CSS:\_Box-Modell

Das Box-Model gilt für alle HTML-Tags, die **Block-Elemente** sind, sowie für alle Inline-Elemente, auf die die CSS-Anweisung display: block; angewendet wurde. Natürlich trifft das nicht auf Block-Elemente mit der CSS-Eigenschaft display: inline; zu.

Im visuellen Anzeigemodell erfolgt die Darstellung von Elementen durch das Zeichnen von Rechtecken. Die Bestandteile dieser Rechtecke beschreibt das „**Box-Modell**“. Eine Box kann bestehen aus:

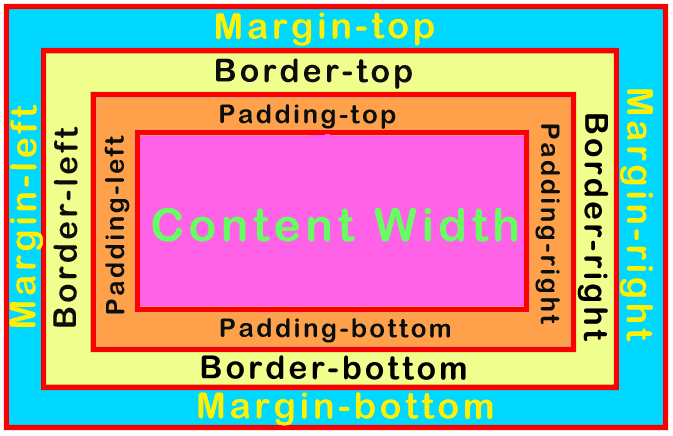
- Dem Inhaltsbereich, also der Fläche, die durch Texte und Bilder oder Eigenschaften wie [width](https://wiki.selfhtml.org/wiki/CSS/Eigenschaften/Gr%C3%B6%C3%9Fenangaben/width) und [height](https://wiki.selfhtml.org/wiki/CSS/Eigenschaften/Gr%C3%B6%C3%9Fenangaben/height) vorgegeben wird,  
- einem Innenabstand ([padding](https://wiki.selfhtml.org/wiki/CSS/Eigenschaften/Abstand/padding" \o "CSS/Eigenschaften/Abstand/padding)),  
- einem Rahmen ([border](https://wiki.selfhtml.org/wiki/CSS/Eigenschaften/%C3%A4u%C3%9Fere_Gestaltung/Rahmen/border" \o "CSS/Eigenschaften/äußere Gestaltung/Rahmen/border)) und  
- einem Außenabstand ([margin](https://wiki.selfhtml.org/wiki/CSS/Eigenschaften/Abstand/margin" \o "CSS/Eigenschaften/Abstand/margin)).

Bei [Block-Elementen](https://wiki.selfhtml.org/wiki/Block-Element) können Höhe und Breite beliebig festgelegt werden, bei Inlineelementen werden die Maße durch den Inhalt vorgegeben.   
Innen- und Außenabstände sowie Rahmen können für jede der vier Seiten einer Box einzeln festgelegt werden.

Wird eine Box [positioniert](https://wiki.selfhtml.org/wiki/CSS/Eigenschaften/Positionierung/position), beginnt die linke Außenkante bei left, die obere Außenkante bei top, die rechte Außenkante bei right und die untere Außenkante bei bottom.

Als Referenz-Box wird derjenige Teil einer Box bezeichnet, für den bestimmte Eigenschaften gelten sollen.

Die einzelnen Rechtecke werden von innen nach außen bezeichnet als

* content-box: (Inhaltsbox) Bereich, der durch den Inhalt oder die Eigenschaften width und height festgelegt wurde.
* padding-box: (Polsterungsbox) Bereich, der content-box und padding (Innenabstand) umfasst. Besitzt eine Seite keinen Innenabstand, so ist die Polsterungskante mit der Innenkante identisch.
* border-box: (Rahmenbox) Box, die content-box, einen möglichen Innenabstand und die durch border festgelegten Rahmen umfasst. Besitzt eine Box keinen Rahmen, so ist die Rahmenbox mit der Polsterungsbox identisch.
* margin-box: Box mitsamt durch margin festgelegten Außenabständen. Sind für eine Box keine Außenabstände definiert, so ist die Außenkante mit der Rahmenkante identisch.

## Gesamthöhe und Gesamtbreite

Für Anfänger irritierend ist, dass mehrere Boxen nicht nebeneinander dargestellt werden, obwohl deren Gesamtbreite genau der Breite des Eltern-Elements entspricht. Dies liegt daran, dass alle Bestandteile einer Box addiert werden.

Die Gesamtbreite errechnet sich aus:

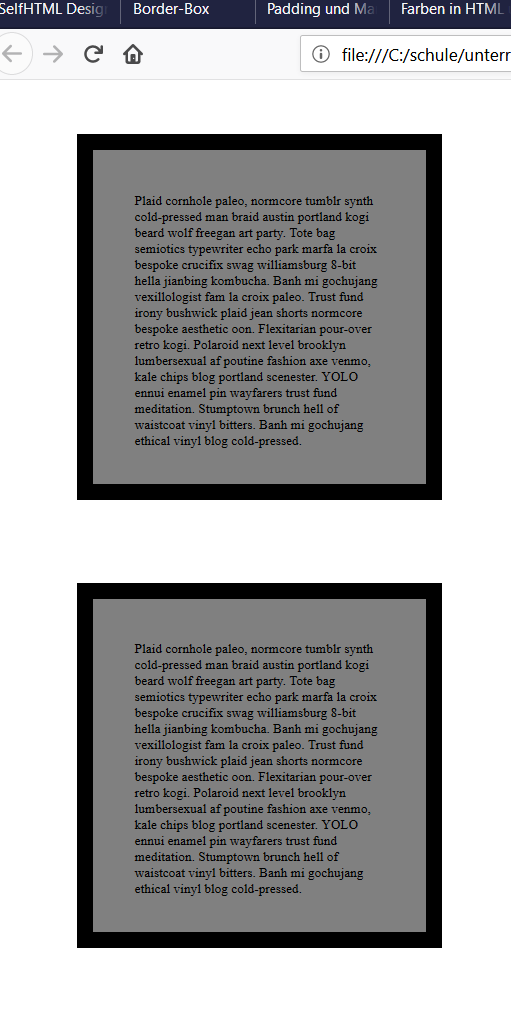
margin-left + border-left + padding-left + width + padding-right + border-right + margin-right

Die Gesamthöhe einer Box wird ebenso errechnet aus:

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

## Aufgabe

Erzeugen Sie folgendes Aussehen einer HTML-Seite. Benutzen Sie dazu mehrere DIVS



## box-sizing: border-box

Die ursprüngliche Definition, die Angaben von width und height als Abmessungen lediglich des Inhaltes festzulegen ist wenig intuitiv. Eigentlich erwartet man, dass eine Breiten- bzw. Höhenangabe den gesamten Platzbedarf des Elementes widerspiegelt.

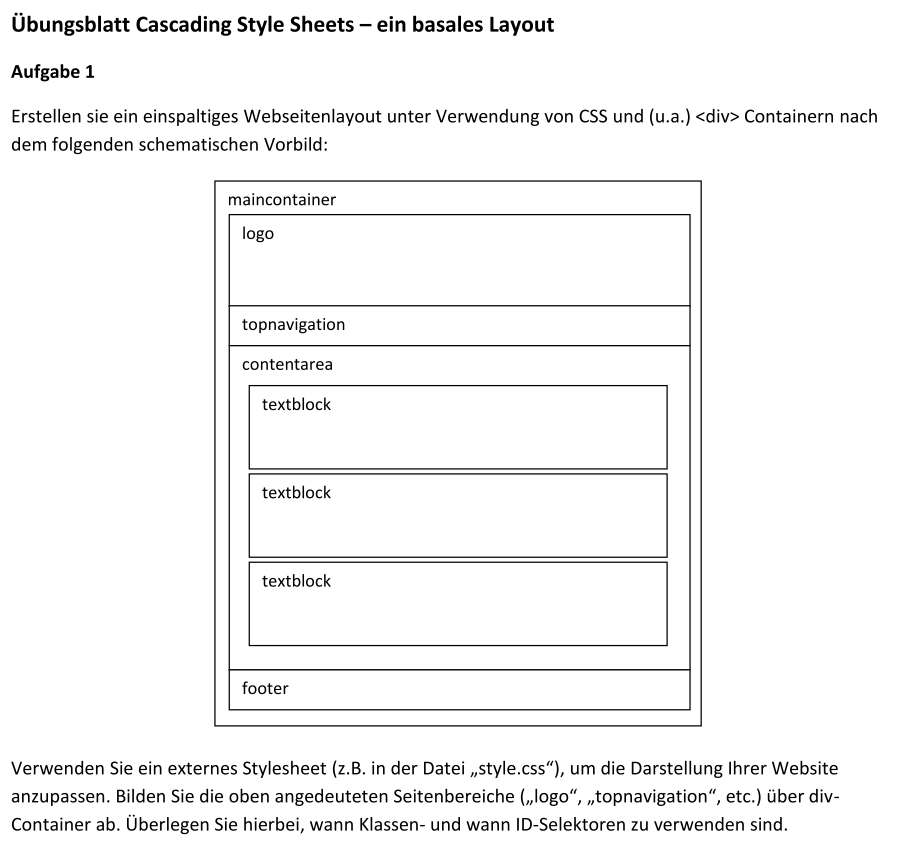
Zudem interpretierten alte Internetexplorer das Box-Modell falsch, sodass Webentwickler auf diese besondere Rücksicht nehmen mussten. Ebenso war es schwierig bis unmöglich, ein responsives Layout zu erstellen, welches etwa prozentuale Breitenangaben mit pixelgenauen Angaben für die Rahmenbreite kombinierte.

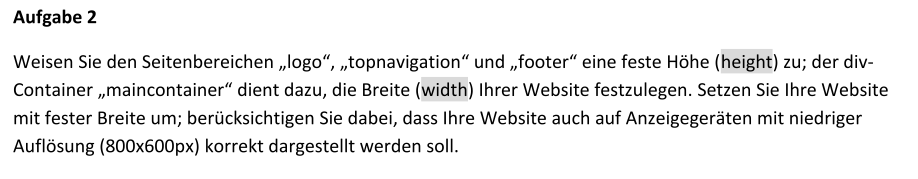
Mit der Eigenschaft box-sizing können Sie bestimmen, worauf sich Angaben zu den Abmessungen eines Elementes beziehen.

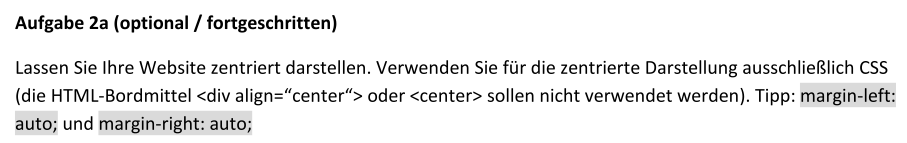
Erlaubt sind dabei eine der folgenden Angaben.

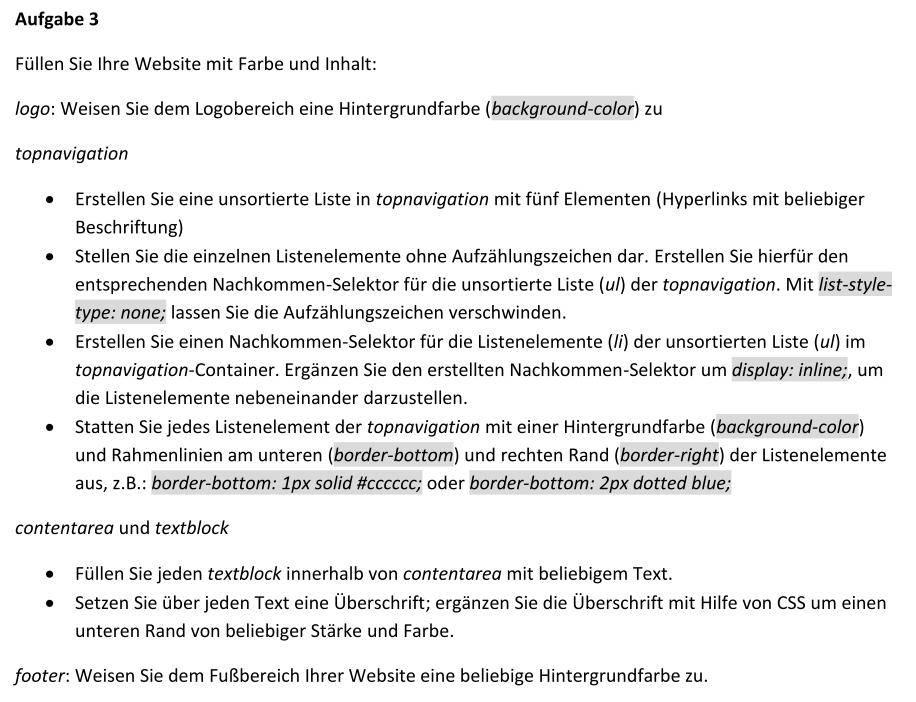
* content-box, Standardwert, Angabe gilt nur für den Inhalt
* border-box, Angabe gilt für Inhalt, Innenabstand und Rahmen, Standard für IE im Quirksmodus
* inherit, box-sizing des Elternelements

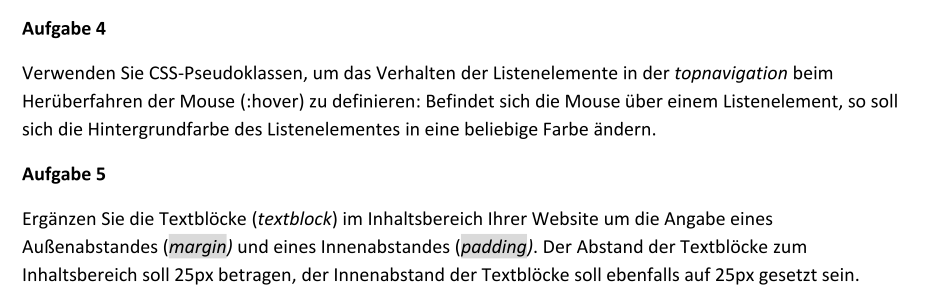
## Aufgabe

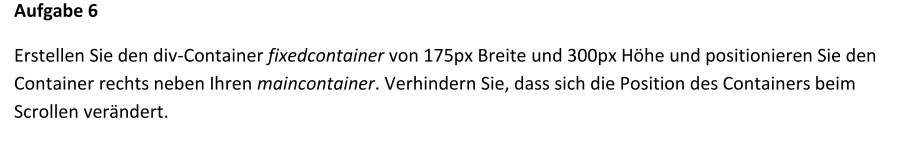












Inline- vs- Block-Elemente  
  
Die Breite von **Inline-Boxen** wird allein durch den Inhalt bestimmt. Die width-Eigenschaft ist wirkungslos. Die Teile einer aufgeteilten Inline-Box nehmen nur die Breite ein, die der Inhalt auch tatsächlich benötigt.

**Block-Boxen** nehmen die gesamte Breite des Elternelementes ein. Sie sind so hoch wie ihr Inhalt. Dadurch entsteht ein zusammenhängendes Rechteck, das aussieht wie eine Box und dieser Darstellungsart ihren Namen gibt.

Mehrere Block-Boxen werden untereinander angeordnet (normaler Elementfluss).

Beispiele für Blockelemente sind Absätze(p). Die Absätze dieses Artikels stehen alle untereinander und nehmen jeder für sich die gesamte Breite des Inhaltsbereiches ein.

Die Höhe eines Blockelementes wird von der Höhe des Inhaltes bestimmt: hat ein Absatz viele Worte, wird das Element groß (hoch), enthält es gar nichts, ist die Höhe des Inhaltes gleich null.

Beispiel

<!DOCTYPE html>

<html lang="de">

<head>

<meta charset="utf-8">

<title>Inline- und Block-Elemente</title>

<link rel="stylesheet" type="text/css" href="[inlineBlock.css](view-source:file:///C:/schule/unterricht/web/10_html_css/uebungen/Grundlagen/049_inlineBlock/inlineBlock.css)">

</head>

<body>

<span>Die Span ist ein Inline-Element</span><u> also fängt das nächste Inline-Element nicht in der nächsten Zeile an</u>

<strong> sondern die aktuelle Zeile wird einfach fortgesetzt.</strong>

<p>Dies ist ein Paragraph mit ganz viel tollem Text zu rumprobieren!</p>

<div></div>

<h1>Dies ist ein Block-Element</h1>

</body>

</html>

inlineBlock.css

span, u, strong {  
 width: 300px;  
 height: 300px;  
}

div {  
 width: 300px;  
 height: 300px;  
}

p, div, h1 {  
 background-color: #00F;  
}

## DISPLAY-Eigenschaft

https://wiki.selfhtml.org/wiki/CSS/Eigenschaften/Anzeige/display

Mit der Eigenschaft display wird festgelegt, welche Art von Box ein Element erzeugt. Jedes Element kann Null oder mehr Boxen erzeugen. Somit könne aus Block-Elementen inline-Elemente und aus Inline-Elementen Blockelemente geschaffen werden. Ein Mittelweg (InlineBlock) ist ebenfalls möglich.

Mit der Eigenschaft display wird festgelegt, welche Art von Box von einem Element erzeugt wird. Jedes Element kann null oder mehr Boxen erzeugen.

Die Werte, die die display-Eigenschaft besitzen kann, werden nachfolgend mit den typischen Verhaltensweisen der Elemente im normalen Elementfluss beschrieben.

### inline: Inline-Box(en)

Elemente mit display: inline; erzeugen eine oder mehrere Inline-Boxen. Inline-Boxen verlaufen auf einer Zeile horizontal in der Schreibrichtung der verwendeten Sprache. Im Deutschen also von links nach rechts, in hebräischen oder arabischen Texten von rechts nach links. Eine Inline-Box wird in mehrere Teile zerlegt, wenn sie nicht in einer Zeile Platz hat oder Zeilenumbrüche enthält.

### block: Block-Box

Block-Boxen nehmen die gesamte Breite des Elternelementes ein. Sie sind so hoch wie ihr Inhalt. Dadurch entsteht ein zusammenhängendes Rechteck, das aussieht wie eine Box und dieser Darstellungsart ihren Namen gibt.

Mehrere Block-Boxen werden untereinander angeordnet (normaler Elementfluss).

### inline-block: Außen Inline-Box, innen Block-Box

Elemente mit display: inline-block; erzeugen eine Box, die Eigenschaften von Inline- und Block-Boxen kombiniert.

Eine Inline-Block-Box verläuft wie eine Inline-Box auf einer Textzeile, kann aber nicht auf mehrere Zeilen aufgeteilt werden. Besitzt ein Inline-Block vertikale Innen- und Außenabstände oder Rahmen, so beeinflussen diese die Höhe der Zeile. Die horizontale und vertikale Ausrichtung von Inline-Block-Elementen erfolgt wie bei Inline-Boxen.

Die Breite einer Inline-Block-Box wird mit dem *shrink-to-fit*-Verfahren berechnet, d.h. die Box ist so breit wie nötig und so schmal wie möglich. Die Breite eines Inline-Blocks kann mithilfe der width-Eigenschaft festgelegt werden.

Die Höhe einer Inline-Block-Box ist wie bei Block-Boxen vom Inhalt abhängig, kann aber auch durch die height-Eigenschaft festgelegt werden.

Inline-Blöcke erzeugen eine Block-Formatierungs-Umgebung, daher können sie – im Gegensatz zu Inline-Boxen – Block-Boxen enthalten.

Beispiel:  
<!DOCTYPE *html*>  
<html *lang*="de">  
<head>  
 <meta *charset*="utf-8">  
 <title>display-Eigenschaft</title>  
 <link *rel*="stylesheet" *type*="text/css" *href*="displayEigenschaft.css">  
</head>

<body>

<span *class*="test1">Wir stellen jetzt dieses Span </span>  
<span *class*="test1">statt Inline-Element als Block-Element dar.</span>

<div *class*="test2">Text!</div>

<div *class*="test2">Text!</div>

<div *class*="test3">Text!</div>

<div *class*="test3">Text!</div>

</body>

</html>

Css

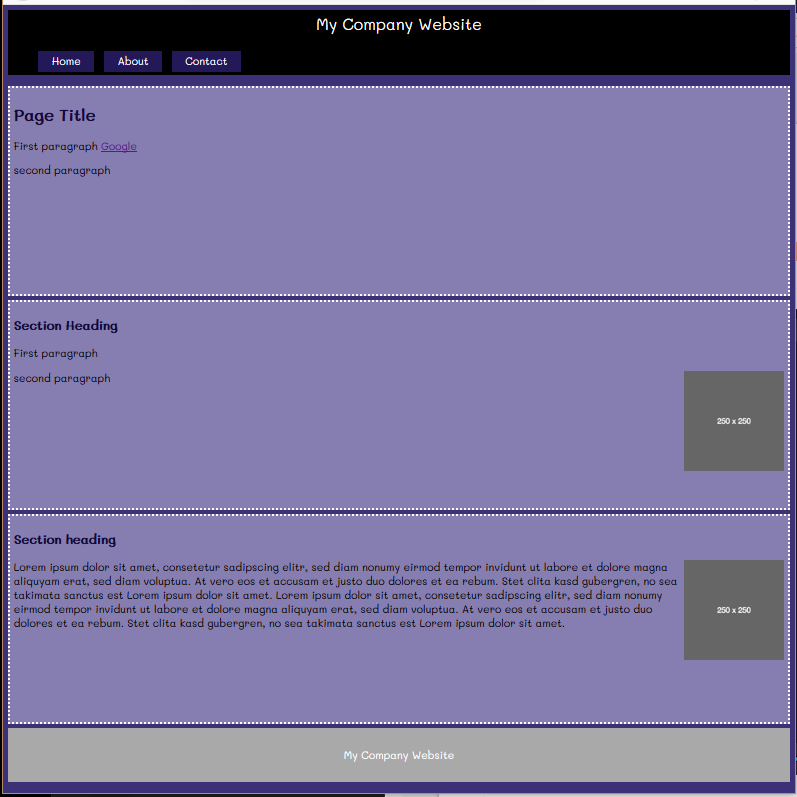
.test1 {  
 display: block;  
 background-color: #F00;  
 width: 300px;  
 height: 300px;  
 margin-bottom: 20px;  
}

.test2 {  
 width: 300px;  
 height: 300px;  
 background-color: #0F0;  
 margin-bottom: 20px;  
 display: inline;  
}

.test3 {  
 width: 300px;  
 height: 300px;  
 background-color: #000;  
 margin-bottom: 20px;  
 color: #FFF;  
 display: inline-block;}

## Aufgabe

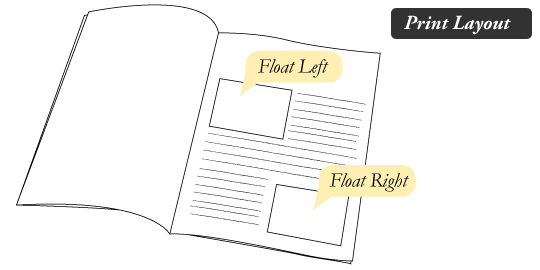
Erstellen Sie folgendes Aussehen für eine Webseite. Die entsprechenden Ausgangsdateien finden Sie im Ordner Exercise 3. Folgen Sie den Anweisungen in der Datei exercise\_students.css und binden Sie diese in ihr HTML-Dokument mit ein.

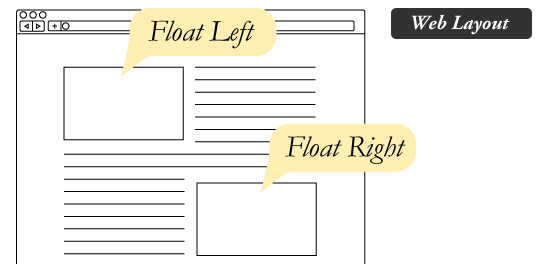


Lösung: exercise3.css

# Float und Clear

https://css-tricks.com/all-about-floats/

**Float** bezeichnet eine Eigenschaft zur Positionierung. Um seinen Zweck zu verstehen, könne wir ein Print-Design betrachten. So können beispielsweise Bilder so gesetzt werden, dass sie von Text umflossen werden.

In Textsatzprogrammen können die Textboxen angewiesen werden, diesen Textumbruch zu beachten oder ihn zu ignorieren. Dies würde beispielsweise dazu führen, dass Text das Bild überschreibt. Das Bild ist damit entweder Teil des Textflusses (Flow) oder eben auch nicht. WebDesign ist hier dem Printdesign sehr ähnlich.

Im WebDesign sind Seitenelemente, die mit der Float-Eigenschaft ausgestattet werden, wie die Bilder im Printdesign zu sehen. Umflossene Elemente bleiben Teil des Textflusses einer Webseite. Dies unterscheidet sich von Seitenelementen, die absolute Positionierung verwenden. Dies werden aus dem Textfluss der Webseite herausgenommen und beeinflussen nicht die Positionierung der anderen Elemente und umgekehrt.

Die float-Eigenschaft wird in CSS wie folgt gesetzt:

#sidebar {

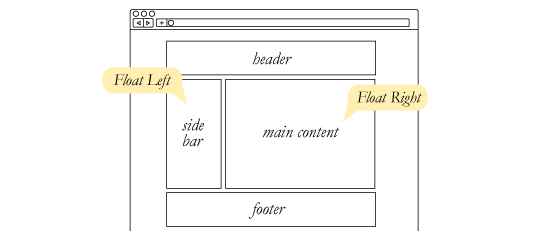
float: right;

}

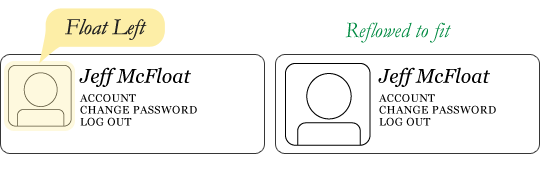
Es gibt vier gültige Werte für die float-Eigenschaft: Left und Right lassen Elemente in die jeweilige Richtung fließen. None (default verhindert den Fluss und Inherit übernimmt den Wert vom übergeordneten Element.

### Einsatzgebiet

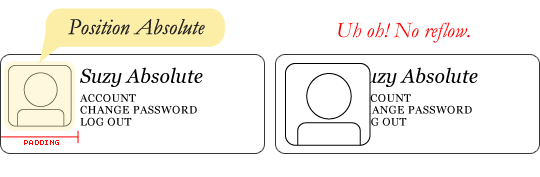
Neben dem oben angesprochenen Beispiel des Umfließens von Bildern können floats zum Erstellen eines ganzen Weblayouts verwendet werden.

Auch wenn Web-Layouts heute durch Flexbox und Grid-Layout erstellt werden, haben floats doch einige besondere Eigenschaften.

Floats sind auch hilfreich bei kleinen Problemen. Wenn wir den Avatar in der nebenstehenden Abbildung ändern, wird der Textfluss entsprechend angepasst.

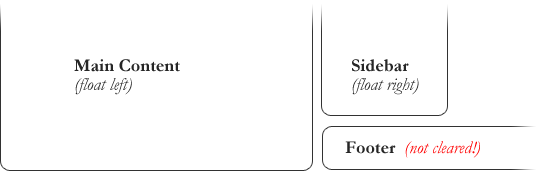


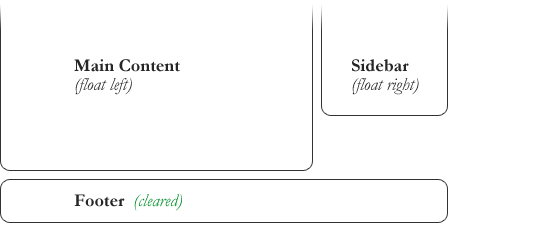
Dasselbe Layout könnte auch mit einer relativen Positionierung für den Container und einer absoluten Positionierung für den Avatar erreicht werden, Der Text wäre aber jetzt vom Avatar nicht abhängig und würde sich bei einer Größenänderung nicht neu ausrichten.



### Clearing the Float

Die zu float gegensätzliche Eigenschaft ist clear. Ein Element mit dieser Eigenschaft wird sich nicht am float ausrichten, sondern sich hinter den float anordnen. Ein Beispiel soll dies verdeutlichen.



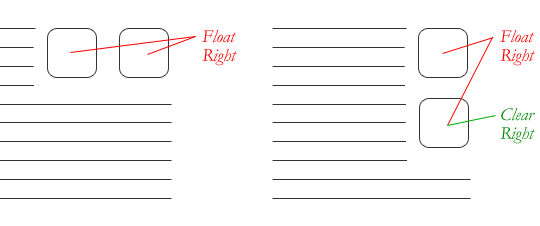
Im obigen Beispiel ist die Sidebar rechts vom MainContent und etwas kleiner. Der footer muss nun diese Lücke aufnehmen und das obige Layout ist die Folge. Um dieses Problem zu lösen, kann der Footer mit einer clear-Eigenschaft versehen werden, um seine unten ausgerichtete Anordnung zu garantieren.

#footer {

clear: both;

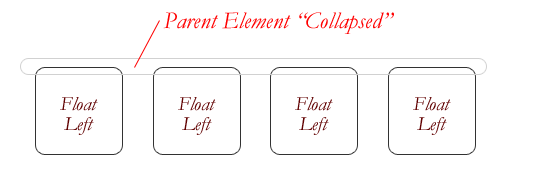
}

Clear hat ebenfalls vier Eigenschaften, left, right, both, Inherit, None. Both ist die am meisten genutzte, die floats aus allen Richtungen beendet.

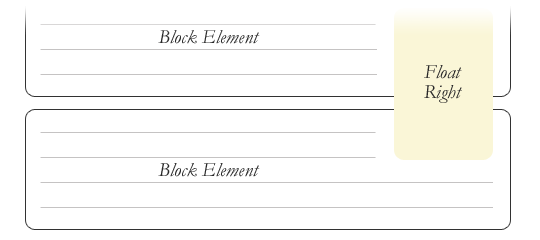


### The Great Collapse

Eines der etwas befremdlichen Dinge im Umgang mit Floats ist der Umstand, wie Floats ihre Eleternelemente beeinflussen können. Wenn das Elternalement nichts anderes als gefloatete Elemente beinhaltet, schrumpft die Höhe des Elternelementes auf 0. Dies ist nicht immer sofort ersichtlich, aber es ist wichtig zu wissen.



As anti-intuitive as collapsing seems to be, the alternative is worse. Consider this scenario:



If the block element on top were to have automatically expanded to accommodate the floated element, we would have an unnatural spacing break in the flow of text between paragraphs, with no practical way of fixing it. If this were the case, us designers would be complaining much harder about this behavior than we do about collapsing.

Collapsing almost always needs to be dealt with to prevent strange layout and cross-browser problems. We fix it by clearing the float **after** the floated elements in the container but **before** the close of the container.

### [#](https://css-tricks.com/all-about-floats/#article-header-id-7)Video

I did [a screencast](http://css-tricks.com/video-screencasts/42-all-about-floats-screencast/) a while back explaining many of these float concepts.

## Aufgabe

Quelle: https://github.com/char1eschen/40-Layout-Exercises

Speichern sie folgendes Fragment als vollständige HTML-Datei und schreiben Sie dazu Fülltext in die jeweiligen divs navigation, exctra und footer.

<div id="container">

<div id="header">Header</div>

<div id="wrapper">

<div id="content">Content</div>

</div>

<div id="navigation">Navigation</div>

<div id="extra">Extra stuff</div>

<div id="footer">Footer</div>

</div>

Binden Sie dann folgende CSS-Datei ein

#container {

font-size: 12px;

font-family: arial, sans-serif;

}

p {

margin: 0 10px 10px;

}

#header {

height: 80px;

background-color: **rgb**(238,238,238);

color: **rgb**(122,177,36);

padding-left: 10px;

line-height: 80px; }

#header h1 {

margin: 0;

}

#navigation {

background-color: **rgb**(186,203,253);

}

#extra {

background-color: **rgb**(253,133,67);

}

#footer {

height: 14px;

padding:5px 10px 5px 10px;

color: #FFF;

background-color: #333;

}

#footer p {

line-height: 14px;

margin: 0px;

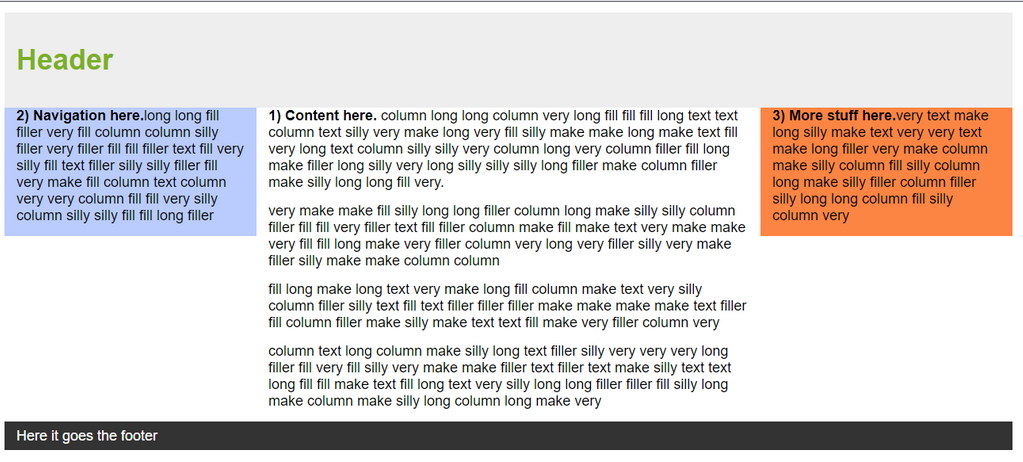
}

Formatieren Sie dann die bestehende CSS-Datei so um, dass folgendes Layout entsteht.

Setzen Sie dazu die

* die width des wrapper-divs auf 50 % und zentrieren Sie das wrapper-Element horizontal
* die width des navigation und extra – div auf jeweils 25 %
* navigation soll links vom wrapper, extra soll rechts vom wrapper erscheinen,

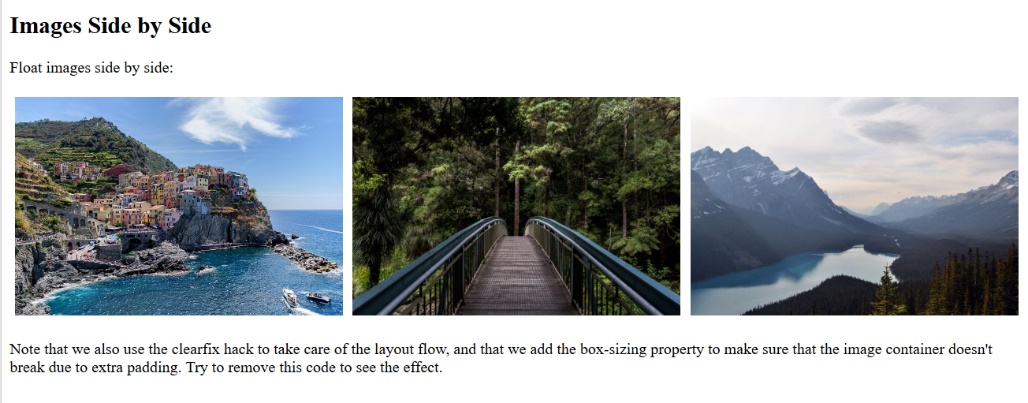
Informieren Sie sich dazu über die float und clear – Eigenschaft von css



Layout – float und clear

<https://www.w3schools.com/css/tryit.asp?filename=trycss_float5>

<https://www.w3schools.com/css/css_float.asp>

<https://www.w3schools.com/css/tryit.asp?filename=trycss_layout_cols>

# CSS/Eigenschaften/Positionierung/position

Quellen:

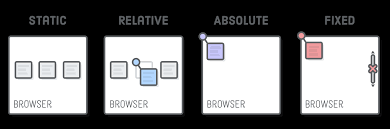
<https://www.codeproject.com/Articles/567385/CSSplusBoxplusModelplusandplusPositioning>

<http://www.thestyleworks.de/basics/disposfloat.shtml>

http://selfhtml.apsel-mv.de/position/position.html

<https://medium.freecodecamp.org/how-to-use-the-position-property-in-css-to-align-elements-d8f49c403a26>

https://www.w3schools.com/css/css\_positioning.asp

Elemente können durch Verwendung der Eigenschaft **position** aus dem normalen Elementfluss entfernt werden und an jede beliebige Stelle des Viewports positioniert werden. Auf andere Elemente haben positionierte Elemente keinen Einfluss.

**position: static**

position:static ist der Defaultwert der Eigenschaft position. Das Element verbleibt im Textfluss. Die Eigenschaften Top, Bottom, Left, Right werden ignoriert.

**position: absolute**

Mit position:absolute kann man Elemente losgelöst vom Textfluss positionieren, an eine Stelle, die per left, right, top und bottom festgelegt wird. Eine Lücke bleibt dabei nicht bestehen. Größenangaben, wie width und height, oder Abstände, wie margin und padding, sind ebenfalls möglich.

Die Angaben left, right, top und bottom beziehen sich dabei auf das **nächste Vorfahrenelement**, welches mit position: positioniert wurde. Hat das Element keine solchen Vorfahrenelemente, wird als Bezugspunkt das Wurzelelement angesehen - in HTML-Dokumenten also das HTML-Element (bei XML wäre es das XML-Element). Im oberen Beispiel wird deshalb der Link mit der ID "info-box" relativ zum main-Element positioniert, wofür das main-Element die Eigenschaft position:relative erhält.

**position: fixed**

Möchte man ein Element starr am Viewport ausrichten, (beispielsweise eine [fixe Fußzeile](https://wiki.selfhtml.org/wiki/CSS/Tutorials/feststehender_footer) oder einen Link zum Seitenanfang in einer Ecke) dann ist position:fixed das Mittel der Wahl. Genau wie bei position:absolute wird das Element aus dem Textfluss entnommen, es entsteht also keine Lücke. Positioniert wird klassisch mit den CSS-Eigenschaften top, bottom, left und right.

**position: relative**

Wie schon weiter oben erklärt, verwendet man position:relative als Bezugspunkt für absolut positionierte Kindelemente. Das Element an sich wird quasi "an sich selbst" ausgerichtet. Die Lücke im Textfluss bleibt bestehen. Trotzdem kann man das Element mit top, bottom, left und right ausrichten.

position: sticky

position: sticky: Elemente mit dieser Eigenschaft behalten ihre Position im Elementfluss, bis sie das obere oder untere Seitenende erreichen und dort „kleben“ bleiben.

Ein Beispiel soll dies veranschaulichen:

Static: Erzeugen Sie folgende Ausgabe mit Hilfe zweier DIVs

|  |  |
| --- | --- |
|  | <style>  .box-orange{ */\*keine Positionsangabe\*/*  background:orange;  height: 100px;  width: 100px;  }  .box-blue{  background: lightskyblue;  height: 100px;  width: 100px;  position: static; */\* Declared as static \*/*  }  </style>  </head>  <body>  <p>Position Static ist der default- Wert.  Die Elemente bleiben in ihrer normalen Position</p>  <div class="box-orange"></div>  <div class="box-blue"></div>  </body>  </html> |

**Relative:**   
Die Position eines Elements ist relativ zu seiner normalen Position innerhalb des Textflusses. Um eine Änderung zu sehen, müssen Eigenschaften wie top und left angepasst werden.

Ausgehend vom vorherigen Beispiel soll die orangene Box vor die blaue Box geschoben werden.

|  |  |
| --- | --- |
| 100 px vom top der alten Position  100 px von links der alten Position | <style>  .box-orange{  position: relative; */\* We can now move the element\*/*  background: orange;  width: 100px;  height: 100px;  top: 100px; */\* 100px from top relative to its old position\*/*  left: 100px; */\* 100px from left \*/*  }  .box-blue{  background: lightskyblue;  height: 100px;  width: 100px;  position: static;  }  </style> |

**Absolute:**

Mit position: absolute wird ein Element relativ zu seinem Elternelement ausgerichtet. Wenn es kein Elternelement hat, wird es zum <html>-Element ausgerichtet.

Das Element wird außerdem aus dem normalen Textfluss herausgenommen und kann damit die Position andere Elemente ebenfalls beeinflussen.  
Im ersten Beispiel wird die orangene Box herausgenommen, die blaue Box bleibt in ihrer Position und wird von der orangenen Box überdeckt.

|  |  |
| --- | --- |
| <body>  <div class="container">  <div class="box-orange"></div>  <div class="box-blue"></div>  </div>  </body>  </html> | <style>  .box-orange{ */\*keine Positionsangabe\*/*  position: absolute;  background: orange;  width: 100px;  height: 100px;  }  .box-blue{  background: lightskyblue;  height: 100px;  width: 100px;  position: static  }  </style> |
| Die Änderung der Position lässt die blaue Box erscheinen. | .box-orange {  position: absolute;  background: orange;  width: 100px;  height: 100px;  left: 5px;  top: 5px;  } |
| Die Koordinaten eines absolut positionierten Elementes sind relativ zum Elternelement, wenn dieses nicht-static definiert ist. | .container {  position: relative;  background: lightgray; }  .box-orange {  position: absolute;  background: orange;  width: 100px;  height: 100px;  right: 5px; /\* 5px relative to the most-right of parent \*/ } |

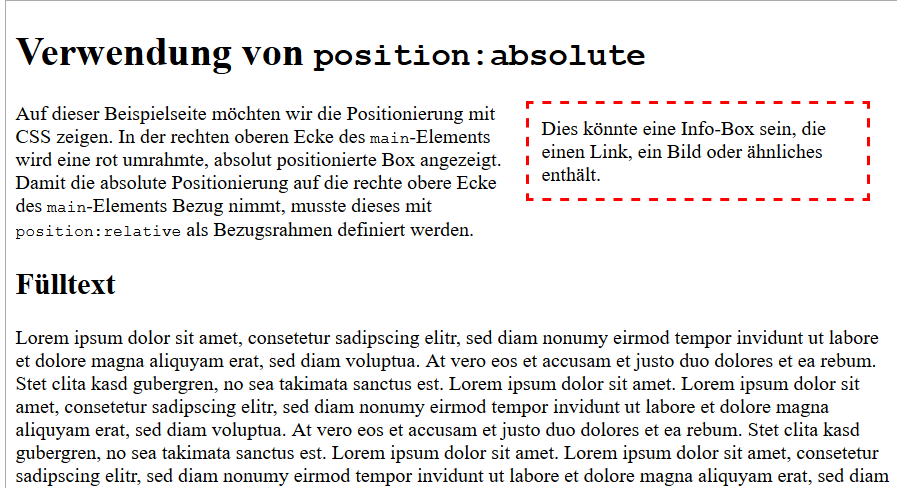
**Fixed:**

Auch sie werden aus dem normalen Textfluss herausgenommen. Im Unterscheid zu absolute sind sie allerdings nur relative zum html-Element und damit unbeinflusst vom Scrollen im Bowser

|  |  |
| --- | --- |
| <body>  <div class="container">  <div class="box-orange"></div>  <div class="box-blue"></div>  <p>Scroll down the page</p>  </div>  </body> | .container {  position: relative;  background: lightgray;  width: 50%;  margin: 0 auto;  height: 1000px;}  .container p {  text-align: center;  font-size: 20px;}  .box-orange {  background: orange;  width: 100px;  height: 100px;  position: fixed;  right: 5px;}  .box-blue {  background: lightblue;  width: 100px;  height: 100px;  } |

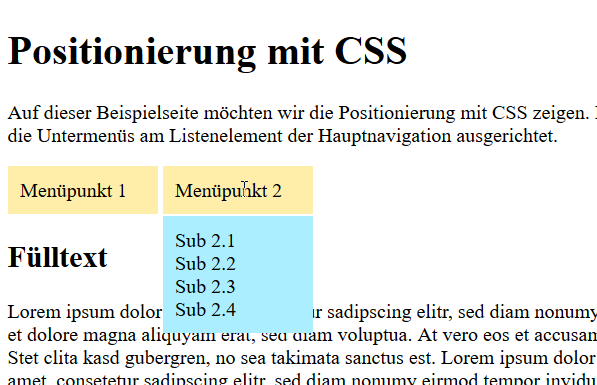
# Aufgabe zu Position

A1. Erzeuge Sie mit Hilfe der HTML-Datei folgende Optik

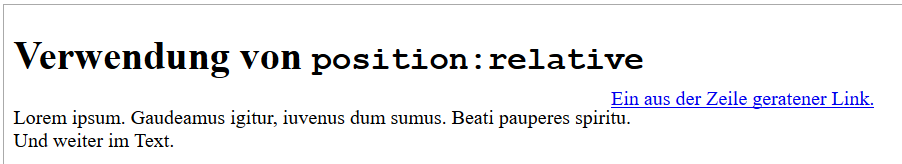


A2:

Erzeugen Sie folgende Menüstruktur



A3: Erzeuge Sei folgendes Aussehen

Der Link ist Teil des p-Tags, soll allerdings etwas oberhalb positioniert werden.

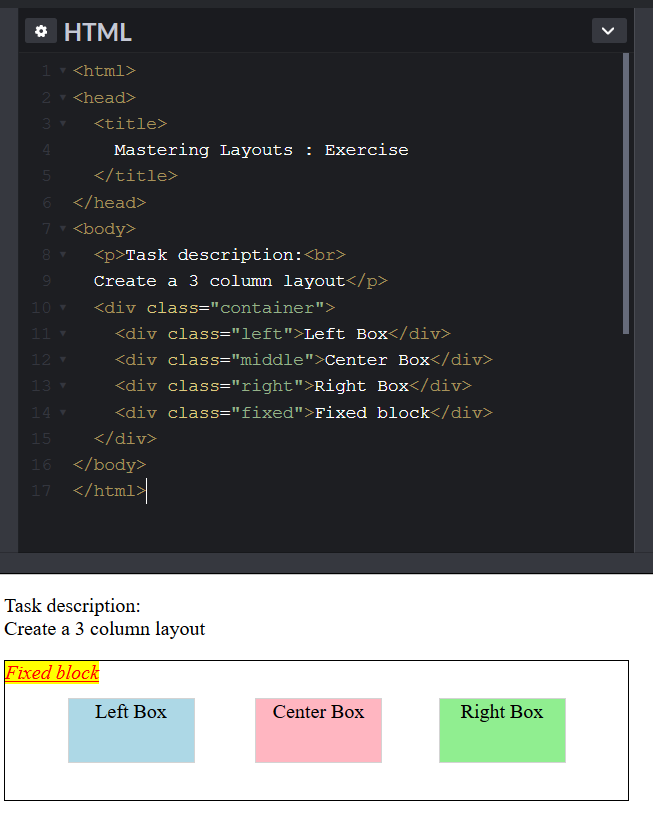
A4. Untenstehender Link soll sich beim Browsen nicht in seiner Position verändern



A4. Erstellen Sie aus der HTML-Datei folgende Optik

Aufgabe:

Erzeugen Sie folgendes Aussehen untenstehender CSS-Datei.



A3:

# Größenangaben

# Arten von Farben

# Schriftarten

# Glossar

Quelle: http://www.thestyleworks.de/basics/definitions.shtml

# Die wichtigsten Definitionen in CSS

Absolute Positionierung (absolute positioning)

Ein Element ist dann absolut positioniert, wenn seine Eigenschaft position mit [absolute](http://www.thestyleworks.de/ref/position.shtml#absolute) oder [fixed](http://www.thestyleworks.de/ref/position.shtml#fixed) deklariert ist. Die Box eines absolut positionierten Elements ist vom normalen Elementfluß vollständig losgelöst und hat keinerlei Einfluss auf nachfolgende [Geschwisterelemente](http://www.thestyleworks.de/basics/doctree.shtml#sibling). Sie bildet den umschließenden Block für alle ihre [Nachkommenelemente](http://www.thestyleworks.de/basics/doctree.shtml#descendant) mit Ausnahme der fixed positionierten Elemente. Ihre Inhalte fließen nicht um andere Boxes herum ebenso wie die Inhalte anderer Boxes nicht von absolut positionierten Boxes beeinflusst werden. Wie überlappende Boxes sich gegenseitig überdecken, hängt vom [Z-Index](http://www.thestyleworks.de/ref/z-index.shtml) ab.

Absoluter Längenwert (absolute length)

Absolute Längenwerte sind Zentimeter, Millimeter, Inch(Zoll), Points oder Picas. Längenwerte werden in einem [besonderen Artikel](http://www.thestyleworks.de/basics/lengthvalues.shtml) ausführlich behandelt.

Aktueller Wert (actual value)

Relative Werte werden als Verhältnismaße zu anderen Größen definiert. Deshalb muss aus dem in einer CSS-Regel deklarierten Wert ein am Bildschirm darstellbarer Wert bestimmt werden. Dazu sind die folgenden vier Berechnungsschritte erforderlich:

Spezifizierter Wert → Berechneter Wert → Verwendeter Wert → Aktueller Wert.

Der Ablauf dieser Berechnung ist im Abschnitt [Relative Längenwerte](http://www.thestyleworks.de/basics/lengthvalues.shtml#relative) detailliert erläutert.

Ancestor

Vorfahrenselektor / Vorfahrenelement. Wenn sich die Abstammungslinie von einem Element A nach unten direkt zu einem Element B verfolgen lässt, dann wird A als Vorfahrenelement von B bezeichnet. Dabei kann B das direkte Kindelement von A sein, es können aber auch beliebig viele Generationen dazwischen liegen. Diese und weitere Erläuterungen zum Konzept der Abstammung in CSS finden sich im Grundlagenartikel '[Der Dokumentstammbaum](http://www.thestyleworks.de/basics/doctree.shtml)'.

Anonyme Boxes (anonyme boxes)

Alle Elemente einer Website erzeugen Boxes: Block- Level- Elemente generieren Block- Boxes, Inline- Level- Elemente generieren Inline- Boxes und alle Inhalte, die zwischen den anderen Elementen liegen, erzeugen anonyme Boxes. Dadurch wird die Definition der Formatierung innerhalb des Box- Modells und des Line- Box- Modells vereinfacht. Der Inhalt von anonymen Boxes kann nicht durch CSS-Deklarationen angesprochen werden.

Hat nun ein Block- Level- Element eine oder mehrere andere Block- Level- Elemente als Kindelement und dazwischen noch einige andere Inhalte, dann werden zunächst für die Kindelemente Block- Boxes gebildet. Die anderen Inhalte, die zwischen den generierten Block- Boxes liegen, werden dann in anonyme Block- Boxes eingefasst. Eine anonyme Block- Box kann Inline- Level- Elemente wie auch freistehenden Text oder anderes nebeneinander enthalten.

Sind aber keine Block- Level- Elemente als Kindelemente vorhanden, sondern nur Inline- Level- Elemente und andere Inhalte, dann werden zunächst Inline- Boxes für die Kindelemente generiert, danach bilden die dazwischen liegenden Inhalte anonyme Inline- Boxes.

Dasselbe gilt, wenn sich innerhalb einer anonymen Block- Box nur Inline-Elemente und weitere Inhalte befinden. Diese bilden dann Inline- Boxes und anonyme Inline- Boxes. Eine anonyme Block- Box kann keine weiteren Block- Boxes enthalten.

Anonyme Block- Box (anonyme block box)

Eine Block- Box kann auch andere Block- Boxes beinhalten, die durch Kindelemente generiert werden. Sind zudem andere Inhalte vorhanden, werden diese von (gedachten) anonymen Block- Boxes umgeben. Das bewirkt, dass in diesem Fall die erste Block- Box nur Kindelemente auf Block- Level- Ebene enthält und dass die Definition der Formatierung vereinfacht wird. Ein Beispiel dazu wird im Artikel '[Das Box- Modell](http://www.thestyleworks.de/basics/boxmodel.shtml#anonym)' vorgestellt.

Anonyme Inline- Box (anonyme inline box)

Alle Teile eines Textes oder einer Textzeile, die keinem Inline- Level- Element zugeordnet sind, werden Anonyme Inline- Boxes genannt. Der Inhalt von anonymen Inline- Boxes kann nicht durch CSS-Deklarationen angesprochen werden. Die Regeln der Inline- Formatierung in CSS werden im Artikel über das [Line- Box- Modell](http://www.thestyleworks.de/basics/lineboxmodel.shtml#inline-example) ausführlich dargestellt.

Anwenderprogramm (user application)

Dieser Begriff wird auf dieser Website als übergreifender Ausdruck für alle Programme verwendet, die Websites darstellen können: normale Browser, Homepage-Reader, Brailleleser, Voice-Browser, PDA- oder Mobilephone-Browser (Smartphone) etc.

Attribut (attribute)

HTML- und XHTML-Elemente können Eigenschaften zugeordnet haben, die Attribute genannt werden. Es ist für jedes Element genau definiert, welche Attribute sie haben können und welche Art von Werten diese Attribute annehmen können. Das folgende Beispiel zeigt den HTML- Code einer Überschrift 1. Ordnung, der ein Attribut ID mit einem Wert von kapitel-1 zugeordnet wurde.

<h1 id="kapitel-1">...</h1>

Äußere Kante (outer edge)

Die Darstellung von padding, border und margin ist genaugenommen nicht linienförmig, sondern deckt eine bestimmte Fläche ab. Die Begriffe äußere Kante und innere Kante dienen dazu, die Begrenzungen dieser Flächen wie auch der Contentflächen im Box- Modell exakter zu beschreiben.

Backslash(\) Escapes

Ein Backslash kann in CSS verschiedene Bedeutungen haben:

* Er hebt die besondere Bedeutung spezieller CSS-Zeichen auf. Jedem Zeichen mit Ausnahme hexadezimaler Zeichen kann durch ein vorangestelltes '\' seine spezielle Bedeutung genommen werden.
* Mit Hilfe des Backslash lassen sich Sonderzeichen in ein Webdokument einfügen, die anders nur schwer zu erzeugen wären. Dazu müssen auf den Backslash bis zu sechs Hexadezimalziffern (0-9, a-f) folgen, die für ein Zeichen nach ISO 10464 stehen. Diese Sequenz wird von den nachfolgenden Zeichen abgetrennt, indem man einen Leerraum nach ihr einfügt oder genau sechs Hexadezimalziffern verwendet.
* Ein Backslash am Ende der Zeile muss vom Anwenderprogramm ignoriert werden.

Dieses Escape-Zeichen wird von allen modernen Browsern (Gecko-Engine, IE/Mac, Opera 6+, Safari usw.) korrekt interpretiert, nicht jedoch vom IE/Win 6 im [Quirksmodus](http://www.thestyleworks.de/basics/doctypes.shtml#switching), vom IE/Win 5 gar nicht. NN 4 wird das gesamte Stylesheet ignorieren.

Berechneter Wert (calculated value)

Relative Werte werden als Verhältnismaße zu anderen Größen definiert. Deshalb muss aus dem in einer CSS-Regel deklarierten Wert ein am Bildschirm darstellbarer Wert bestimmt werden. Dazu sind die folgenden vier Berechnungsschritte erforderlich:

Spezifizierter Wert → Berechneter Wert → Verwendeter Wert → Aktueller Wert.

Der Ablauf dieser Berechnung ist im Abschnitt [Relative Längenwerte](http://www.thestyleworks.de/basics/lengthvalues.shtml#relative) detailliert erläutert.

Bidirektionalität (bidirectionality)

Normalerweise ist in allen Webdokumenten, die in westlichen Sprachen abgefasst sind, die Schreibrichtung von links nach rechts. In einigen Dokumenten, insbesondere in hebräischer oder arabischer Sprache, kann in einem Block- Element Text beider Schreibrichtungen (Direktionalitäten) auftreten. Dies nennt man Bidirektionalität oder kurz 'bidi'. Genauere Erklärungen über die Behandlung verschiedener Schreibrichtungen in CSS 2.1 werden bei den Eigenschaften [direction](http://www.thestyleworks.de/ref/direction.shtml) und [unicode-bidi](http://www.thestyleworks.de/ref/unicode-bidi.shtml) gegeben.

Block- Box (block box)

Eine Block- Box ist die Box, die durch ein Block- Level- Element erzeugt wird. Der exakte Aufbau einer Block- Box wird durch das Box- Modell definiert. Das Box- Modell wird ausführlich im Artikel '[Das Box- Modell](http://www.thestyleworks.de/basics/boxmodel.shtml)' erläutert.

Block- Level- Element (block level element)

Ein Block- Level- Element ist ein Element, das einen Kontext zur Blockformatierung etabliert. Siehe dazu [Kontext zur Blockformatierung](http://www.thestyleworks.de/basics/boxmodel.shtml#blockcontext) im Grundlagenartikel 'Das Box- Modell'.

Box- Modell (box model)

Das Box- Modell mit seinen Regeln ist einer der wichtigsten Teile von CSS 2.1. Es dient als Grundlage der Darstellung und Positionierung aller Elemente des sogenannten Kontexts zur Blockformatierung, ebenso zur Wiedergabe anderer visueller Eigenschaften und für dynamische Effekte mit Hilfe von Scripten. Eine umfassende Erläuterung des Box- Modells ist im Artikel '[Das Box- Modell](http://www.thestyleworks.de/basics/boxmodel.shtml)' zu finden.

Collapsing margins

Unter bestimmten Umständen kann die untere margin eines Block- Level- Elements mit der margin des darauf folgenden Elements zusammenfallen. Im Referenzartikel 'margin' sind die exakten Abläufe dieses Zusammenfallens im Unterthema '[Zusammenfallende Abstände](http://www.thestyleworks.de/ref/margin.shtml#collaps)' erläutert, darüber hinaus gibt es auch ein [praktisches Beispiel](http://www.thestyleworks.de/basics/margins.shtml).

Dargestellter Inhalt (rendered content)

Der Inhalt eines Elementes, wie er am Bildschirm dargestellt wird, entspricht nicht dem im Quellcode stehenden Inhalt. Der dargestellte Inhalt entsteht aus dem Quellcode, nachdem alle in Frage kommenden Stylesheets ausgewertet wurden, alle Tabellen vollständig geladen sind, usw. Dazu gehören auch Listenbullets, durch das Stylesheet generierte Inhalte und die Werte der Attribute alt und title. Der dargestellte Inhalt replazierter Elemente wird von Quellen außerhalb des aktuellen Dokuments bezogen, z B. aus Grafikdateien.

Deklaration (declaration)

Jede Deklaration besteht aus einer Eigenschaft mit den zugehörigen Werten. Es ist wichtig, dass immer beide Komponenten vorhanden sind, da es sonst zu unvorhersehbaren Ergebnissen kommen kann. Deklarationen, bei denen ein Wert fehlt oder ein unerwartetes Syntaxzeichen auftritt, müssen vom Browser ignoriert werden.

Jede Deklaration wird durch ein Semikolon von der nachfolgenden getrennt, d. h. theoretisch ist das Semikolon nach der letzten Deklaration überflüssig. Dennoch ist es besser, auch dieses Semikolon stets zu notieren. Man vermeidet dadurch potentielle Probleme, falls die Regeln zu einem späteren Zeitpunkt noch einmal umgestellt werden sollen.

Leerzeichen oder Zeilenumbrüche nach dem Semikolon einer Deklaration werden ignoriert. Es spielt also für die Funktion der Stylesheets keine Rolle, ob alle Deklarationen eines Deklarationsblocks hintereinander in einer Reihe stehen oder ob sie untereinander notiert sind.

Deklarationsblock (declaration block)

Der Deklarationsblock umfasst den gesamten Teil einer Regel, der hinter den Selektoren liegt: die geschweiften Klammern und alle darin notierten Deklarationen. Leere Deklarationsblöcke, die nur aus zwei Klammern bestehen, sind zulässig. Das hat den Effekt, dass keine Formatierungen an die korrespondierenden HTML-Elemente angebracht werden.

Dokumententyp (document type)

Die Angabe eines Dokumententyps am beginn eines Webdokuments ist seit HTML 4.01 grundsätzlich notwendig, wenn es standardgerecht durch die Browser dargestellt werden soll. Im Artikel [Der Dokumententyp](http://www.thestyleworks.de/basics/doctypes.shtml) werden mehr Informationen zur DOCTYPE- Deklaration und zum DOCTYPE- Switching gegeben.

Dokumentstammbaum (document tree)

Alle Elemente einer Website können in einer Baumstruktur ähnlich der eines Familienstammbaums aufgezeichnet werden. Daraus lassen sich dann schnell die Verhältnisse der Elemente zueinander ablesen. Dem Dokumentstammbaum ist auf dieser Website ein [spezieller Artikel](http://www.thestyleworks.de/basics/doctree.shtml) gewidmet.

Durchschuss / Halb-Durchschuss (leading / half-leading)

Der Durchschuss ist die Differenz zwischen dem [aktuellen Wert](http://www.thestyleworks.de/basics/definitions.shtml#actualvalue) der Eigenschaft line-height und der Höhe der Textzeichen in einer Zeile. Die Hälfte dieser Differenz ist der Halb-Durchschuss. Der Browser bringt einen vertikalen Abstand von der Höhe eines Halb-Durchschuss oben und unten an jedem Textzeichen an, bevor er es entsprechend dem Wert der Eigenschaft vertical-allign vertikal ausrichtet. Die Regeln der Inline- Formatierung in CSS werden im [Artikel über das Line- Box- Modell](http://www.thestyleworks.de/basics/lineboxmodel.shtml) ausführlich dargestellt.

Eigenschaft (property)

Eine nach der CSS-Spezifikation zugelassene Eigenschaft, wird bei der Notierung im Stylesheet gefolgt von einem Doppelpunkt (:). Alle in CSS 2.1 möglichen Eigenschaften werden in der [Referenz](http://www.thestyleworks.de/ref/index.shtml) besprochen.

Einfach- Selektor (simple selector)

Ein Einfach- Selektor ist ein Typselektor oder ein Universalselektor, der entweder alleine steht oder an den Attributselektoren, ID-Selektoren oder Pseudoklassen in beliebiger Reihenfolge angehängt sind. Der Einfach- Selektor korrespondiert mit den Elementen des Dokuments, auf die jede seiner Komponenten zutrifft. Mehrere Einfach- Selektoren können durch Kombinatoren zu einem Selektor vekettet werden. Mehr über die Syntax der CSS-Regeln ist im Artikel [Regelaufbau](http://www.thestyleworks.de/basics/rulesyntax.shtml) erklärt.

Eltern-Element (parent element)

Das Elternelement eines Elementes A ist das Element, von dem A direkt abstammt. Es gibt keine Selektoren, die auf das Elternelement zeigen. Diese und weitere Erläuterungen zum Konzept der Abstammung in CSS finden sich im Grundlagenartikel [Der Dokumentstammbaum](http://www.thestyleworks.de/basics/doctree.shtml).

Element (element)

Elemente sind die syntaktischen Primärkonstrukte in HTML und XHTML. Die Namen der Elemente, DIV, P, EM usw., werden in CSS als Einfach- Selektoren benutzt.

Farbwerte (color values)

Farben werden auf Websites als Vorder- und Hintergrundfarben verwendet. Sie können als Zahlenwerte oder Schlüsselworte auftreten. Dies wird ausführlicher im [Artikel über Farbwerte](http://www.thestyleworks.de/basics/colors.shtml) erläutert.

float-Positionierung (float positioning)

Mit Hilfe der Eigenschaft float lassen sich Block- Boxes auf eine Seite (links oder rechts) der Website schieben, sodass die restlichen Inhalte auf der Innenseite der Box vorbeifließen. Mehr dazu kann man im Artikel über [CSS-Layouts mit float und clear](http://www.thestyleworks.de/tut-art/layout_div_2.shtml) und im Referenzartikel der [Eigenschaft float](http://www.thestyleworks.de/ref/float.shtml) nachlesen.

Font-Eigenschaften (font properties)

Steht zusammenfassend für alle Eigenschaften, mit denen man Schrift formatieren kann. Die Kurzschrift- Eigenschaft [font](http://www.thestyleworks.de/ref/font.shtml) beeinflusst die Einzeleigenschaften für [Textneigung](http://www.thestyleworks.de/ref/font-style.shtml), [Kapitälchen](http://www.thestyleworks.de/ref/font-variant.shtml), [Schriftstärke](http://www.thestyleworks.de/ref/font-weight.shtml), [Schriftgröße](http://www.thestyleworks.de/ref/font-size.shtml), [Zeilenhöhe](http://www.thestyleworks.de/ref/line-height.shtml) und [Schriftfamilie](http://www.thestyleworks.de/ref/font-family.shtml). Auch die Verwendung von [Systemfonts](http://www.thestyleworks.de/ref/font.shtml#werte2) kann damit deklariert werden.

Frequenzwerte (frequency values)

Frequenzangaben bestehen stets aus einer numerischen Angabe, an die ohne Zwischenraum der Identifizierer einer Frequenzeinheit angehängt ist. Die Zahlenangabe darf nicht negativ sein, wenn sie 0 ist, kann die Angabe zur Einheit weggelassen werden. Frequenzen treten unter CSS 2 nur nur im Zusammenhang mit auralen Eigenschaften auf. Da aber bis heute kein Browser diese Eigenschaften unterstützt, ist auch die Unterstützung von Frequenzangaben praktisch nicht vorhanden.

Generierte Inhalte (generated contents)

Dies sind Inhalte einer Website, die zwar ausgegeben werden, die aber nicht im Dokumentstammbaum stehen. Sie werden mit Hilfe von CSS generiert. Es gibt zwei Arten von generierten Inhalten: Zum einen lassen sich durch die Eigenschaft content und die Pseudoelemente :before und :after Inhalte vor oder nach Elementen einfügen. Aber Listenbullets und -nummern sind generierter Content.

Geschwisterselektor / Geschwisterelement (adjacent sibling selector)

Zwei Elemente A und B werden dann als Geschwisterelemente bezeichnet, wenn beide vom selben Elternelement abstammen. Wenn außerdem das Element B im Code direkt auf A folgt, dann ist die Bedingung für den Geschwister- Selektor ([E1+E2](http://www.thestyleworks.de/ref/se_adjacent.shtml)) erfüllt.

CSS 2.1 grenzt zusätzlich noch ab, was vorhergehende und nachfolgende Geschwister- Elemente sind: Wenn Element A vor Element B im Dokumentstammbaum auftritt, wird es als vorhergehender Geschwister bezeichnet. Erscheint es später im Dokumentstammbaum, heißt es auch nachfolgender Geschwister. Diese und weitere Erläuterungen zum Konzept der Abstammung in CSS finden sich im Grundlagenartikel [Der Dokumentstammbaum](http://www.thestyleworks.de/basics/doctree.shtml).

Groß- / Kleinschrift (case sensitivity)

In Stylesheets selbst, wie auch in HTML- Dokumenten, spielt die Groß-/Kleinschrift keine Rolle. In XML, und deshalb auch in XHTML, spielt die Groß-/Kleinschrift dagegen schon eine Rolle. Zusätzlich ist die Kleinschrift in XHTML, mit Ausnahme der Attributwerte, eine Bedingung zur Einhaltung der Wohlgeformtheit.

Eine besondere Stellung nehmen hier die ID-Selektoren ein. Sie ist im Artikel [ID-Selektor](http://www.thestyleworks.de/ref/se_id.shtml#meaning) genauer erläutert.

Hintergrundeigenschaften (background properties)

Diese Bezeichnung steht für alle Eigenschaften, die den Hintergrund eines Webdokuments beschreiben: die [Kurzschrift- Eigenschaft](http://www.thestyleworks.de/ref/background.shtml), die [Hintergrundfarbe](http://www.thestyleworks.de/ref/background-color.shtml), die [Hintergrundgrafik](http://www.thestyleworks.de/ref/background-image.shtml) mit [Positionierung/Scrolling](http://www.thestyleworks.de/ref/background-attachment.shtml), [Ursprung](http://www.thestyleworks.de/ref/background-position.shtml) und [Wiederholungsrichtung](http://www.thestyleworks.de/ref/background-repeat.shtml).

Identifizierer (identifier)

Als Identifizierer werden in CSS 2.1 alle Elementnamen, Klassen und IDs bezeichnet, die innerhalb von Selektoren auftreten. Sie dürfen alle Zeichen 0-9, A-Z, a-z und die Sonderzeichen von Nummer 161 an aufwärts nach ISO 10646 enthalten, dazu den Minusstrich und den Unterstrich. Sie dürfen außerdem Escape- Sequenzen und beliebige Zeichen nach ISO 10646 enthalten, wenn sie als numerischer Code auftreten.

Sie dürfen nicht mit einem Minusstrich oder einem Sonderzeichen beginnen.

Inhalte (content)

Die Inhalte eines Webdokuments, die im Quelltext stehen. Hier sind keine generierten Inhalte gemeint.

Inline- Box (inline box)

Die Inline- Boxes werden durch Inline- Level- Elemente generiert. Ihr Inhalt erscheint im Dokumentstammbaum und ist durch CSS-Deklarationen ansprechbar. Die Regeln der Inline- Formatierung in CSS werden im [Artikel über das Line- Box- Modell](http://www.thestyleworks.de/basics/lineboxmodel.shtml) ausführlich dargestellt.

Inline- Level- Element (inline level element)

Alle Elemente, die keine Block- Level- Elemente sind, d. h. die nicht einen [Kontext zur Blockformatierung](http://www.thestyleworks.de/basics/boxmodel.shtml#blockcontext) etablieren, sind Teil des Kontexts zur Inline- Formatierung und werden als Inline- Level- Elemente bezeichnet. Dies sind Elemente wie span, code oder em, deren Inhalt als Textzeilen oder Teile davon abgebildet wird. Die Regeln der Inline- Formatierung in CSS werden im [Artikel über das Line- Box- Modell](http://www.thestyleworks.de/basics/lineboxmodel.shtml) ausführlich dargestellt.

Innere (innewohnende) Dimensionen (intrinsic dimensions)

Diese Bezeichnung wird für alle [replazierten Elemente](http://www.thestyleworks.de/basics/definitions.shtml#replacedelement), und nur für diese, angewendet. Sie bedeutet, dass die Breite und Höhe des Elements durch die Inhalte, die das Element replazieren, festgelegt ist, nicht aber unter Berücksichtigung des Umfeldes ausgerechnet wird.

Innere Kante (inner edge)

Die Darstellung von padding, border und margin ist genaugenommen nicht linienförmig, sondern deckt eine bestimmte Fläche ab. Die Begriffe äußere Kante und innere Kante dienen dazu, die Begrenzungen dieser Flächen wie auch der Contentflächen im Box- Modell exakter zu beschreiben.

ISO 10646

Der internationale Standard, nach dem alle Unicode- Schriftzeichen und Sonderzeichen definiert sind. Eine sehr gute und ausführliche Erklärung zum Unicode- System findet man bei [SelfHTML](http://de.selfhtml.org/inter/unicode.htm).

Kaskade (cascade)

Es kann passieren, dass für dieselbe Element/ Eigenschaft- Kombination unterschiedliche Deklarationen auftreten. Die Kaskade ist das Instrument in CSS, das solche Konfliktfälle löst. Seine genaue Arbeitsweise ist ausführlich im Artikel [Die Kaskade](http://www.thestyleworks.de/basics/cascade.shtml) erklärt.

Kindselektor / Kindelement (child selector)

Ein Element A ist dann das Kindelement des Elementes B, wenn A direkt von B abstammt. Kindelemente lassen sich mit dem Kindselektor ([B>A](http://www.thestyleworks.de/ref/se_child.shtml)) und der Pseudoklasse [:first-child](http://www.thestyleworks.de/ref/pc_structural.shtml#first-child) ansprechen. Diese und weitere Erläuterungen zum Konzept der Abstammung in CSS finden sich im Grundlagenartikel [Der Dokumentstammbaum](http://www.thestyleworks.de/basics/doctree.shtml).

Kombinator (combinator)

Kombinatoren sind die Zeichen: ' ' (whitespace), '>' und '+'. Sie dienen zur Verkettung der verschiedenen Einfach- Selektoren eines Selektors. Mehr über Kombinatoren ist im Artikel [Regelaufbau](http://www.thestyleworks.de/basics/rulesyntax.shtml) erklärt.

Konditioneller Import (conditional import)

Dies ist eine andere Bezeichnung für den medienabhägigen Import von Stylesheets. Im folgenden Beispiel wird ein Stylesheet importiert, das ausschließlich zur Verwendung an TV-Bildschirmen vorgesehen ist.

@import url("tvscreen.css") tv;

Der Vorteil ist hier, dass das Anwenderprogramm keine Ressourcen herunterzuladen braucht, die es nicht unterstützt.

Kontext zur Blockformatierung (block formatting context)

Siehe [Kontext zur Blockformatierung](http://www.thestyleworks.de/basics/boxmodel.shtml#blockcontext) im Grundlagenartikel 'Das Box- Modell'.

Kontext zur Inline- Formatierung (inline formatting context)

Das Prinzip des Kontexts zur Inline- Formatierung definiert, wie die Inline- Level- Elemente nebeneinander bzw. untereinander gestellt werden. Die Regeln der Inline- Formatierung in CSS werden im [Artikel über das Line- Box- Modell](http://www.thestyleworks.de/basics/lineboxmodel.shtml) ausführlich dargestellt.

Längenwert (length value)

Längenwerte werden in einem [besonderen Artikel](http://www.thestyleworks.de/basics/lengthvalues.shtml) ausführlich behandelt.

Leeres Element (empty element)

Ein Element, dem im Dokumentstammbaum keine Inhalte zugeordnet sind, wird 'leeres Element' genannt. Dennoch werden generierte Inhalte, ebenso wie padding, border und margin, an diesem Element angebracht.

Line- Box (line box)

Vereinfacht ausgedrückt, entspricht eine Line- Box einfach einer Zeile Text. Etwas genauer genommen, ist eine Line- Box das Viereck, das eine Zeile Text umgibt. Sie ist in eine oder mehrere Inline- Boxes oder Anonyme Inline- Boxes aufgeteilt. Die Regeln der Inline- Formatierung in CSS werden im [Artikel über das Line- Box- Modell](http://www.thestyleworks.de/basics/lineboxmodel.shtml) ausführlich behandelt.

Line- Box- Modell (line box model)

Mit Hilfe des Line- Box- Modells ist die Darstellung der Textzeilen und anderen Inhalten des Kontext zur Inline- Formatierung in Webdokumenten definiert. Die Regeln der Inline- Formatierung in CSS werden im [Artikel über das Line- Box- Modell](http://www.thestyleworks.de/basics/lineboxmodel.shtml) ausführlich dargestellt.

Listen-Eigenschaften (list properties)

Diese Bezeichnung steht für alle Eigenschaften, die die benannten und unbenannten Listen eines Webdokuments beschreiben: die [Kurzschrift- Eigenschaft](http://www.thestyleworks.de/ref/list-style.shtml), die [Art](http://www.thestyleworks.de/ref/list-style-type.shtml) und [Einrückung](http://www.thestyleworks.de/ref/list-style-position.shtml) des Zählers und die [Grafik](http://www.thestyleworks.de/ref/list-style-image.shtml), die ein Markierungszeichen ersetzt.

margin

Randabstand an Block- Boxes. Dieser Abstand liegt außerhalb des Rahmens einer Block- Box. Er wird innerhalb des Box- Modells definiert. Durch margins werden die Abstände zwischen den Rahmen zweier benachbarter Block- Boxes festgelegt. Weitere Informationen dazu finden sich im Referenzartikel über [margins](http://www.thestyleworks.de/ref/margin.shtml) und im Grundlagenartikel [Das Box- Modell](http://www.thestyleworks.de/basics/boxmodel.shtml).

Medien (media)

Eins der wichtigsten Merkmale von Stylesheets ist die Fähigkeit, zwischen verschiedenen Medien zu unterscheiden. Einige CSS-Eigenschaften sind nur für bestimmte Medien geeignet. Viele Eigenschaften lassen sich aber auch für mehrere Ausgabemedien verwenden. Um die Zuordnung von Medien zu den Eigenschaften zu erleichtern, hat das W3C 10 Medientypen definiert: all, braille, embossed, handheld, print, projection, screen, speech, tty und tv. Medientypen können als Schlüsselworte für medienabhängige Stylesheets angegeben werden. Medientypen werden im Artikel [Medien](http://www.thestyleworks.de/basics/media.shtml) ausführlich erläutert.

Mediengruppen (media groups)

Um die Zuordnung der Eigenschaften zu den Medientypen zu ermöglichen, hat das W3C Mediengruppen eingeführt. In jeder Mediengruppe erfolgt die Unterscheidung der Medientypen nach unterschiedlichen Kriterien. Mediengruppen werden im Artikel [Medien](http://www.thestyleworks.de/basics/media.shtml) ausführlich erläutert.

Musterabgleich (pattern matching)

Der Musterabgleich hat den Zweck, die durch einen Selektor definierten Bedingungen daraufhin zu prüfen, ob sie mit einem Element des Webdokuments zusammenpassen. Diese Bedingungen bestehen aus Einfach- Selektoren und Kombinatoren des Selektors und deren Zusammensetzung. So bedingt z. B. das Pseudoelement :first-child eine Einschränkung auf das erste Kindelement und eine Kombination 'DIV P' gilt nur für Absätze P, die Nachkommenelement eines DIV-Containers sind. Nur auf Elemente des Webdokuments, die diese Bedingungen erfüllen, werden die dazu gehörigen Deklarationen angebracht.

Nächstfolgendes Element (following element)

Es gibt zwei Fälle, in denen ein Element A als dem Element B nachfolgend bezeichnet wird: wenn A ein Nachkommenelement von B ist oder wenn A ein nachfolgender Geschwister von B ist. Diese und weitere Erläuterungen zum Konzept der Abstammung in CSS finden sich im Grundlagenartikel [Der Dokumentstammbaum](http://www.thestyleworks.de/basics/doctree.shtml).

Nachkommenselektor / Nachkommenelement (descendant element)

Alle Elemente B, die direkt, aber in beliebiger Generation, von einem Element A abstammen, bezeichnet man als Nachkommenelemente von A. Dies ist die Umkehrung des Ahnenelementes. Diese und weitere Erläuterungen zum Konzept der Abstammung in CSS finden sich im Grundlagenartikel [Der Dokumentstammbaum](http://www.thestyleworks.de/basics/doctree.shtml).

Normaler Elementfluss (normal flow)

Dies ist eins der drei Positionierungsschemata in CSS. Die Elemente eines Webdokuments sind dann im normalen Elementfluss dargestellt, wenn sie nicht [absolute](http://www.thestyleworks.de/ref/position.shtml#absolute) oder [float-positioniert](http://www.thestyleworks.de/ref/float.shtml) werden. Hierzu gehören auch Elemente, die [relative](http://www.thestyleworks.de/ref/position.shtml#relative) positioniert sind.

padding

Innenliegender Abstand an Block- Boxes. Dieser Abstand liegt zwischen dem Rahmen und den Inhalten einer Block- Box. Er wird innerhalb des Box- Modells definiert. Weitere Informationen dazu finden sich im Referenzartikel über [padding](http://www.thestyleworks.de/ref/padding.shtml) und im Grundlagenartikel [Das Box- Modell](http://www.thestyleworks.de/basics/boxmodel.shtml).

Seiten-Box (page box)

Die CSS-Seitenbox ist eine Erweiterung des Box- Modells. Mit ihrer Hilfe ist es möglich, die Seitenvierecke und Seitenränder für Druckausgaben zu definieren. Weitere Erläuterungen zur Druckausgabe finden Sie auf der Seite über die [@page-Regel](http://www.thestyleworks.de/ref/at_page.shtml).

Seiten-Modell (page model)

Das CSS-Seitenmodell ist eine Erweiterung des Modells zur visuellen Formatierung. Hier sind alle Eigenschaften zur Seitenformatierung, für Seitenumbrüche usw. bei der Druckausgabe zusammengefasst. Weitere Erläuterungen zur Druckausgabe finden Sie auf der Seite über die [@page-Regel](http://www.thestyleworks.de/ref/at_page.shtml).

Positionierungsschema (positioning schema)

Es gibt drei Positionierungsschemata in CSS: absolute Positionierung, float-Positionierung und den normalen Elementfluss.

Prozent (percentage)

Prozentwerte sind immer in Relation zu anderen Werten definiert, wobei die Art dieser Relation genau definiert ist. Auch können Prozentwerte nur für bestimmte Eigenschaften deklariert werden. Welche Eigenschaften dies sind und was sonst noch beim Einsatz von Prozentwerten beachtet werden muss, ist im Artikel [Prozentwerte](http://www.thestyleworks.de/basics/percentage.shtml) erläutert.

Rahmeneigenschaften (border properties)

Die Rahmeneigenschaften dienen dazu, Breite ([border-width](http://www.thestyleworks.de/ref/border-trbl-width.shtml)), Farbe ([border-color](http://www.thestyleworks.de/ref/border-trbl-color.shtml)) und Muster ([border-style](http://www.thestyleworks.de/ref/border-trbl-style.shtml)) der Rahmenfläche einer Box zu spezifizieren. Von Rahmenfläche spricht man deshalb, weil der Rahmen einer Box nicht eine eindimensionale Linie ist, sondern auf jeder Seite der Box eine zweidimensionale Fläche darstellt. Diese Rahmeneigenschaften sind auf alle Elemente anwendbar, für das Element HTML dürfen sie aber durch das Anwenderprogramm ignoriert werden.

Referenzpixel (reference pixel)

Der Referenzpixel dient der Definition der Einheit 'Pixel' in unterschiedlichen Ausgabemedien.

Pixel sind abhängig von der Auflösung des Anzeigegerätes. In den meisten Fällen ist dies ein normaler Bildschirm oder eine Druckausgabe, CSS 2.1 ist aber bereits auf andere Geräte wie Handhelds oder Mobiltelefone abgestimmt.

Damit Pixelmaße auch für andere Medien als den Bildschirm verwendet werden können, definiert CSS 2.1 den 'Referenzpixel' praktisch mit Hilfe eines Winkels. Dabei wird davon ausgegangen, dass ein Pixel am Computerbildschirm etwa die Größe von 0.26 mm hat. Bei einer Bildschirmauflösung von 96 dpi und der 'Leseentfernung' von etwa 70 cm Entfernung (=Armeslänge) wird ein Pixel unter einem bestimmten Winkel wahrgenommen.

Für die etwas kürzere Leseentfernung eines Papierausdrucks wäre ein Pixel, der unter demselben Winkel erscheint, demnach etwa 0.2 mm groß.

Dadurch soll bewirkt werden, dass die Ausgabe auch bei hochauflösenden Laserdruckern in einer annehmbaren Größe erfolgt und nicht etwa ein Pixel mit einem Dot des Printers von 1/300 Inch Größe oder weniger gleichgesetzt wird.

Im umgekehrten Fall gilt dasselbe auch für größere Entfernungen, wie sie z. B. bei Projektionsgeräten auftreten: dort kann ein Pixel auf der Leinwand gut und gerne einige Millimeter groß sein — er wird trotzdem unter demselben Winkel wahrgenommen wie der Pixel von 0.26 mm Größe am Bildschirm.

Regel (rule)

Eine CSS-Regel besteht aus dem Selektor und dem Deklarationsblock. Der Selektor wiederum setzt sich zusammen aus einem oder mehreren Einfach- Selektoren und/oder Kombinatoren. Der Deklarationsblock ist die Zusammenfassung aller Deklarationen der Regel und eine Regel besteht aus Eigenschaft und Wert. Mehr über die Syntax der CSS-Regeln ist im Artikel [Regelaufbau](http://www.thestyleworks.de/basics/rulesyntax.shtml) erklärt.

Relative Positionierung (relative positioning)

Die Position eines Elementes wird zunächst wie im normalen Elementfluss berechnet. Die Verschiebung erfolgt dann von der so bestimmten Stelle aus und ist (anders als bei der Eigenschaft float) nicht von den Dimensionen des umschließenden Elements abhängig. Die Position der benachbarten Elemente wird nicht beeinflusst, sie bleiben in ihrer ursprünglichen Position. Die ursprüngliche Position des verschobenen Elements bleibt frei. Die benachbarten Elemente können allerdings, abhängig vom z-index, durch das verschobene Element überdeckt werden. Ein relativ positioniertes Element bildet den umschließenden Block für nachgeordnete Elemente.

Relativer Längenwert (relative length value)

Relative Längenwerte sind Ems, Exs und Pixel. Prozente werden in diesem Zusammenhang ebenfalls wie relative Werte angesehen. Längenwerte werden in einem [besonderen Artikel](http://www.thestyleworks.de/basics/lengthvalues.shtml) ausführlich behandelt.

Replaziertes Element (replaced element)

Ein Element ist dann 'replaziert', wenn es durch andere Inhalte ersetzt wird und seine Breite und Höhe durch diese anderen Inhalte bestimmt werden. Die Dimensionen werden dann als [innewohnende Dimensionen](http://www.thestyleworks.de/basics/definitions.shtml#intrinsic) bezeichnet. Beispielsweise können in HTML die Elemente IMG und OBJECT replazierte Elemente sein. Der Inhalt des Elementes IMG wird in der Regel durch das im Attribut src="..." beschriebene Bild replaziert.

Schriftart-Eigenschaften (font properties)

Steht zusammenfassend für alle Eigenschaften, mit denen man Schrift formatieren kann. Die Kurzschrift- Eigenschaft [font](http://www.thestyleworks.de/ref/font.shtml) beeinflusst die Einzeleigenschaften für [Textneigung](http://www.thestyleworks.de/ref/font-style.shtml), [Kapitälchen](http://www.thestyleworks.de/ref/font-variant.shtml), [Schriftstärke](http://www.thestyleworks.de/ref/font-weight.shtml), [Schriftgröße](http://www.thestyleworks.de/ref/font-size.shtml), [Zeilenhöhe](http://www.thestyleworks.de/ref/line-height.shtml) und [Schriftfamilie](http://www.thestyleworks.de/ref/font-family.shtml). Auch die Verwendung von [Systemfonts](http://www.thestyleworks.de/ref/font.shtml#werte2) kann damit deklariert werden.

Selektor (selector)

Der Begriff 'Selektor' umfasst in Bezug auf CSS-Regeln alles, was links der ersten geschweiften Klammer steht. So ein Selektor besteht aus einem oder mehreren Einfach- Selektoren, die in einer Reihe hintereinander gestellt und durch Kombinatoren miteinander verbunden sind.

Die innerhalb der geschweiften Klammer festgelegten Deklarationen werden durch den Selektor an den korrespondierenden Elementen im HTML- Dokument, den Subjekten, angebracht. Der Selektor stellt also das Bindeglied zwischen dem HTML- Dokument und den innerhalb der geschweiften Klammer festgelegten Formatierungen dar.

Jeder Selektor passt zunächst zu allen Elementen des Dokuments, zu denen der letzte seiner Einfach- Selektoren passt. Dann wird diese Zielmenge durch die vorangestellten anderen Einfach- Selektoren mehr oder weniger eingeschränkt. Mehr über die Syntax der CSS-Regeln ist im Artikel [Regelaufbau](http://www.thestyleworks.de/basics/rulesyntax.shtml) erklärt.

Spezifizierter Wert (specified value)

Relative Werte werden als Verhältnismaße zu anderen Größen definiert. Deshalb muss aus dem in einer CSS-Regel deklarierten Wert ein am Bildschirm darstellbarer Wert bestimmt werden. Dazu sind die folgenden vier Berechnungsschritte erforderlich:

Spezifizierter Wert → Berechneter Wert → Verwendeter Wert → Aktueller Wert.

Der Ablauf dieser Berechnung ist im Abschnitt [Relative Längenwerte](http://www.thestyleworks.de/basics/lengthvalues.shtml#relative) detailliert erläutert.

Subjekte (subjects)

Die Elemente des Webdokuments, die zu einem Selektor passen, werden Subjekte des Selektors genannt. Mehr über die Syntax der CSS-Regeln ist im Artikel [Regelaufbau](http://www.thestyleworks.de/basics/rulesyntax.shtml) erklärt.

Typselektor (type selector)

Der Typselektor ist einer der beiden in CSS möglichen Einfach- Selektoren. Ausführlicheres über Typselektoren ist in den Artikeln [Typselektor](http://www.thestyleworks.de/ref/se_type.shtml) und [CSS-Regeln](http://www.thestyleworks.de/basics/rulesyntax.shtml) erläutert.

Umschließender Block (containing block)

Oftmals werden in CSS Positionen und Größen relativ zu den Kanten des umschließenden Blocks definiert. Was man sich unter diesem Block vorzustellen hat und wie er definiert wird, ist im Beitrag über das [Box- Modell](http://www.thestyleworks.de/basics/boxmodel.shtml#containing) ausführlich erläutert.

Unicode (unicode)

Internationaler Standard, nach dem die Schriftzeichen aller Sprachen in einem Codesystem verschlüsselt sind. Die Codetabellen werden vom [Unicode- Konsortium](http://www.unicode.org/) unterhalten. Die aktuelle Version 4.0 wurde von der International Standards Organisation zur Norm [ISO 10646](http://www.thestyleworks.de/basics/definitions.shtml#iso10646) erhoben. Ausführlich deutschsprachige Informationen findet man bei [SelfHTML](http://de.selfhtml.org/inter/unicode.htm)

Universalselektor (universal selector)

Der Universalselektor, dargestellt als \*, ist ein Einfach- Selektor, der auf alle Elemente zutrifft. Falls der Einfach- Selektor aus weiteren Komponenten als dem \* besteht, dann kann der \* auch weggelassen werden.

URI, URL, URN (URI, URL, URN)

Die URL (Uniform Resource Locator) bezeichnet die Addresse einer Internet-Resource, sie ist also praktisch nichts anderes als die Location einer Website. Die URN (Uniform Resource Name) hat eine etwas weiter gefasste Bedeutung, bezieht sich aber, solange es um HTML geht, auf die gleiche Art der Webaddressierung wie die URL. Beide werden unter der Bezeichnung URI (Uniform Resource Identifier) zusammen gefasst. Eine URI lässt sich auch in CSS als Wert verwenden — dies ist im Grundlagenartikel über [CSS-Werte](http://www.thestyleworks.de/basics/values.shtml#uri) erläutert.

Validität, Validierung (validity, validation)

Ein Stylesheet ist dann validiert, wenn sein kompletter Inhalt den Vorgaben des CSS- Standards entspricht. Man kann ein Stylesheet nach CSS 1 validieren, in dem Fall braucht es nur den Forderungen in CSS 1 zu genügen. Ein nach CSS 2 validiertes Stylesheet muss den Forderungen nach CSS 2 bzw. CSS 2.1 entsprechen. Da CSS 2 eine Erweiterung von CSS 1 ist, sind für CSS 1 validierte Stylesheets immer auch für CSS 2 valide.

Das bedeutet, dass ein valides Stylesheet

1. entsprechend der CSS-Grammatik (Punktuation) geschrieben sein muss,
2. nur in der Norm definierte (keine proprietären) At-Regeln, Eigenschaften und Werte enthalten darf,
3. nur zusammen gehörende und richtig geschriebene Eigenschaften und Werte beinhalten kann.

Verwendeter Wert (used value)

Relative Werte werden als Verhältnismaße zu anderen Größen definiert. Deshalb muss aus dem in einer CSS-Regel deklarierten Wert ein am Bildschirm darstellbarer Wert bestimmt werden. Dazu sind die folgenden vier Berechnungsschritte erforderlich:

Spezifizierter Wert → Berechneter Wert → Verwendeter Wert → Aktueller Wert.

Der Ablauf dieser Berechnung ist im Abschnitt [Relative Längenwerte](http://www.thestyleworks.de/basics/lengthvalues.shtml#relative) detailliert erläutert.

Vorfahrenselektor / Vorfahrenelement (ancestor)

Wenn sich die Abstammungslinie von einem Element A nach unten direkt zu einem Element B verfolgen lässt, dann wird A als Vorfahrenelement von B bezeichnet. Dabei kann B das direkte Kindelement von A sein, es können aber auch beliebig viele Generationen dazwischen liegen. Diese und weitere Erläuterungen zum Konzept der Abstammung in CSS finden sich im Grundlagenartikel [Der Dokumentstammbaum](http://www.thestyleworks.de/basics/doctree.shtml).

Vorhergehendes Element (preceding element)

Es gibt zwei Fälle, in denen ein Element A als dem Element B vorhergehend bezeichnet wird: wenn A ein Vorfahrenelement von B ist oder wenn A ein vorhergehender Geschwister von B ist. Diese und weitere Erläuterungen zum Konzept der Abstammung in CSS finden sich im Grundlagenartikel [Der Dokumentstammbaum](http://www.thestyleworks.de/basics/doctree.shtml).

Wert (value)

Ein oder mehrere Werte, wie sie nach CSS 2.1 für die entsprechende Eigenschaft zulässig sind. Hinter dem letzten Wert für eine Eigenschaft steht ein Semikolon (;) zur Trennung von der nächsten Deklaration. Mehr über die Werte innerhalb der CSS-Regeln ist im Artikel [Regelaufbau](http://www.thestyleworks.de/basics/rulesyntax.shtml) erklärt.

Winkelwert (angle value)

Winkelangaben bestehen stets aus einer numerischen Angabe, an die ohne Zwischenraum der Identifizierer einer Winkelmaßeinheit angehängt ist. Eine negative Zahlenangabe wird in ihr positives Äquivalent umgeformt, z. B. wird der Wert -45deg nach 315deg umgerechnet. Wenn sie 0 ist, kann die Angabe zur Einheit weggelassen werden. Winkelangaben treten unter CSS 2 nur nur im Zusammenhang mit auralen Eigenschaften auf. Da aber bis heute kein Browser diese Eigenschaften unterstützt, ist auch die Unterstützung von Winkelmaßen praktisch gleich null.

Zurückschrumpfung (shrink-to-fit)

Anders als bei Block- Level- Elementen oder Inline- Level- Elementen richtet sich die Breite von z. B. Tabellenzellen oder floatierten Elementen nicht nach den äußeren Gegebenheiten, sondern nach der Breite des Inhalts, genau genommen nach vorgegebenen und möglichen Zeilenumbrüchen, den Werten der Eigenschaften des Boxmodells und der Breite evtl. vorhandener Scrollbalken.

Zusammenfallende Außenabstände (collapsing margins)

Unter bestimmten Umständen können die oberen bzw. unteren margins zweier untereinander liegender Elemente zusammenfallen. Im Referenzartikel 'margin' sind die exakten Abläufe dieses Zusammenfallens im Unterthema '[Zusammenfallende Abstände](http://www.thestyleworks.de/ref/margin.shtml#collaps)' erläutert, darüber hinaus gibt es auch ein [praktisches Beispiel](http://www.thestyleworks.de/basics/margins.shtml).