2018

Karl Steinam

Klara-Oppenheimer-Schule

28.11.2018

HTML und CSS



Inhalt

[Historie von HTML und css 4](#_Toc728448)

[Zusammenhang HTML / Webserver 4](#_Toc728449)

[5](#_Toc728450)

[Eigenschaften 6](#_Toc728451)

[Elemente 7](#_Toc728452)

[7](#_Toc728453)

[HTML-Tags 7](#_Toc728454)

[HTML-Attribute 9](#_Toc728455)

[Das HTML-Grundgerüst 10](#_Toc728456)

[Die Grundlagen von CSS 15](#_Toc728457)

[Aufbau von CSS-Regeln 17](#_Toc728458)

[Einbinden von CSS 18](#_Toc728459)

[CSS und Selektoren 20](#_Toc728460)

[TypSelektoren 20](#_Toc728461)

[Klassenselektor 20](#_Toc728462)

[Übung: 21](#_Toc728463)

[ID-Selektor 22](#_Toc728464)

[Kontextselektor 23](#_Toc728465)

[Lösungen 29](#_Toc728466)

[Verdeutlichung am Strukturbaum 29](#_Toc728467)

[Dein bester Freund DIV 32](#_Toc728468)

[Das BOX-Modell von CSS 33](#_Toc728469)

[Gesamthöhe und Gesamtbreite 34](#_Toc728470)

[Aufgabe 34](#_Toc728471)

[box-sizing: border-box 35](#_Toc728472)

[Aufgabe 36](#_Toc728473)

[DISPLAY-Eigenschaft 39](#_Toc728474)

[inline: Inline-Box(en) 39](#_Toc728475)

[block: Block-Box 39](#_Toc728476)

[inline-block: Außen Inline-Box, innen Block-Box 39](#_Toc728477)

[Aufgabe 41](#_Toc728478)

[Float und Clear 42](#_Toc728479)

[Einsatzgebiet 43](#_Toc728480)

[# 43](#_Toc728481)

[Clearing the Float 44](#_Toc728482)

[#The Great Collapse 45](#_Toc728483)

[#Video 45](#_Toc728484)

[Aufgabe 46](#_Toc728485)

[Größenangaben 51](#_Toc728486)

[Arten von Farben 52](#_Toc728487)

[Schriftarten 53](#_Toc728488)

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.

## 

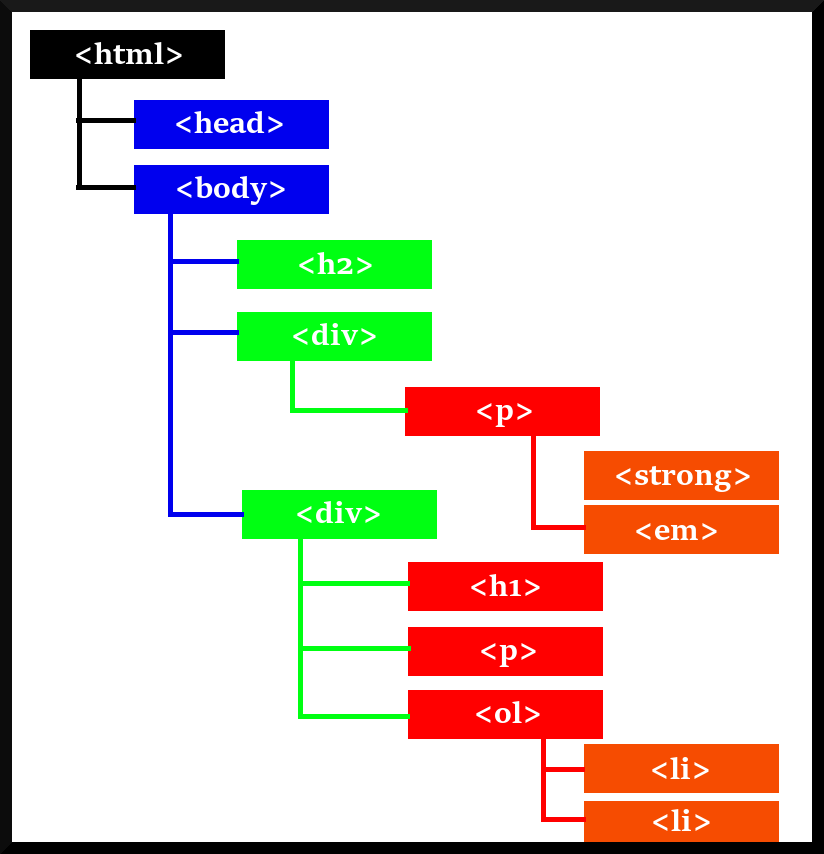
## Eigenschaften

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)

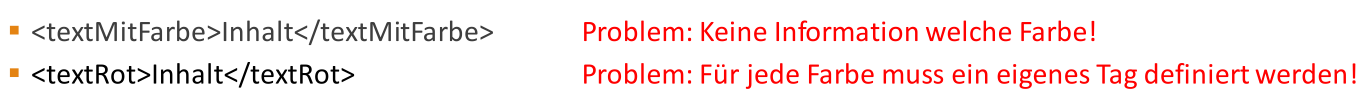
### 

### 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

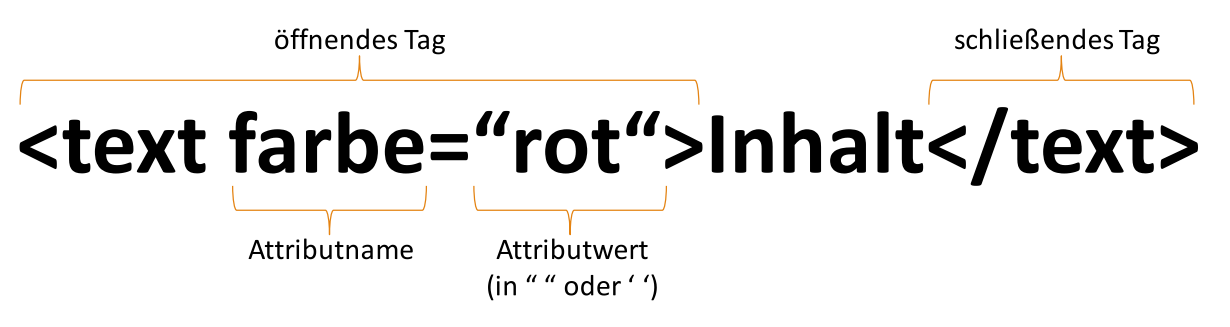
Im konkreten Fall einer Farbe wäre es schwierig, für jedes Element verschiedene Versionen des Tags vorzuhalten.



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>

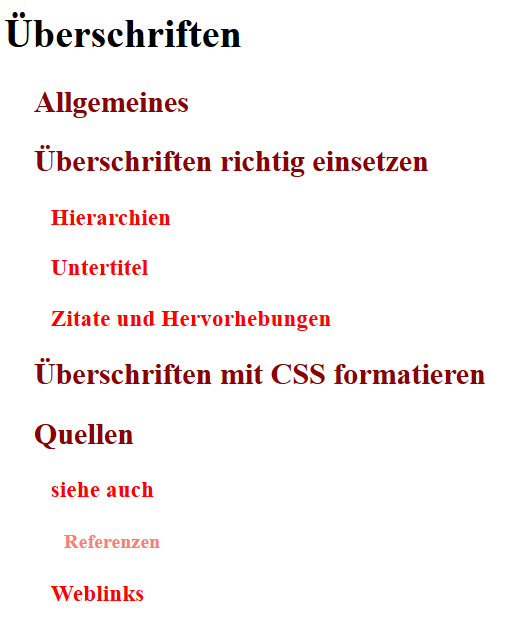
</body>

</html>

Übungsaufgabe:

A1.

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



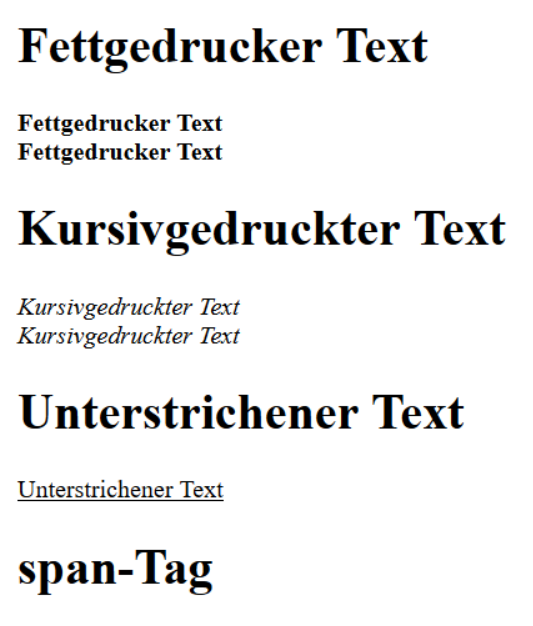
Erweiterung:

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

A2:

Erzeugen sie eine HTML-Datei mit folgendem Aussehen

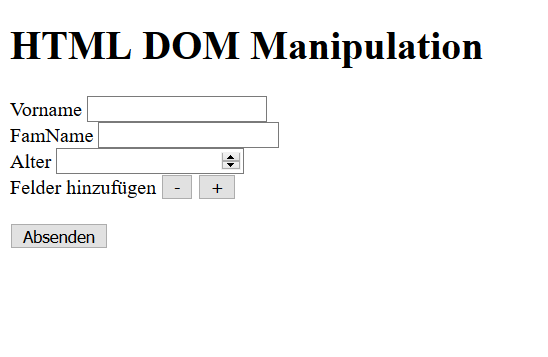
Hilfe gibt es unter https://wiki.selfhtml.org/wiki/HTML/Textauszeichnung



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>

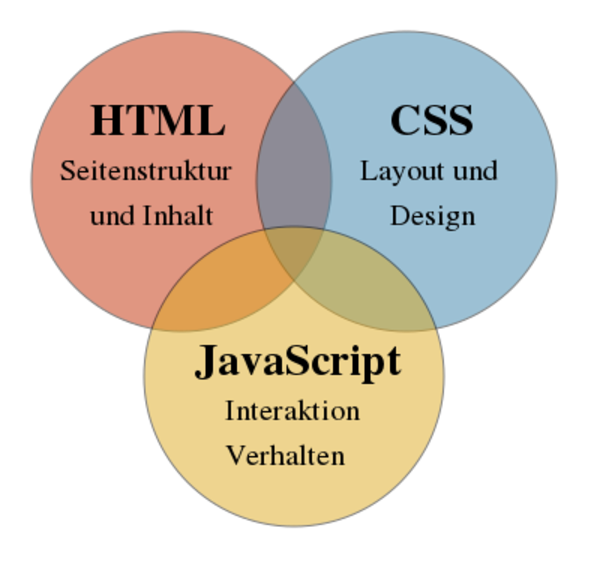
</html>

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.

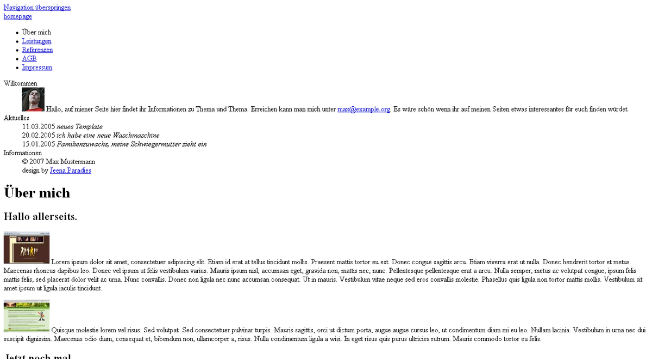
Es ist aber beinahe unmöglich, einen so über die Zeit gewucherten Internetauftritt umzugestalten - man müsste dutzende oder mehr Einzeldateien umschreiben und im Gewusel der Klassen und Elemente wird man sich schnell verlieren. Letztlich dauert die Änderung beinahe länger als die Neuerstellung. Und genau 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.“**

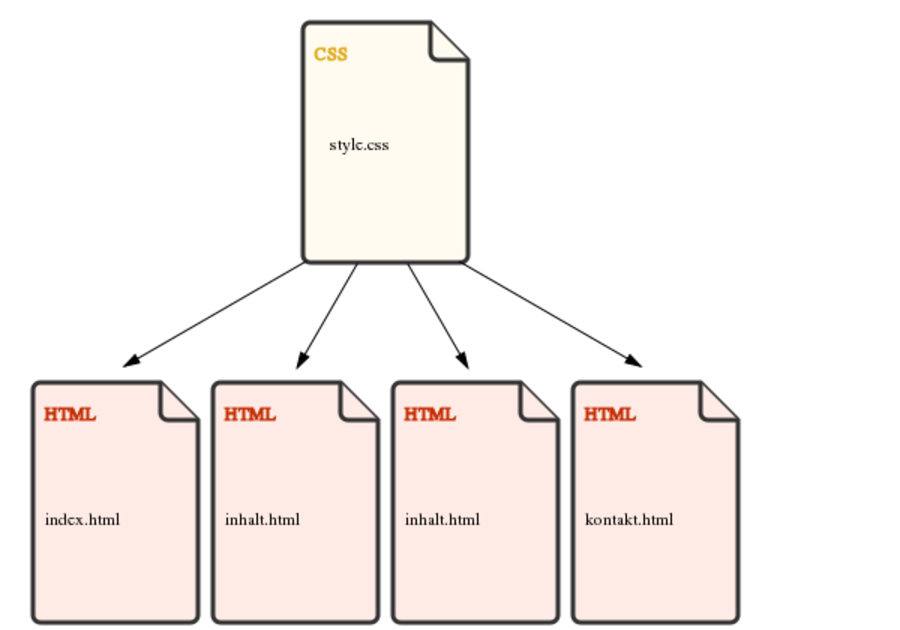
Soll heißen - man gestaltet die Seite erst komplett ohne CSS, nur die Seitenstrukturierung in Kapitel und Absätze, bei diesen dann die Textauszeichnung von Überschriften und Ähnlichem. Es stellt auch die „allgemeine Gültigkeit“ des Layouts sicher - ist die Grundlage an sich stimmig, kann man die Seite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

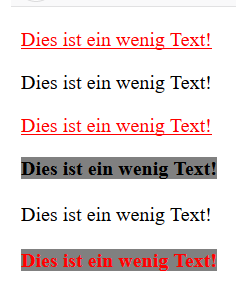
### Übung:

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:

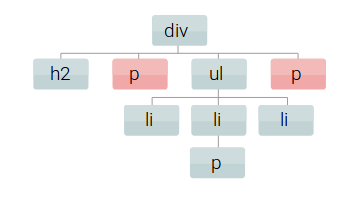
<!doctype html>  
<html>  
 <head>  
 <meta charset="utf-8">  
 <title>CSS-Beispiel: IDSelektor</title>  
 <style>  
   
 your code here.......  
   
 </style>  
 </head>  
 <body>  
 <div id="beispiel">  
 <h1>ID-Selektoren</h1>  
 <p>Dieses Beispiel demonstriert die Wirkung des ID-Selektors.</p>  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

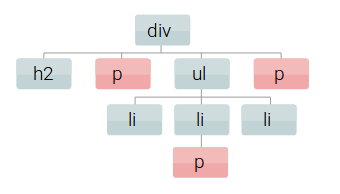
* Ein Nachfahren-Selektor (**Descendant Selector)** filtert Elemente unterhalb bestimmter Elemente.
* Ein Nachbar-Selektor (**Adjacent Selector)** filtert aufeindnerfolgende Elemente der gleichen Ebene,

#### Kindselektor

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

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

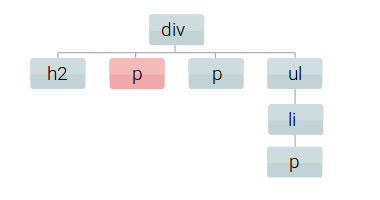
#### Nachfahrenselektor

* Leerzeichen " " zwischen den Selektoren

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

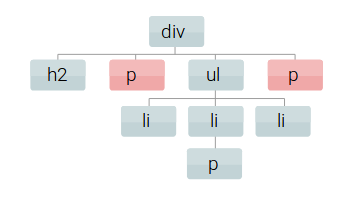
div p { color: black; background-color: red; }

#### 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>

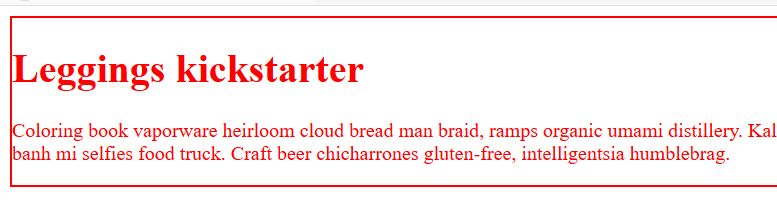
Craft beer chicharrones gluten-free, intelligentsia humblebrag.

</p>

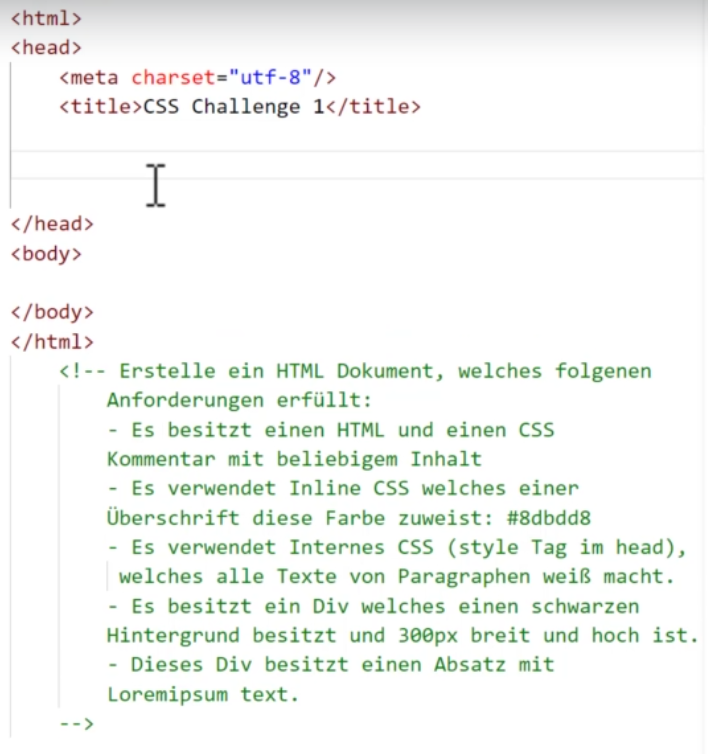
</body>

</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.

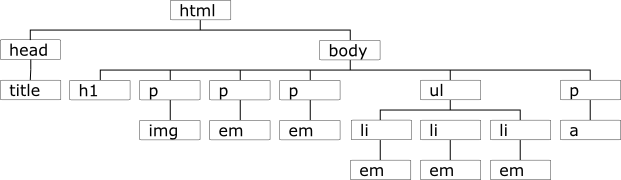
body

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

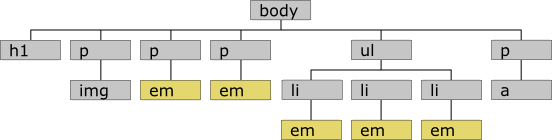
{

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

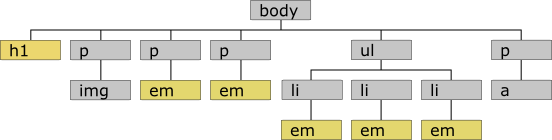
{

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).

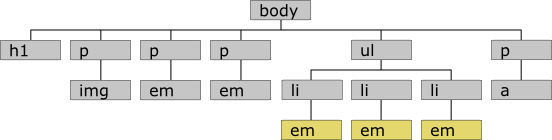
ul em

{

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;

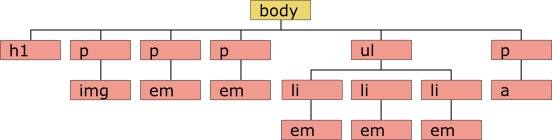
color: black;

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

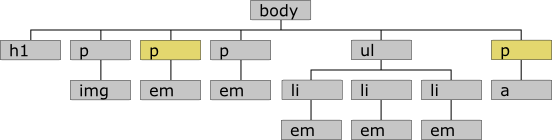
{

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

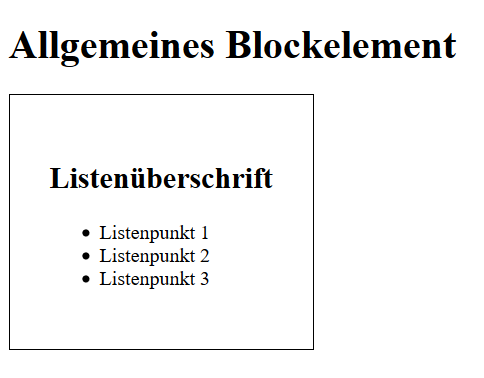
Alles recht kompliziert, oder? Es gibt noch weitere Möglichkeiten, bestimmte Elementgruppen auszuwählen. Das kannst du dann in einem Nachschlagewerk wie [SELFHTML](https://www.inf-schule.de/information/informationsdarstellunginternet/hinweise) nachlesen.

# 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>

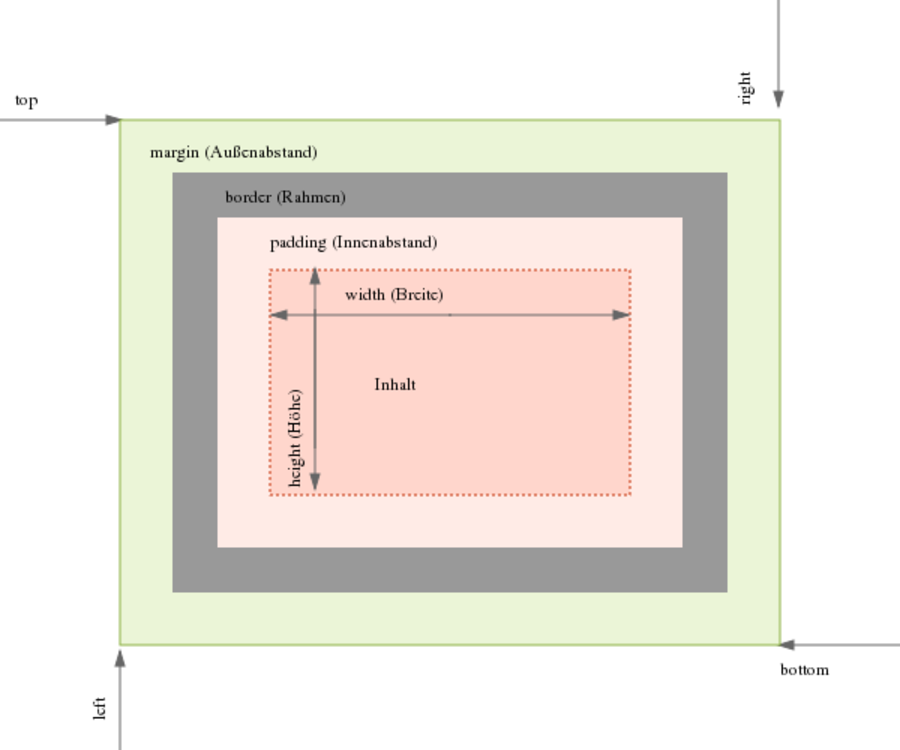
# Das BOX-Modell von CSS

<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:

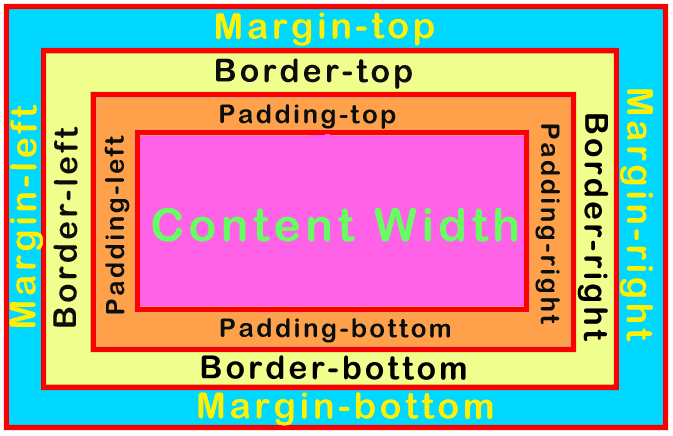
- Dem Inhaltsbereich, also der Fläche, die durch Texte und Bilder oder Eigenschaften wie [width](https://wiki.selfhtml.org/wiki/CSS/Eigenschaften/Gr%C3%B6%C3%9Fenangaben/width) und [height](https://wiki.selfhtml.org/wiki/CSS/Eigenschaften/Gr%C3%B6%C3%9Fenangaben/height) vorgegeben wird,  
- einem Innenabstand ([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:

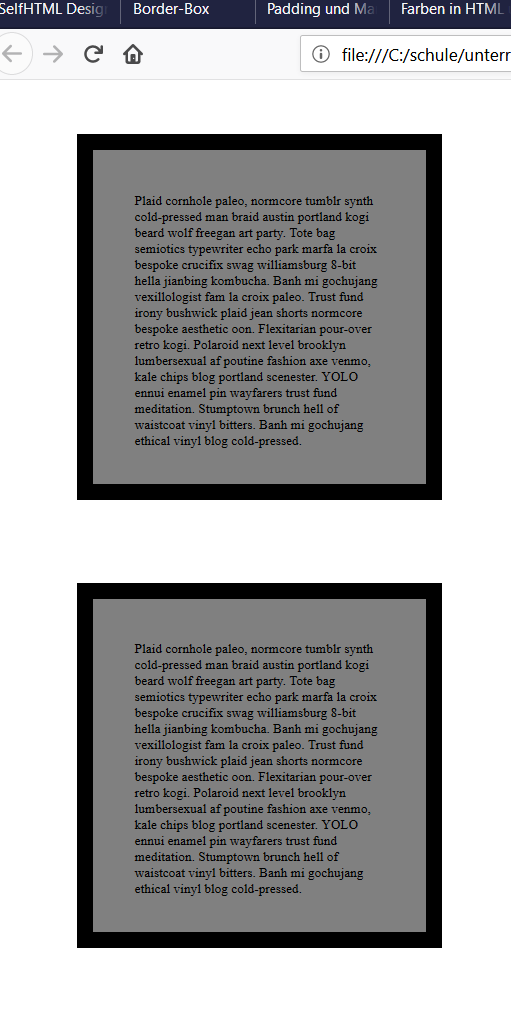
margin-left + border-left + padding-left + width + padding-right + border-right + margin-right

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.

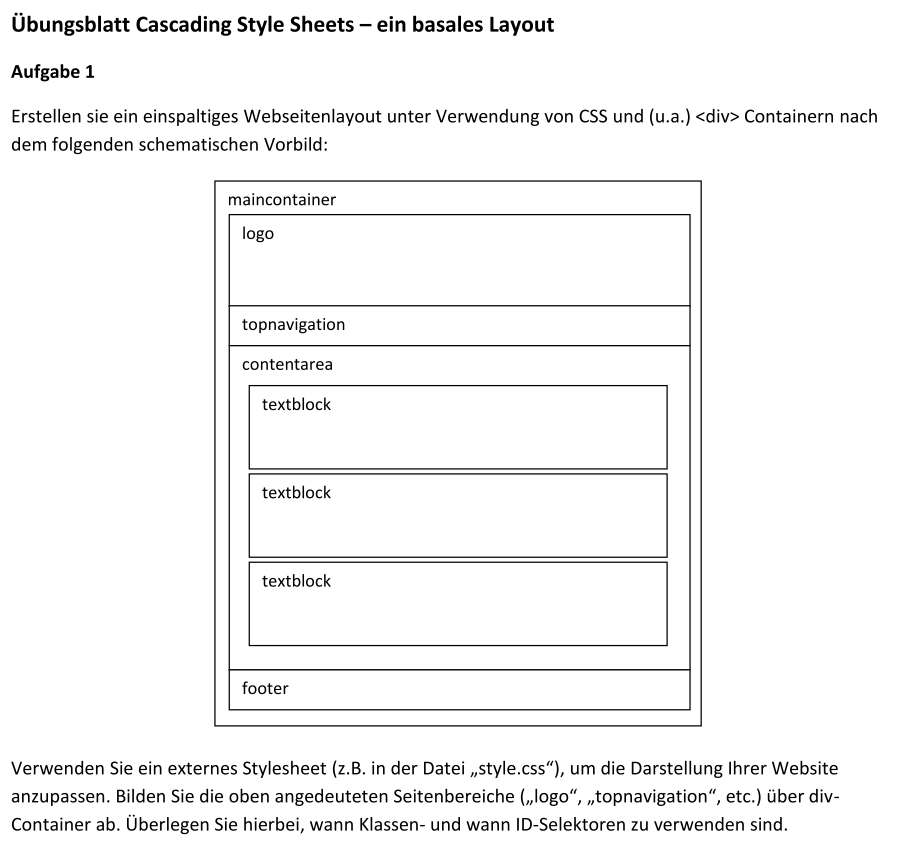
Zudem interpretierten alte Internetexplorer das Box-Modell falsch, sodass Webentwickler auf diese besondere Rücksicht nehmen mussten. Ebenso war es schwierig bis unmöglich, ein responsives Layout zu erstellen, welches etwa prozentuale Breitenangaben mit pixelgenauen Angaben für die Rahmenbreite kombinierte.

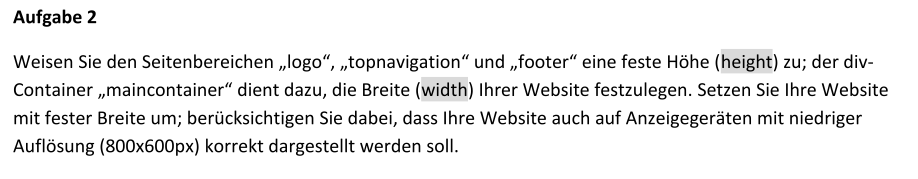
Mit der Eigenschaft box-sizing können Sie bestimmen, worauf sich Angaben zu den Abmessungen eines Elementes beziehen.

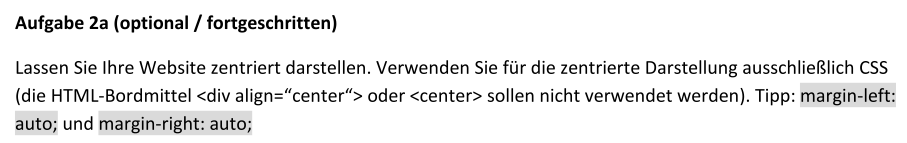
Erlaubt sind dabei eine der folgenden Angaben.

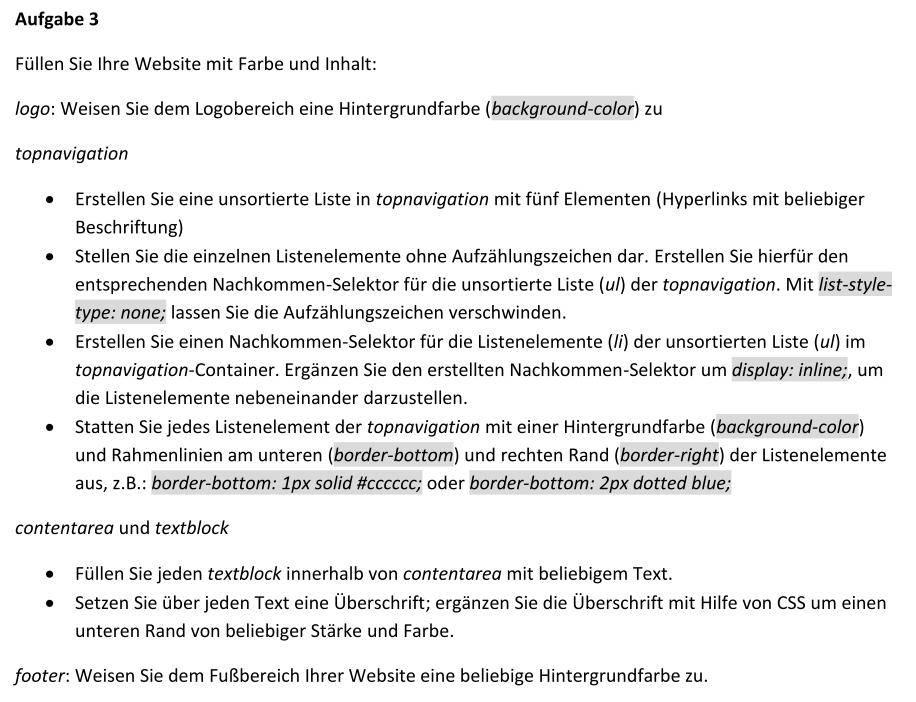
* content-box, Standardwert, Angabe gilt nur für den Inhalt
* border-box, Angabe gilt für Inhalt, Innenabstand und Rahmen, Standard für IE im Quirksmodus
* inherit, box-sizing des Elternelements

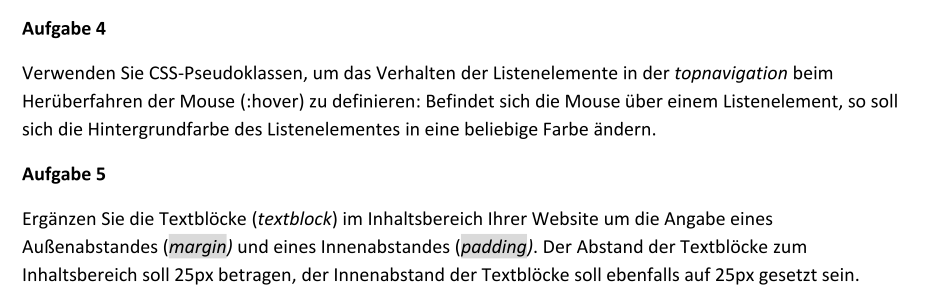
## Aufgabe

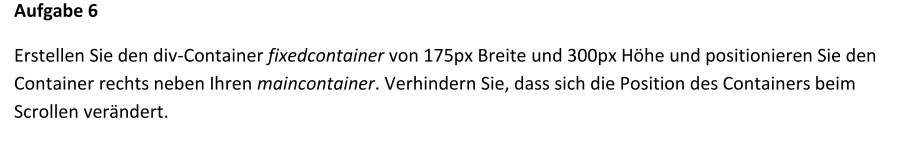












Inline- vs- Block-Elemente  
  
Die Breite von **Inline-Boxen** wird allein durch den Inhalt bestimmt. Die width-Eigenschaft ist wirkungslos. Die Teile einer aufgeteilten Inline-Box nehmen nur die Breite ein, die der Inhalt auch tatsächlich benötigt.

**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

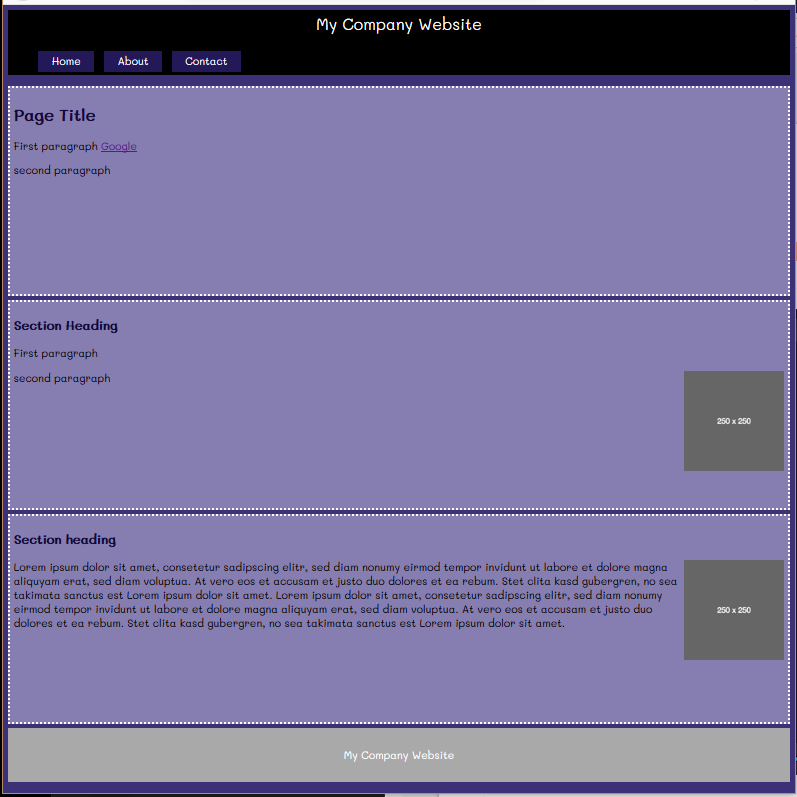
.test1 {  
 display: block;  
 background-color: #F00;  
 width: 300px;  
 height: 300px;  
 margin-bottom: 20px;  
}

.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

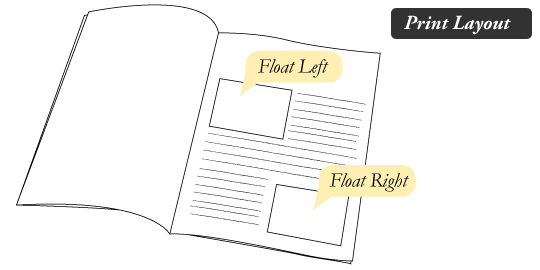
Erstellen Sie folgendes Aussehen für eine Webseite. Die entsprechenden Ausgangsdateien finden Sie im Ordner Exercise 3. Folgen Sie den Anweisungen in der Datei exercise\_students.css und binden Sie diese in ihr HTML-Dokument mit ein.

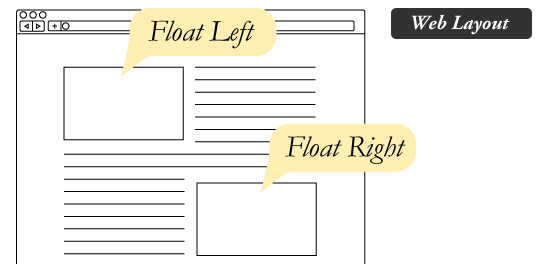


Lösung: exercise3.css

# 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 {

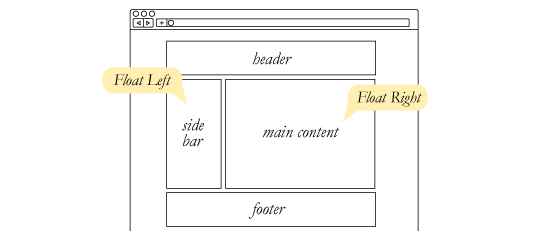
float: right;

}

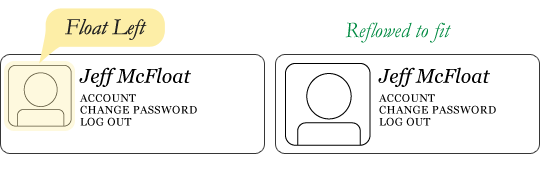
Es gibt vier gültige Werte für die float-Eigenschaft: Left und Right lassen Elemente in die jeweilige Richtung fließen. None (default verhindert den Fluss und Inherit übernimmt den Wert vom übergeordneten Element.

### Einsatzgebiet

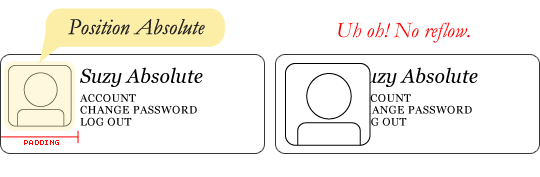
Neben dem oben angesprochenen Beispiel des Umfließens von Bildern können floats zum Erstellen eines ganzen Weblayouts verwendet werden.

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.

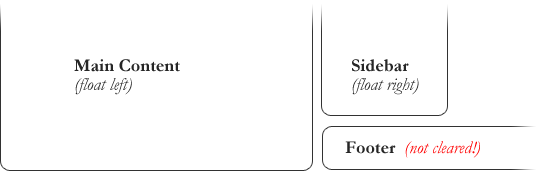


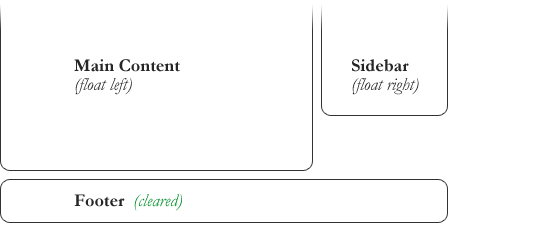
Dasselbe Layout könnte auch mit einer relativen Positionierung für den Container und einer absoluten Positionierung für den Avatar erreicht werden, Der Text wäre aber jetzt vom Avatar nicht abhängig und würde sich bei einer Größenänderung nicht neu ausrichten.



### 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.



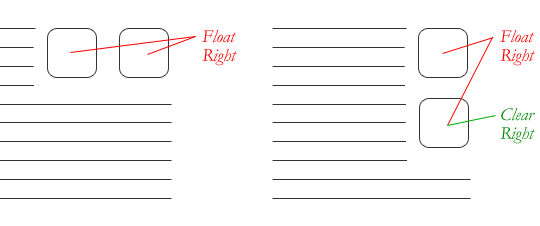
Im obigen Beispiel ist die Sidebar rechts vom MainContent und etwas kleiner. Der footer muss nun diese Lücke aufnehmen und das obige Layout ist die Folge. Um dieses Problem zu lösen, kann der Footer mit einer clear-Eigenschaft versehen werden, um seine unten ausgerichtete Anordnung zu garantieren.

#footer {

clear: both;

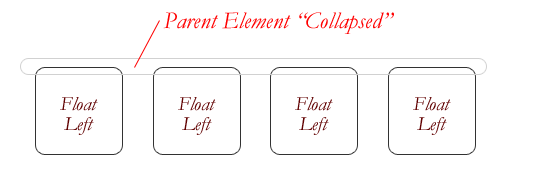
}

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

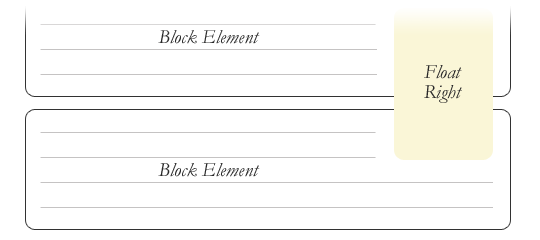


### The Great Collapse

One of the more bewildering things about working with floats is how they can affect the element that contains them (their "parent" element). If this parent element contained nothing but floated elements, the height of it would literally collapse to nothing. This isn't always obvious if the parent doesn't contain any visually noticeable background, but it is important to be aware of.



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;

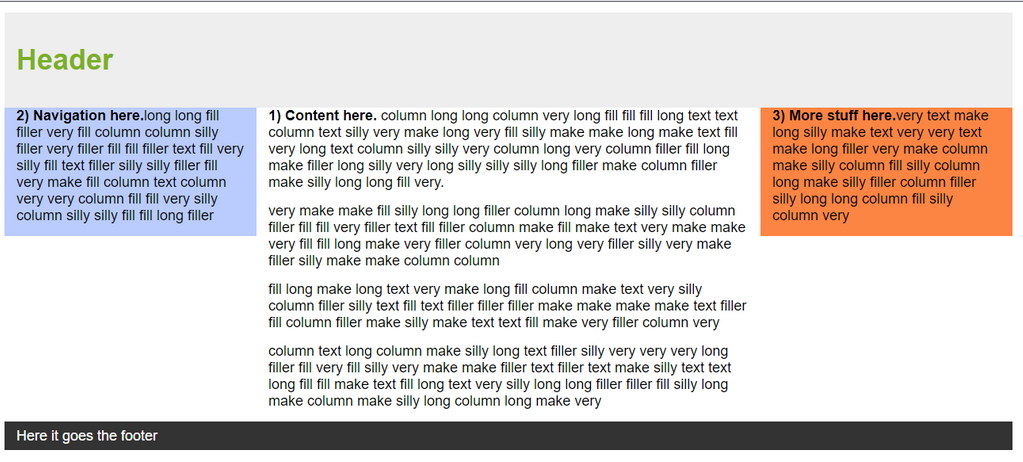
}

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,

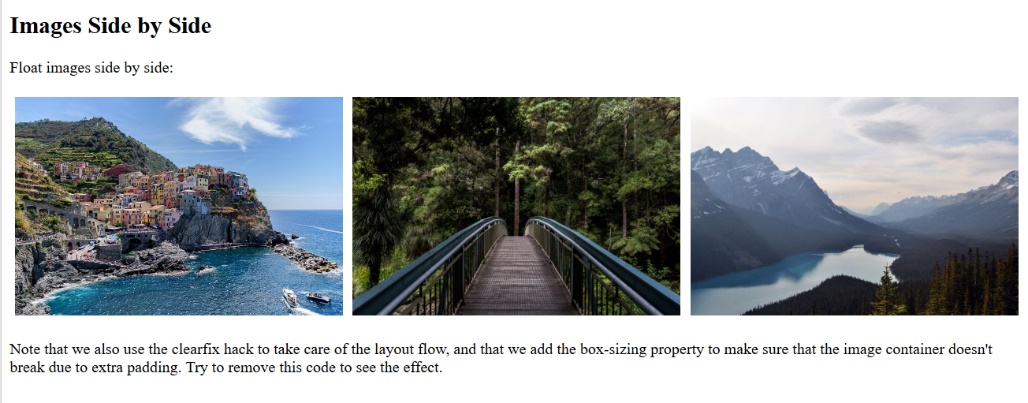
Informieren Sie sich dazu über die float und clear – Eigenschaft von css



Layout – float und clear

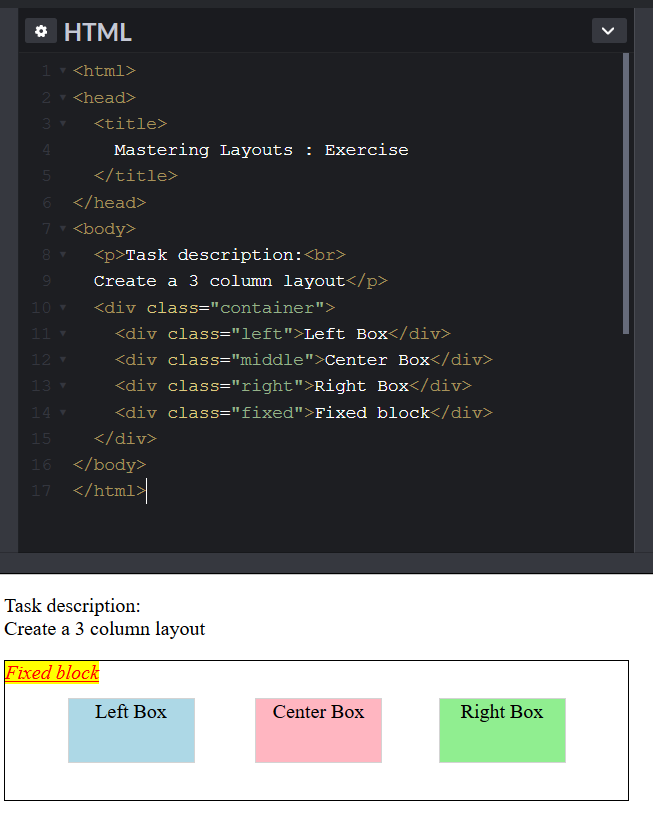
<https://www.w3schools.com/css/tryit.asp?filename=trycss_float5>

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

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

Aufgabe:

Erzeugen Sie folgendes Aussehen untenstehender CSS-Datei.



# Größenangaben

# Arten von Farben

# Schriftarten