Webdesign mit HTML5 und CSS3

W.Rose

Inhalt

Kapitel 1 - Vorwort *3*
Zielgebung: Aufbau einer eignen Webseite 3
Quo vadis? Viele Wege führen zum Ziel 4
Theorie: Was benötigt man für Webdesign? 4
Praxis: Wo bekomme ich Hilfe und eine Testumgebung? 5
Kapitel 2 - HTML-Grundlagen *6*
Theorie: Das HTML-Grundgerüst 6
Praxis: Elemente, Attribute, Werte und URLs 7
Kapitel 3 - CSS-Gestaltung *8*
Theorie: CSS 8
CSS im <Style> Element 9
CSS in externer Datei 9
Typ- und Klassenselektoren 10
..... 11
..... 11
Weitere HTML-Elemente 12
Theorie: Listen 12
Übung: Listen 13
Weiterführung: Bilder und Textumlauf 13
Übung: Textumlauf 13
Theorie: Tabellen 15
Übung: Tabellen 16
Theorie: Absolute Positionierung 17
Übung: Absolute Positionierung 19
Zusatz: CSS3 19
Kapitel 4 - Aufbau von Webseiten *20*
Theorie: Semantisches HTML 20
Theorie: Navigation 20
Theorie: Box-Modell 20
Übung: Einspaltiges Layout 21
Übung: Zweispaltiges Layout 22
Übung: Dreispaltiges Layout 23
Übung: Menü mit Buttons 24
Theorie: Webseite mit mehreren Seiten 25
Kapitel 5 - Projekt *26*
Kapitel 6 - Überleitung zu JavaScript: Eine Bildergallerie mit HTML und CSS ? ..*27*

Kapitel 1 - Vorwort

Zielgebung: Aufbau einer eignen Webseite

Unser langfristiges Ziel ist der Aufbau einer Webseite zu einem Thema unserer Wahl: Denkt an Themen wie Nachhaltigkeit und Klimaschutz, Medien und Technik, Hobbies und Spiele. Für den Aufbau der Seiten setzen wir uns realistische Ziele:

- Eine Startseite mit sauber formatiertem Texten und Bildern (frontpage)
- Ein Menü mit Links zu untergeordneten Seiten (dropdown menu)
- Ein Quelltext der übersichtlich und nachvollziehbar ist. (externes CSS stylesheet)
- Ein Design, dass auf unterschiedlichen Bildschirmen funktioniert (responsive grid design)

Später ergänzen wir dynamische Inhalte (Bildgalerien, Quiz, Gästebuch) mit Plugins oder sogar mit einem eigenen Programm in Javascript und PHP. Wozu die vielen Sprachen? Das ist der ganz normale Alltag im Leben eines Informatikers. Über die 3-4 Sprachen im Webdesign kann er nur lachen:

Mit **HTML** erzeugt man die Struktur einer Webseite. („DOM“-Grundgerüst und Inhalt)

Mit **CSS** gestaltet man ihr Aussehen. (Buttons, Container, Dropdown-Menus)

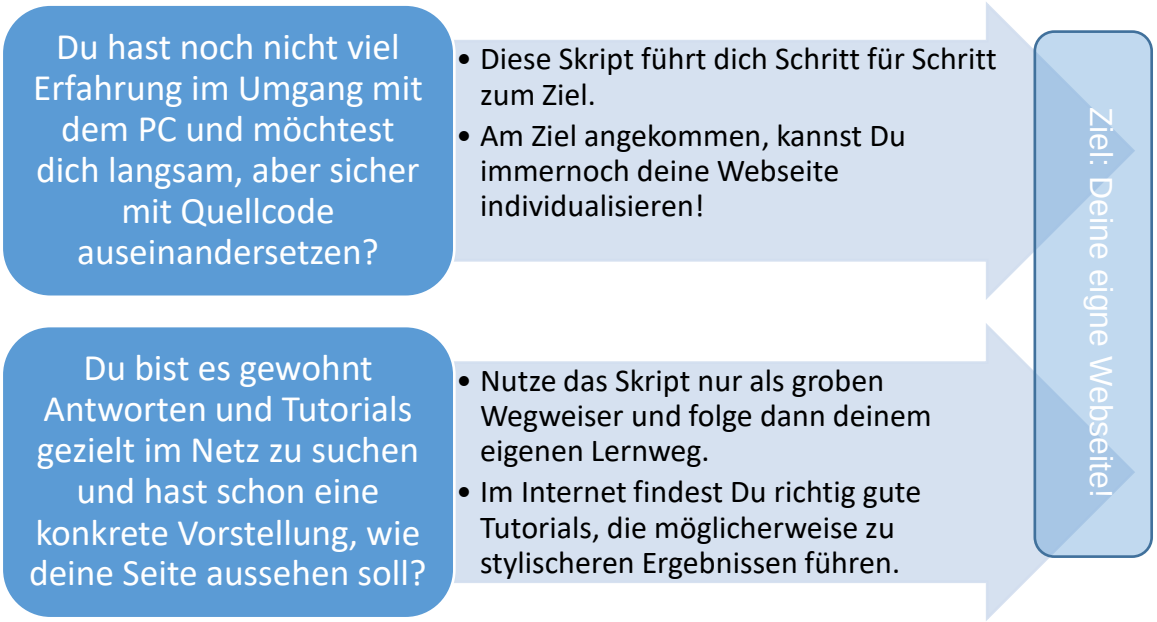
Mit **JavaScript** baut man kleine Programme und Kontrollabfragen ein. (Slideshow, Quiz)

Mit **PHP / MySQL** kann man auf Datenbanken zugreifen. (Gästebuch, Datenbank)

Letzteres funktioniert übrigens nur auf einem Server oder in einem Serveremulator – einer Umgebung die so tut als wäre sie ein Webserver (XAMPP).

Quo vadis? Viele Wege führen zum Ziel

***Paint by numbers* oder doch lieber *learning by doing*?**



Je nach Motivation und Vorkenntnissen gibt es viele Wege, die zum Ziel führen:

Bedenke: Der erste Weg bietet weniger Raum für die freie, kreative Entfaltung deiner Ideen. ***„paint by numbers“*** führt schnell zum Ergebnis, aber selber gestalten macht mehr Spaß – vorausgesetzt man kann halbwegs zeichnen.

Umgekehrt kann ***„learning by doing“*** auch sehr frustrierend sein, wenn man auf halben Weg feststellt, dass viele Versuche scheitern und nicht alles auf Anhieb klappt. Beschreitest Du den 2. Weg braucht es ein hohes Maß an Motivation und Durchhaltevermögen!

Theorie: Was benötigt man für Webdesign?

Es genügt ein einfacher Texteditor. Eine HTML-Datei (Hypertext Markup Language) ist im Prinzip ein Textdokument mit Querverweisen zu Bildern und anderen Medien. Mittels einiger unsichtbarer Befehle bzw. Tags wird das Dokument formatiert. Wenn ihr also Wörter markiert und das Format Überschrift wählt, setzt das Programm Word oder Writer das Tag <h1> bzw. </h1> vor und hinter die Markierung und speichert diese Markierung separat in XML-Dateien.

Wechselt man die Endung eines Word-Dokuments von *.docx zu *.zip und öffnet die Zip-Datei stellt man fest, dass ein Word-Dokument tatsächlich viele weitere Dateien und sogar Unterordner enthält.

Praxis: Wo bekomme ich Hilfe und eine Testumgebung?

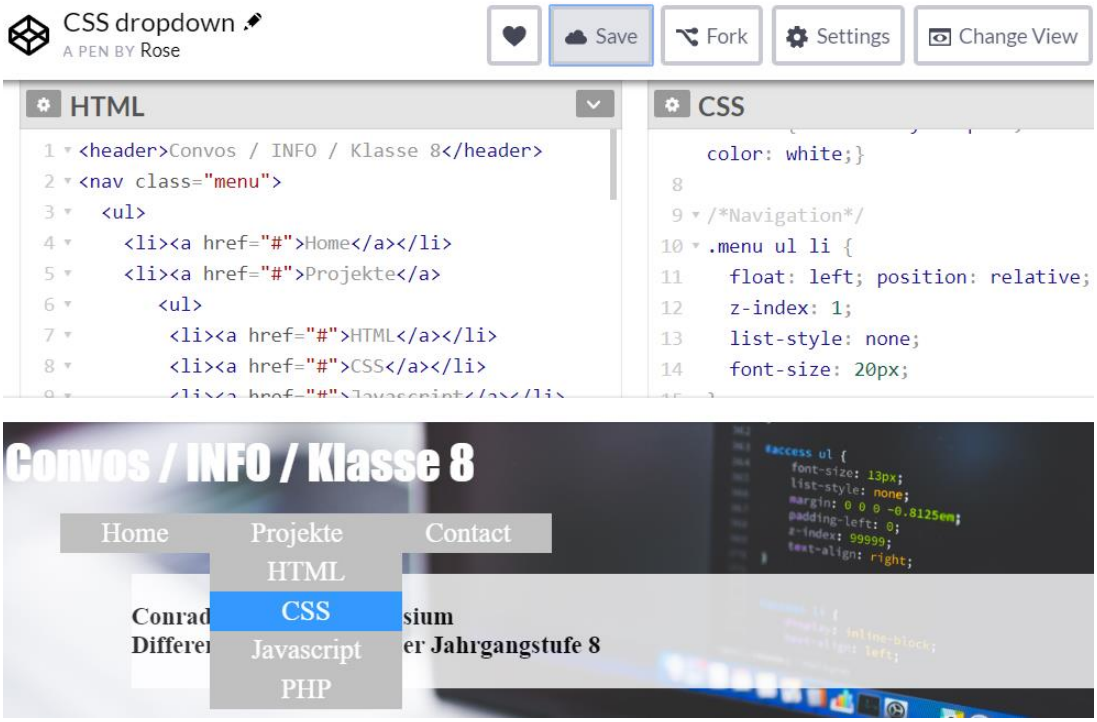
Programmieren in einem einfachen Editor ist etwas für Profis und Puristen. Für Anfänger und Fortgeschrittene empfehle ich die folgenden Webseiten und Tutorials:

selfhtml wiki .org w3schools .com CODEPEN CSS HTML JS mit {Stil} mediaevent.de

selfhtml.org ist ein deutschsprachiges Wiki mit zahlreichen Tutorials und einem aktiven Forum.

w3schools.com bietet dasselbe in englischer Sprache, ist aber noch umfangreicher und sehr beliebt! Klickt auf die Erdkugel um Google die komplette Seite übersetzen zu lassen!

codepen.io ist ein sog. WYSIWYG-Editor, der es euch ermöglicht, in bis zu 3 Fenstern Code in den Formaten HTML, CSS und Javascript einzugeben. Im 4. Fenster seht ihr dann direkt das Ergebnis: *What you see is what you get!*



codepen.io

Sowohl w3schools und codepen.io bieten nicht nur zahlreiche Turorials und Beispiele, die man selbst verändern kann. Sie ermöglichen auch das Testen, Speichern und Teilen von HTML, CSS und JavaScript. Einzelne Seiten und kleinere Projekte können hier schnell realisiert werden.

Kapitel 2 - HTML-Grundlagen

Theorie: Das HTML-Grundgerüst

Das HTML-Grundgerüst ist die Basis einer Webseite. Obwohl die meisten Browser eine HTML-Seite auch so erkennen, sollte sie für alle Fälle das Grundgerüst enthalten.

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="UTF-8">
        <title>Hier stehen Infos für Suchmaschinen und Browser</title>
    </head>
    <body>
                Hier steht der sichtbare Inhalt
    </body>
</html>
```

- Mit Doctype wird festgelegt, welche HTML-Version benutzt wird. Wir benutzen hier HTML 5.
- Im <Head> stehen Informationen, die der Besucher größtenteils nicht sehen kann, z.B. <style> oder <meta> Tags. Das sind Angaben für Suchmaschinen oder den Browser.
- Der Meta-Tag im Beispiel enthält eine Anweisung für den Browser. Er legt mit UTF-8 den deutschen Zeichensatz fest, damit Umlaute werden korrekt dargestellt werden.
- Der <Title> enthält den Titel der Seite.
- Die eigentliche Seite mit sichtbaren Inhalten wird in den Bereich <body> geschrieben. In diesem Beispiel ein Gedicht:

```
    <body>
        <h1>
            Gedicht eines Programmierers
        </h1>
        <p>
            Laut erschallt das Tasten hämmern<br>
            Im Kopf beginnt ein leises dämmern.<br>
            Wie bekomme ich das Programm zum Laufen?<br>
            Das ist doch echt zum Haare raufen
        </p>
        <p>
            Stöhnend über dem Quellcode sitze<br>
            warte ich auf Geistesblitze<br>
            Man tage Therm A und pastet Therm B<br>
            die Fehlermeldung tut richtig weh
        </p>
        <h4>
            von Sundance Raphael
        </h4>
    </body>
</html>
```

Übertrage diesen Quelltext in den HTML Bereich eines neuen Projekts auf codepen.io
Notiere:
Was bewirken die Tags <h1>, <p>, <br>, <h4>?
Welche Regeln folgt die Einrückung mit Hilfe der Tabulatorentaste?

Die **Tags** (übersetzt Markierung) oder auch **HTML-Elemente** legen die Struktur einer Webseite fest. Einige Tags stehen alleine (z.B.
), die meisten Tags kommen aber paarweise vor und bestehen dann aus einem Anfangs- und Endtag (z.B.).

Man unterscheidet **Inline-Elemente** und **Block-Elemente**:

➔ Inline-Elemente stehen im Textfluss und bewirken keinen Zeilenumbruch.

	kursiv
	fett
	betont
<big> </big>	größerer Text
	kleinerer Text
 	Einfacher Zeilenumbruch (Kein End-Tag)

➔ Block-Elemente fügen am Ende automatisch einen Zeilenumbruch ein.

	Überschrift (1-6)
	Absatz
<hr>	Horizontale Linie (Kein End-Tag)

Die Unterscheidungen zwischen diesen **HTML-Elementen *(Tags)*** sollten sich in eueren Quelltexten wiederfinden. Das heißt: **Inline-Elemente** gehören in eine Textzeile, **Block-Elemente** sollten eingerückt über bzw. unter einem Block stehen. (vgl. Gedicht des Programmierers“)

Praxis: Elemente, Attribute, Werte und URLs

Die oben genannten HTML-Elemente ändern die Eigenschaften von Text. Um die Eigenschaften von Links, Bildern und anderen Medien zu verändern braucht man Attribute. Die folgenden HTML-Elemente funktionieren beispielsweise nur mit einem Attribut, dass auf den Pfad (URL) des gewünschten Inhalts verweist. Die URL steht in der Regel in Anführungszeichen hinter dem Attribut im öffnenden HTML-Element.

So fügt man einen Link (anchor) oder ein Bild (image) ein:

```
<a href="https://codepen.io/WRose">Meine Projekte</a>
<img src="https://preview.ibb.co/hS5ppG/img5.jpg">
```

Meistens stehen noch weitere individuelle Attribute bei Links und Bildern. Sie definieren beispielsweise bei Bildern die Maße, die Positionierung oder den Textumbruch. Bei Links kann beispielsweise definiert werden, ob die neue Seite in einem bestimmten Bereich, im aktuellen Browserfenster oder in einem neuen Fenster geöffnet werden soll.

Probiere es! Suche in Tutorials nach nützlichen Attributen für Bilder:

https://wiki.selfhtml.org/wiki/HTML

https://www.w3schools.com/tags/default.asp

https://www.mediaevent.de/xhtml/HTML-Attribute.html

Wie bestimme ich die Höhe und die Breite eines Bildes?

Wie bestimme ich, dass das Bild zentriert ist?

Wie erreiche ich, dass ein Link in einem neuen Fenster geöffnet wird?

Kapitel 3 - CSS-Gestaltung

Theorie: CSS

Du hast bereits gelernt, dass man Eigenschaften von Texten, Bildern und Links mit Inline- und Block-Elementen bzw. mit Attributen definieren kann. Ein Attribut, mit dem man sehr viel bewirken kann ist das style Attribut.

Finde mit dem Editor codepen.io heraus, was die Attribute (text-decoration; display; color; background-color) und die Werte innerhalb des style-Attributs bewirken!

```
<a href= “https://codepen.io/WRose” target= "_blank "
    style= "text-decoration:none; display:inline-block; color:white; background-
    color:black;" >Meine Projekte</a>
```

Eine Seite mit einer Navigationsleiste oder einem Menü enthält zahlreiche Links. Stelle dir einmal vor, wie mühselig es wäre, wenn man jedem Link all diese Attribute und Werte zuweisen müsste? Übersichtlich ist etwas anderes. Und was wäre, wenn man nachträglich etwas ändern möchte oder einen Fehler sucht? Man müsste dies auf jeder Seite und für jedem <a> Tag einzeln vornehmen!

Hier kommt nun CSS ins Spiel! Es handelt sich dabei um eine Sprache zur Definition von Formateigenschaften einzelner HTML-Elemente.

Übrigens ist alles, was innerhalb des Werts beim HTML-Style-Attribut steht CSS. Achte auf die Satzzeichen! Wie unterscheidet sich die Syntax im Vergleich zu HTML?

- **CSS kann z.B. zentral festlegen, dass alle Links automatisch bestimmte Attribute haben.**
- **CSS kann mit Hilfe des HTML Attributs *class* auch zwischen verschiedenen Links unterscheiden bzw. selektieren (z.B. die Links eines Menüs und die auf der Seite)**
- **CSS kann Elemente pixelgenau im Anzeigefenster positionieren.**
- **CSS kann in separaten Dateien notiert werden und dann in beliebige Dateien eingebunden werden. Dies vereinfacht ein einheitliches Layout.**
- **Mit CSS *Stylesheets* kann man das Aussehen einer Seite mit einem Klick vollständig verändern**

Webdesigner lieben CSS!

Eine beeindruckende Demonstration findest Du hier:
https://www.w3schools.com/css/demo_default.htm

CSS im <Style> Element

CSS enthält Informationen, die lediglich für den Browser bestimmt sind. Diese Informationen gehören deshalb in den <head>-Bereich innerhalb eines <style> Blockelements. In diesem Bereich gilt die CSS Syntax!

Die Attribute stehen hinter den Selektoren in {geschweiften Klammern!} (Tastenkombination: Alt Gr + 7 oder 0)

Style-Element mit 2 Selektoren

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="UTF-8">
        <title>CSS mit Style-Element</title>
        <style type="text/css">
            h1 { color:red; font-size:38px; }
            p { word-spacing:0.5em; font-family:'Verdana',sans-serif; }
        </style>
    </head>
    <body>
        <h1>
            CSS mit Style-Element
        </h1>
        <p>
            Mit dem Style-Element werden HTML-Tags Formatierungen
zugewiesen.
        </p>
    </body>
</html>
```

Vorteil: Du kannst HTML Elementen globale Attribute zuweisen und brauchst dafür keine weitere Datei.

Nachteil: In der Regel befinden sich die Informationen einer Webseite in unterschiedlichen .html Dateien. Ändert man etwas im <style> Bereich einer Datei, muss man es in allen anderen manuell nachtragen.

CSS in externer Datei

Vor allem in größeren Homepages macht es Sinn, die CSS-Formatierungen in einer externen Datei zu speichern. Die Datei wird unter der Endung .css gespeichert und kann dann in allen HTML-Seiten eingebunden werden. Alle Seiten mit dem <link> Verweis innerhalb des <head> Bereichs haben so die gleiche Formatierung.

seite_1.htm

```
<html>
    <head>
        <meta charset="UTF-8">
        <title>Seite 1</title>
        <link rel="stylesheet" type="text/css" href="styles.css"/>
    </head>
    <body>
        <h1>
            Seite 1
        </h1>
        <p>
            Dies ist die Seite 1.
        </p>
        <p>
            Hier geht es zu <a href="seite_2.html">Seite 2</a>
        </p>
      </body>
</html>
```

Öffne den Windows Texteditor und übertrage den Quellcode (copy + paste mit Strg + C und Strg + V). Speichere einmal als seite_1.html

Ersetze im <body> jede 1 durch eine 2 und jede 2 durch eine 1. Speichere nochmal als seite_2.html

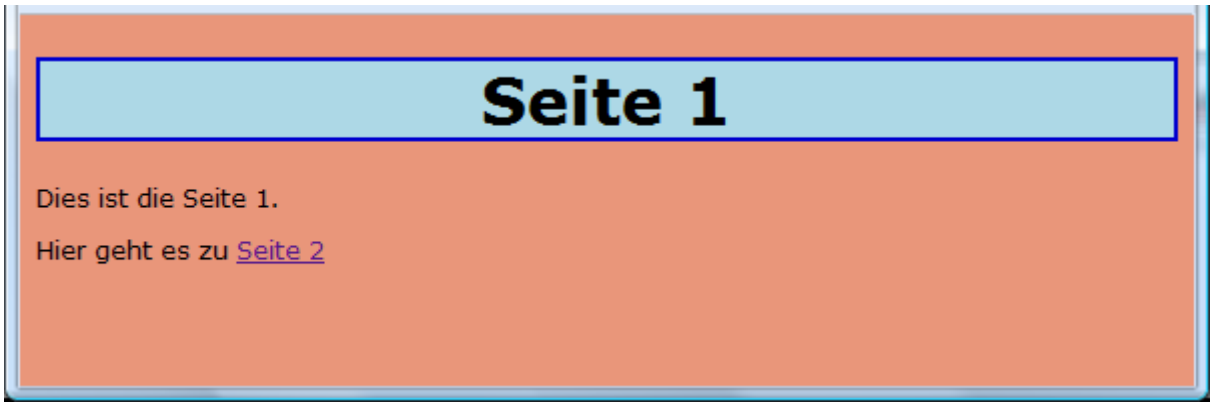
Erstelle nun eine style.css mit folgenden **Typselektoren** und speichere:

Globale Typselektoren beziehen sich auf alle HTML-Elemente eines Typs. Auch der Asterisk * ist ein globaler Selektor. Er bestimmt, dass die Standardschrift für alle HTML-Elemente durch die Schrift Verdana ersetzt wird.

styles.css

```
*      { font-family:Verdana,sans-serif; }
body   { background: darksalmon; }
h1     { text-align:center; border:2px solid mediumblue; background-
         color: lightblue; }
p      { font-size:12px; }
```

Wenn alles richtig ist, sollte die Seite 1 so aussehen und funktionieren. Das bekommst Du bestimmt noch schöner hin 😊



Typ- und Klassenselektoren

Häufig kommt es vor, dass sich einzelne HTML-Elemente von gleichnamigen unterscheiden sollen. Angenommen nicht alle … -Absätze sollen die gleiche Farbe erhalten.

Hier können im Style-Element bzw. in der CSS-Datei Klassen erzeugt werden, die dann den HTML-Elementen zugewiesen werden können.

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="UTF-8">
        <title>Der Zauberlehrling</title>
        <style type="text/css">
            body { background: silver; }
            h2 { color:midnightblue; font-size:22px; }
            p { font-variant:small-caps; }
            .gruen { color:#006500; }
            .lila { color:purple; )
        </style>
    ad>
```

Universalselektor:
Gilt im ganzen Dokument, wie auch das Asterisk *

Typselektor:
Gilt für alle HTML-Elemente des Typs unabhängig ihrer Klasse.

Klassenselektor:
Gilt nur für das HTML-Element

```
    <body>
        <h2>
            Der Zauberlehrling
        </h2>
        <p class="gruen">
            Hat der alte Hexenmeister<br>
            Sich doch einmal wegbegeben!<br>
            Und nun sollen seine Geister<br>
            Auch nach meinem Willen leben.<br>
            Seine Wort und Werke<br>
            Merkt ich und den Brauch,<br>
            Und mit Geistesstärke<br>
            Tu ich Wunder auch.
        </p>
        <p class="lila">
            Walle! walle<br>
            Manche Strecke,<br>
            Daß zum Zwecke,<br>
            Wasser fließe<br>
            Und mit reichem, vollem Schwalle<br>
            Zu dem Bade sich ergieße.
        </p>
    </body>
</html>
```

HTML-Klassenattribut

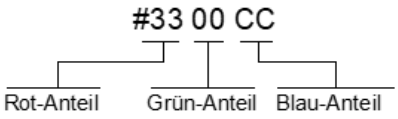
Hier kannst Du beliebige Namen vergeben. Meide nur Umlaute etc.

Klassenformatierungen und globale Formatierungen können beliebig ergänzt werden.

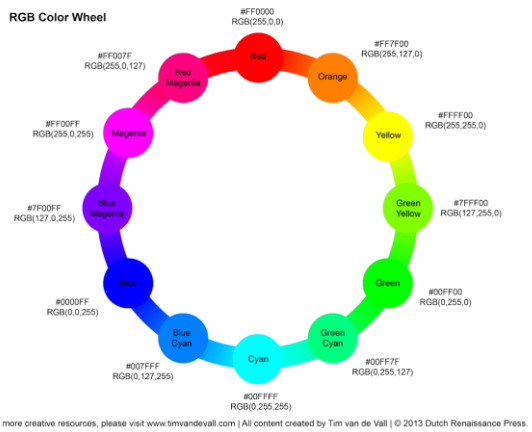
Suche im Internet den kompletten Text des Zauberlehrlings von Goethe und formatiere jede Strophe in einer eigenen Farbe. Die Farbnamen findest du ebenfalls im Internet.

Geläufige Farben haben kannst du in englischer Sprache angeben. Alternativ kannst du mit einem Hexadezimalwert einen genauen Farbton angeben. Z.B. color:#3300CC;

Binary	Decimal	Hexadecimal
0000	0	0
0001	1	1
0010	2	2
0011	3	3
0100	4	4
0101	5	5
0110	6	6
0111	7	7
1000	8	8
1001	9	9
1010	10	A
1011	11	B
1100	12	C
1101	13	D
1110	14	E
1111	15	F

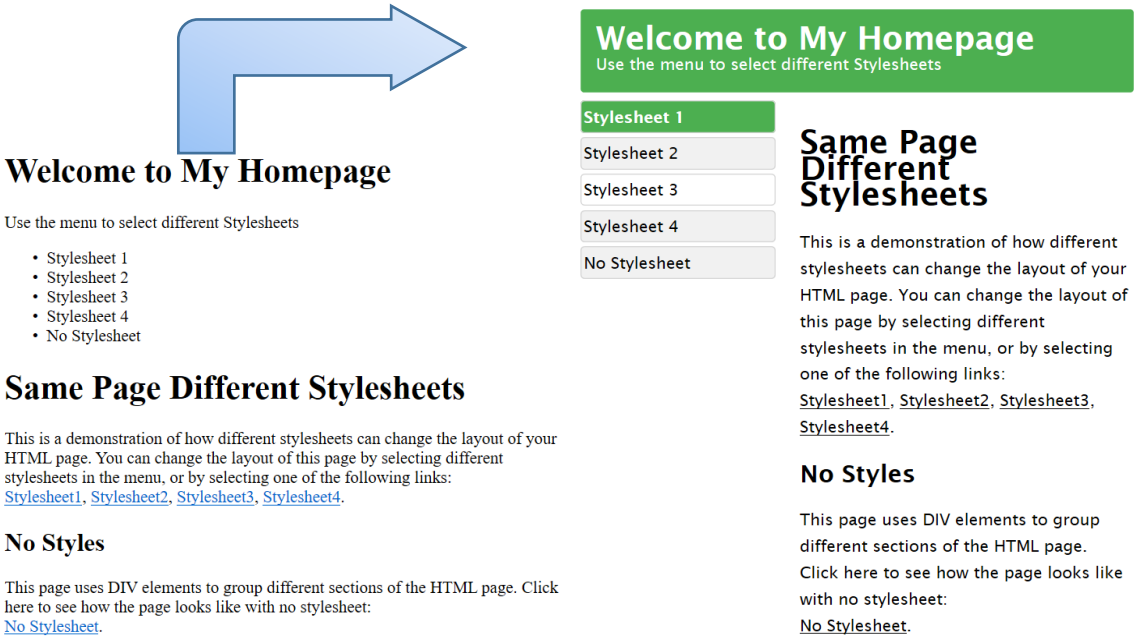


Praktischer Farbwähler:
https://www.w3schools.com/colors/colors_picker.asp



Weitere HTML-Elemente

Hier nochmal der Vergleich: Ohne CSS-Stylesheet (links) und mit Stylesheet (rechts).



https://www.w3schools.com/css/demo_default.htm

Was ist rechts aus den Listeneinträgen geworden?
Was sind eigentlich die besagten Elemente?
Wie wurde der Inhalt nach rechts eingerückt?
Wieso wandert der Inhalt unter das Menü,
wenn das Browserfenster schrumpft?

Theorie: Listen

Im HTML unterschiedet zwischen sortierten und unsortierten Listen. Beide werden zunächst mit HTML erstellt und dann ggf. mit CSS gestaltet. Wie im Beispiel oben, kann man später aus verschachtelten Listen Dropdown-Menüs machen.

Beispiel: Unsortierte Liste

```
<ul>
   <li>Deutsch</li>
   <li>Mathematik</li>
   <li>Englisch</li>
</ul>
```

- Deutsch
- Mathematik
- Englisch

Beispiel: Sortierte Liste

```
<ol>
   <li>Hausaufgaben machen</li>
   <li>Lernen für die Arbeit</li>
   <li>Mit Freunden treffen</li>
</ol>
```

1. Hausaufgaben machen
2. Lernen für die Arbeit
3. Mit Freunden treffen

Mit dem **CSS-Style** list-style-type: kann das Aussehen von Listenzeichen kontrollieren.

`list-style-type: decimal`	für `ol`-Listen:	Nummerierung 1.,2.,3.,4.
`list-style-type: upper-roman`	für `ol`-Listen:	Nummerierung I.,II.,III.,IV.
`list-style-type: upper-latin`	für `ol`-Listen:	Nummerierung A.,B.,C.,D.
`list-style-type: disc`	für `ul`-Listen:	gefüllter Kreis (Bullet)
`list-style-type: circle`	für `ul`-Listen:	leerer Kreis
`list-style-type: square`	für `ul`-Listen:	rechteckiges Bullet-Zeichen
`list-style-type: none`		kein Zeichen

Übung: Listen

- Erstelle eine unsortierte Liste mit einer Einkaufsliste mit Überschrift.

1. Erstelle eine sortierte Liste mit Urlaubswünschen
2. Erstelle eine verschachtelte Liste mit Unterpunkten
   - wie hier
   - oder hier

Weiterführung: Bilder und Textumlauf

Du weißt bereits, dass Bilder in HTML mit dem <img>-Element eingefügt werden und dass das <img>-Element keinen abschließenden Tag hat. Bilder auf Internetseiten dürfen nur die Formate JPEG, GIF oder PNG haben. Dabei ist darauf zu achten, dass die Dateigröße möglichst klein ist, um lange Ladezeiten zu vermeiden. Die Bildgröße sollte auch deshalb nur in Ausnahmefällen mit HTML verkleinert werde. Besser ist es, das Bild vor dem Einbau mit einem Grafikprogramm zu verkleinern. In der Regel genügen 72dpi (dots per inch).

Übung: Textumlauf

Kopiere den folgenden Code in den HTML-Bereich eines codepens.

- Ebenso wie in einem Word-Dokument verdrängt das Bild den ganzen Text selbst dann, wenn rechts oder links noch Platz wäre.
- Ebenso sieht ein Text im Blocksatz sauberer aus und er sollte auch nicht so in die Breite gezogen werden.

```
<body>
      <p>
      <img src=https://preview.ibb.co/hS5ppG/img5.jpg>
      In a distant, but not so unrealistic, future where mankind has
      abandoned earth because it has become covered with trash from
      products sold by the powerful multi-national Buy N Large
      corporation, WALL-E, a garbage collecting robot has been left to
      clean up the mess. Mesmerized with trinkets of Earth's history and
      show tunes, WALL-E, short for Waste Allocation Load Lifter Earth-
      class, is alone on Earth except for a sprightly pet cockroach. He
      spends his days tidying up the planet, one piece of garbage at a
      time. But during 700 years, WALL-E has developed a personality,
      and he's more than a little lonely.
      </p>
```

Vergleiche die Funktionen die man mit Word bzw. mit CSS gebrauchen kann. Um den verfügbaren Platz besser zu nutzen und die Proportionen sinnvoller zu gestalten:

Word	CSS-Attribut	Wert	weitere Werte
Blocksatzfunktion	text-align	justify	left, center, inherit
Textumbruch bei Bildeigenschaften	float	left	right, inherit
Rechter Einzug in Absatzformatierung	margin-right	z.B. 30px	z.B. feste Pixelwerte oder flexible Prozentwerte
Skalierung der Bildgröße	width	z.B. 30%	z.B. ein fester Pixelwert

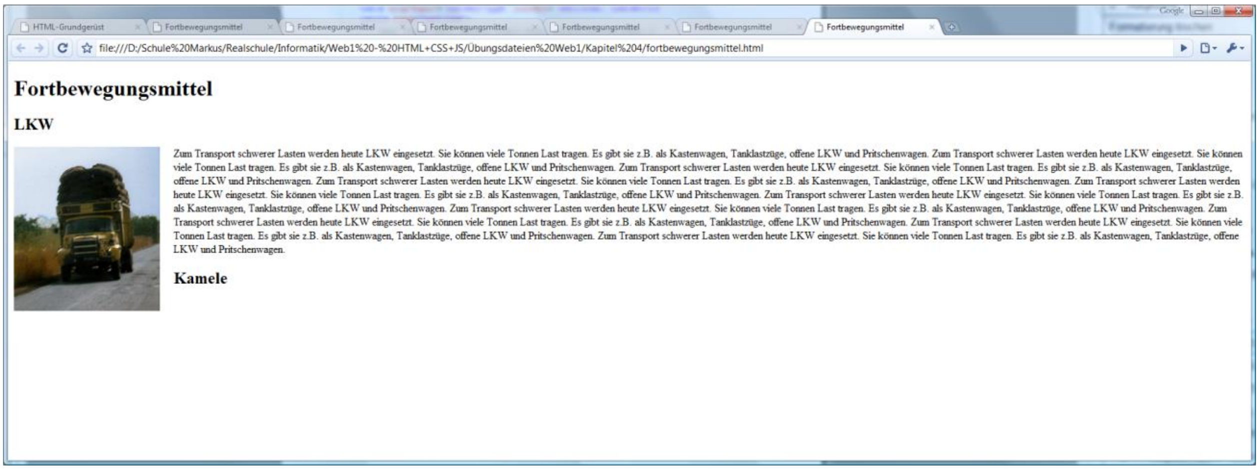
Experimentiere in codepen.io mit diesen CSS-Attributen. Denk an die Syntax und schau dir ggfls. nochmal das Beispiel auf Seite 10 an und die folgende Hilfeseite an:

wiki.selfhtml.org/wiki/CSS/Eigenschaften/Positionierung/float

Das Ergebnis soll in etwa wie folgt aussehen:

In a distant, but not so unrealistic, future where mankind has abandoned earth because it has become covered with trash from products sold by the powerful multi-national Buy N Large corporation, WALL-E, a garbage collecting robot has been left to clean up the mess. Mesmerized with trinkets of Earth's history and show tunes, WALL-E, short for Waste Allocation Load Lifter Earth-class, is alone on Earth except for a sprightly pet cockroach. He spends his days tidying up the planet, one piece of garbage at a time. But during 700 years, WALL-E has developed a personality, and he's more than a little lonely.

Wenn man das CSS-Format *float* benutzt, kann es aber auch zu unerwünschten Nebeneffekten kommen. Hier wurde die Seite sehr breit geöffnet. Achte auf die Überschrift „Kamele":



Um zu verhindern, dass der nächste Absatz auch um ein Bild umgebrochen wird, muss man hier den Befehl *style="clear:left"* bzw *style="clear:right"* ergänzen.

```
Zum Transport schwerer Lasten werden heute LKW eingesetzt.
</p>
<h2 style="clear:left">
```

```
    Kamele
  </h2>
```

Du kannst die Seite mit den Filmen noch um einige erweitern. Versuche auch, den Textumfluss abwechselnd zu tauschen. Dafür kannst du bei einigen Bildern mit style= die Formatierung im Head verändern.

```
  <h2 style="clear:left">
    Kamele
  </h2>
  <p>
    <img src="kamel.jpg" alt="Kamel" style="float:right"/>
    Das Kamel lebt hauptsächlich in Wüsten.
  </p>
```

Natürlich solltest du jetzt auch den Abstand zur Seite anpassen.

Theorie: Tabellen

Tabellen haben in HTML zwei Aufgaben:

➔ Darstellung von tabellarischen Daten, z.B. Fußballergebnisse

➔ Seitenformatierung mit unsichbaren Tabellen

Grundsätzlich sieht der HTML-Code für eine Tabelle so aus:

```
<body>
    <table>
        <tr>
            <th>Kopfzelle 1</th>
            <th>Kopfzelle 2</th>
            <th>Kopfzelle 3</th>
        </tr>
        <tr>
            <td>Inhalt 1</td>
            <td>Inhalt 2</td>
            <td>Inhalt 3</td>
        </tr>
        <tr>
            <td>Inhalt 4</td>
            <td>Inhalt 5</td>
            <td>Inhalt 6</td>
        </tr>
</body>
```

Kopfzelle 1	Kopfzelle 2	Kopfzelle 3
Inhalt 1	Inhalt 2	Inhalt 3
Inhalt 4	Inhalt 5	Inhalt 6

Einige Erklärungen zur Tabelle:

<table>: Hier beginnt die Tabelle. Standardmäßig wird kein Rahmen angezeigt. Wir sprechen dann von einer **versteckten Tabelle**.

Soll ein Rahmen angezeigt werden, muss dies in CSS folgendermaßen angegebne werden: `table, td, th {border: 1px solid red;}`

<tr></tr>: Das Element *<tr></tr>* (row) legt jeweils eine neue Tabellenzeile an.

<th></th>: Das Element *<th></th>* (head) legt jeweils eine Tabellenkopf-Zelle an. Diese hebt sich von einer normalen Zelle durch eine fette Schrift ab und kann mit CSS einzeln formatiert werden.

<td></td>: Das Element *<td></td>* (data) legt jeweils eine normale Tabellen-Zelle an.

Eine Tabellenformatierung mit CSS kann dann so aussehen:

```
        <style type="text/css">
            table, td, th      {border: 1px solid red;}
            table              {width:400px; height:200px;}
            th                 {text-align:center; background-
color:gray;}
            td                 {text-align:justify; vertical-align:top;}
        </style>
```

Weitere Formatierungsmöglichkeiten werden in den Übungen erklärt.

Kopfzelle 1	Kopfzelle 2	Kopfzelle 3
Inhalt 1	Inhalt 2	Inhalt 3
Inhalt 4	Inhalt 5	Inhalt 6

Übung: Tabellen

Erstelle folgende Tabellen:

Fußballtabelle

Heim	Gast	Ergebnis
FC Naunheim	WFC	3 : 1
Essener CV	Eintracht Herne	0 : 2

Soll nur eine Spalte eine bestimmte Größe erhalten, gibt man im ersten Tag der Spalte die Breite an: *<th style="width:200px">Adresse</th>*

Sollen in einer Tabelle zwei nebeneinander liegende Zellen verbunden werden, benutzt man das Attribut *colspan*.

```
<table>
   <tr>
     <th colspan="2">Zelle über 2 Spalten</th>
   </tr>
   <tr>
     <td>rechte Zelle</td>
     <td>linke Zelle</td>
   </tr>
 </table>
```

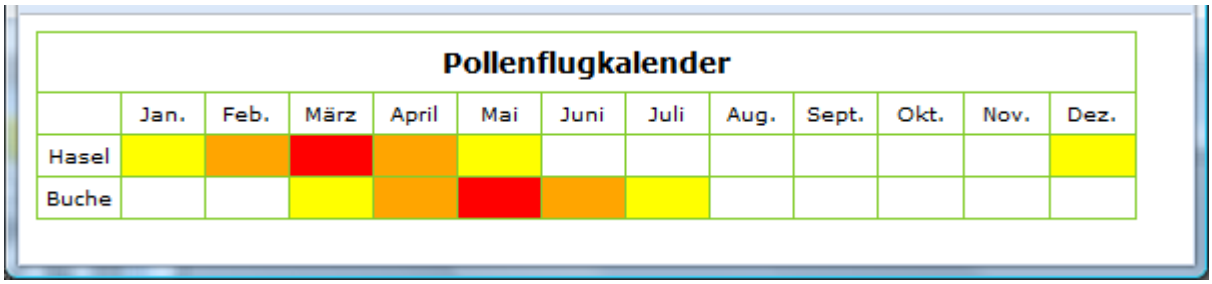
Zelle über 2 Spalten	
rechte Zelle	linke Zelle

Um mehrere untereinander liegende Zellen zu verbinden, nutzt man das Attribut *rowspan*.

```
<table>
  <tr>
    <td rowspan="2">Zelle über 2 Zeilen</td>
    <td>obere Zelle</td>
  </tr>
  <tr>
    <td>untere Zelle</td>
  </tr>
</table>
```

Zelle über 2 Zeilen	obere Zelle
	untere Zelle

Für diese Seite ist viel CSS nötig.



Pollenflugkalender

	Jan.	Feb.	März	April	Mai	Juni	Juli	Aug.	Sept.	Okt.	Nov.	Dez.
Hasel												
Buche												

```
<style type="text/css">
    *       { font-family:Verdana,sans-serif;
              text-align:center; }
    table, td, th {border: 1px solid #85CD2E;
              border-collapse: collapse}   /* kein Doppelrahmen */
    table   { width:550px; }
    th      { font-size:14px;
              padding:6px;}
    td      { font-size:10px;
              height:14px;
              padding:3px;                  /* Innenabstand */
              min-width:35px; }             /* Mindestbreite */
    .stark  { background:red; }
    .mittel { background:orange; }
    .schwach{ background:yellow; }
</style>
```

Theorie: Absolute Positionierung

Absolut positionierte Objekte richten sich nicht nach dem vorausgehenden HTML Elementen. Sie orientieren sich in der Regel am Browserfenster oder an einem relativ positionierten HTML-Element, dass sie beinhaltet.(Die Position wird in Pixeln angegeben. Top:0; left:0 ist oben links im Browserfenster.)

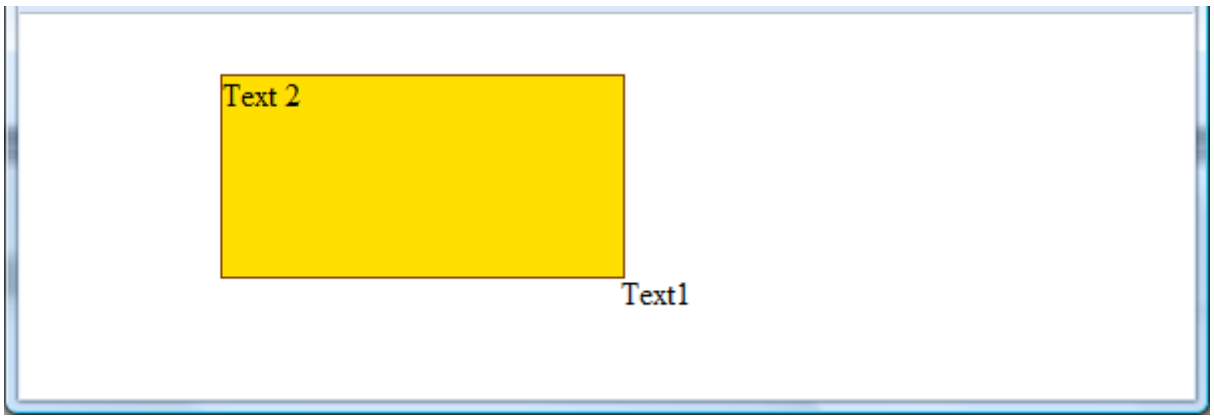
Man kennt das Prinzip in Textverarbeitungsprogrammen unter dem Begriff „Textfeld“  , ein „Post-IT“ würde auf einem Blatt Papier den gleichen Zweck erfüllen.

```
<body>
  <div style="position:absolute; top:130px; left:300px">Text</div>
</body>
```

Das erzeugt einen Bereich (auch „Container“), für den in CSS zahlreiche Eigenschaften deklariert werden können. Das HTML-Element verhält sich wie ein Blockelement. Da es jedoch weitere Elemente (Texte, Bilder, Tabellen, Listen) enthalten kann, muss ein Schluss-Tag verwendet werden.

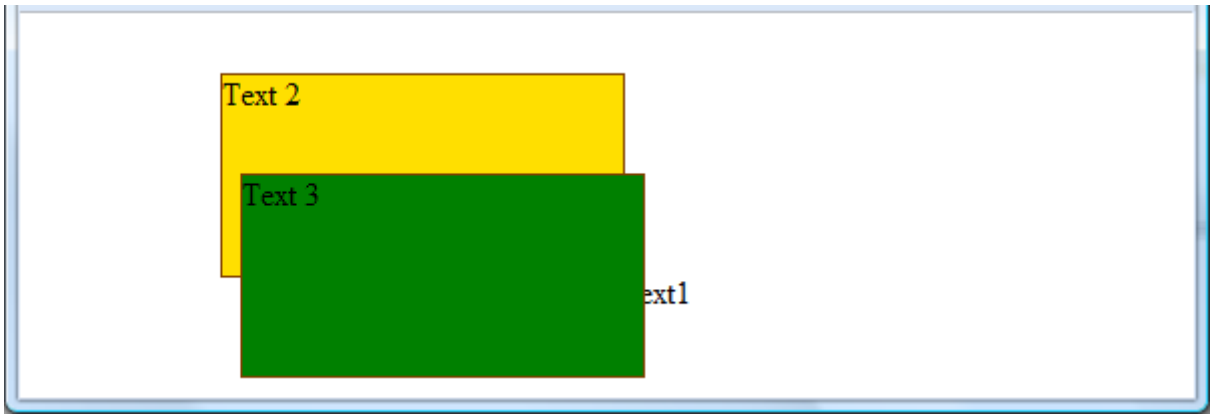
Gibt man dem Div-Element weitere CSS-Attribute hinzu, erhält das Element z.B. einen Rahmen oder einen Hintergrund.

```
 9  <body>
10    <div style="position:absolute; top:130px; left:300px">Text1</div>
11    <div style="position:absolute; top:30px; left:100px; width:200px;
12      height:100px; background-color:#FFDF00; border:1px solid #804000;">
13      Text 2
14    </div>
15  </body>
```



Eine weitere Besonderheit der absoluten Positionierung, ist die Überlagerung anderer Inhalte. Liegen mehrere Ebenen übereinander, entscheidet die Reihenfolge im Quelltext. Es gilt: Je später im Quelltext, umso weiter oben.

```
 9   <body>
10     <div style="position:absolute; top:130px; left:300px">Text1</div>
11     <div style="position:absolute; top:30px; left:100px; width:200px;
12       height:100px; background-color:#FFDF00; border:1px solid #804000;">
13       Text 2
14     </div>
15     <div style="position:absolute; top:80px; left:110px; width:200px;
16       height:100px; background-color:green; border:1px solid #804000;">
17       Text 3
18     </div>
19   </body>
20 </html>
```

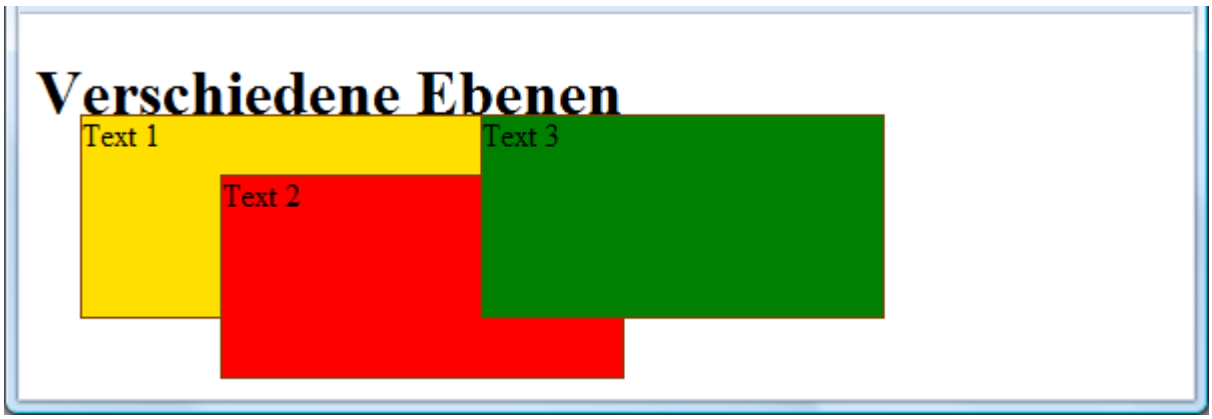


Diese Eigenschaft kann man sich in einem Dropdown-Menü zu Nutze machen, indem man ein absolut positioniertes Element (z.B. oder <ul>) innerhalb eines relativ positionierten Elements einfügt. Das absolute positionierte Element orientiert sich dann am relativ positioniert Element (z.B. ein Button oder ein Link).

(siehe „Menü mit Buttons und Untermenü“)

Übung: Absolute Positionierung

Erstelle folgende Webseite:



Erstelle eine Webseite mit mehreren Ebenen, die auch Bilder enthalten.

Zusatz: CSS3

Mit CSS3 kannst du Schattierungen oder abgerundete Rahmen (border-radius box-shadow) erstellen. Wenn du in einer Suchmaschine „CSS3 generator“ eingibst, findest du Webseiten, mit denen du dies einfach erstellen kannst.

Kapitel 4 - Aufbau von Webseiten

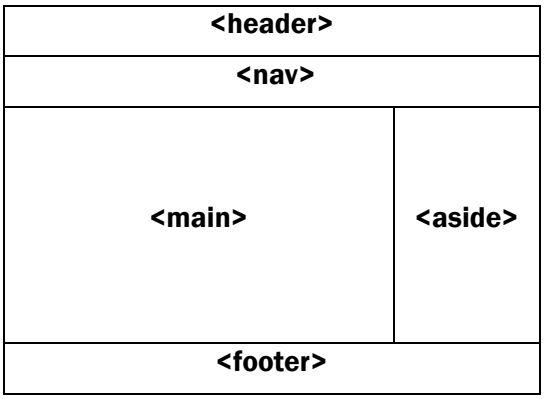
Theorie: Semantisches HTML

Semantisches HTML bedeutet, die Webseite zu strukturieren und dieses mit eigenen „bedeutungsvollen“ Tags zu strukturieren. Das Layout der Seite ist dabei zunächst nebensächlich und wird erst später mit CSS festgelegt.

Beispiele für Semantisches HTML:

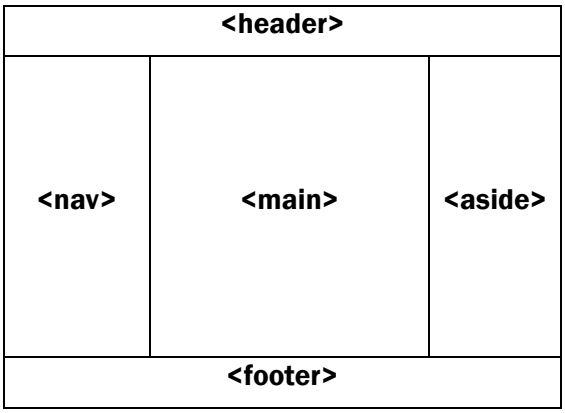
Zweispaltiges Layout

(horizontale Navigation)



Dreispaltiges Layout

(vertikale Navigation)



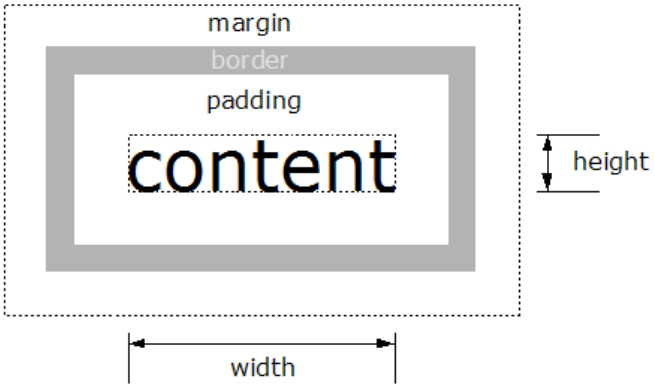
Theorie: Navigation

Will man nicht nur eine einzelne Seite erstellen, benötigt man ein Navigationsmenü, um zwischen den verschiedenen Seiten des Web-Auftrittes (Website) wechseln zu können.

Momentan ist es üblich, die Navigationspunkte in eine unsortierte Liste <ul> zu schreiben und dann mit CSS zu formatieren.

Theorie: Box-Modell

Das Box-Modell zeigt, welche Bedeutung die Größen- und Abstandsattribute haben:



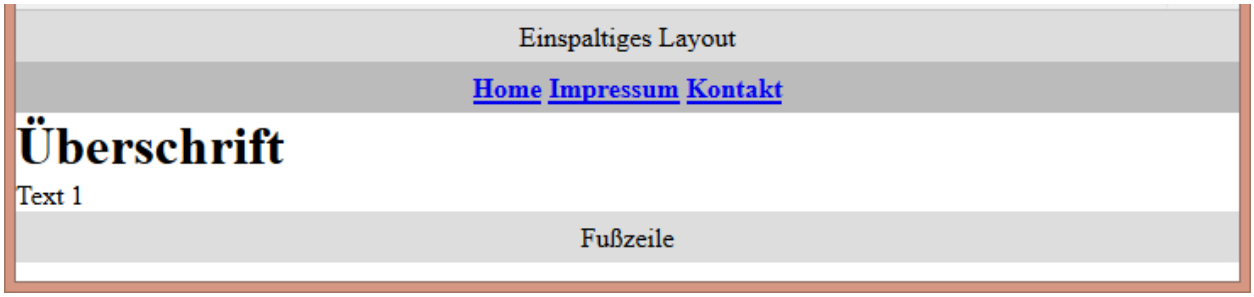
Noch viel anschaulicher wird es in diesem anschaulichen Pen!

codepen.io/carolineartz/full/ogVXZj

Übung: Einspaltiges Layout

Wichtig: Der Body-Teil ist bei drei Layout-Beispielen in diesem Kapitel sehr ähnlich aufgebaut. Die Unterschiede liegen überwiegend an Veränderungen in den CSS-Styles.

```
<!doctype html>
<html>
    <head>
        <title>Einspaltiges Layout</title>
        <meta charset="UTF-8">
        <style type="text/css">
            *               {   margin:0px;
                                padding:0px;
                                border: 0px;  }
            header, footer  {   background-color: #DDDDDD;
                                padding: 5px;
                                text-align:center; }
            nav             {   background-color: #BBBBBB;
                                padding: 5px;
                                text-align:center;
                                font-weight: bold; }
            nav ul          {   list-style-type: none;  }
            nav ul li       {   display: inline;    }
        </style>
    </head>
    <body>
        <header>Einspaltiges Layout</header>
        <nav>
            <ul>
                <li><a href="#">Home</a></li>
                <li><a href="#">Impressum</a></li>
                <li><a href="#">Kontakt</a></li>
            </ul>
        </nav>
        <main>
            <h1>Überschrift</h1>
            <p>Text 1</p>
        </main>
        <footer>Fußzeile</footer>
    </body>
</html>
```



Einige Erklärungen zum Style-Element:

*	Der Universalselektor * setzt alle Abstände zunächst auf 0, da sie sonst in den Browsern unterschiedliche Standardwerte haben können.
nav ul	Hier wird die Liste formatiert, die innerhalb der Navigation liegen.
nav ul li	Mit display: inline wird festgelegt, das die Listeneinträge in der Navigation keinen Zeilenumbruch haben.

Übung: Zweispaltiges Layout

```
<!doctype html>
<html>
    <head>
        <title>Zweispaltiges Layout</title>
        <meta charset="UTF-8">
        <style type="text/css">
            *               {   margin:0px;
                                padding:0px;
                                border: 0px; }
            header, footer  {   background-color: #DDDDDD;
                                padding: 5px;
                                text-align:center; }
            nav             {   background-color: #BBBBBB;
                                padding: 5px;
                                text-align:center;
                                font-weight: bold; }
            nav ul          {   list-style-type: none; }
            nav ul li       {   display: inline;
                                padding: 0px 15px }
            #spalten        {   display: table;
                                min-width: 720px;
                                max-width: 60em; }
            aside           {   background-color: #AAAAAA;
                                padding: 5px;
                                width: 180px;
                                display: table-cell; }
            main            {   padding:  5px;
                                display: table-cell; }
        </style>
    </head>
    <body>
        <header>Zweispaltiges Layout</header>
        <nav>
            <ul>
                <li><a href="#">Home</a></li>
                <li><a href="#">Impressum</a></li>
                <li><a href="#">Kontakt</a></li>
            </ul>
        </nav>
        <div id="spalten">
            <aside>
                <h3>Seitenleiste</h3>
                <p>z.B. Links oder Untermenü</p>
                <ul>
                    <li><a href="#">Link 1</a></li>
                </ul>
            </aside>
            <main>
                <h1>Inhalt</h1>
                <p>Hier steht der Text. … Hier steht der Text.</p>
            </main>
        </div>
        <footer>Fußzeile</footer>
    </body>
</html>
```

Mit #spalten formatieren wir das HTML-Element mit der ID „spalten“. Die IDs ähneln den Klassen

padding: 0px 5px legt den Abstand zwischen den Listenpunkten fest.

O nach oben und unten

15 nach links und rechts

Mit CSS display:table wird festgelegt, dass sich die darin liegenden Elemente (display: table-cell) wie Tabellenzellen benehmen sollen.

Sie haben dann z.B. die gleiche Höhe.

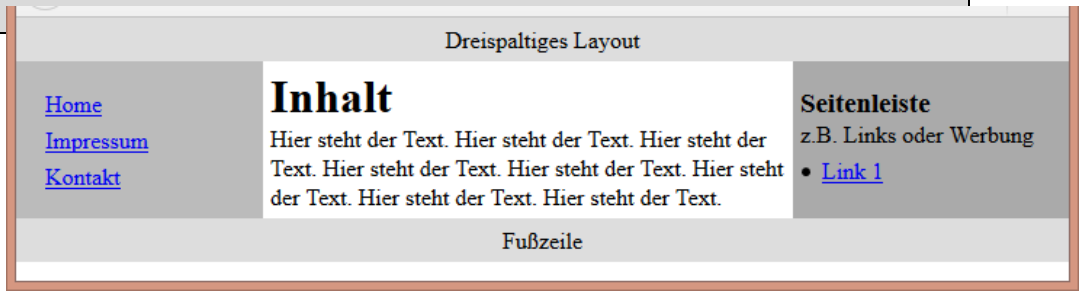
Übung: Dreispaltiges Layout

```
<!doctype html>
<html>
    <head>
        <title>Dreispaltiges Layout</title>
        <meta charset="UTF-8">
        <style type="text/css">
            *                {   margin:0px;
                                 padding:0px;
                                 border: 0px;  }
            body             {   min-width: 720px;
                                 max-width: 60em;}
            header, footer   {   background-color: #DDDDDD;
                                 padding: 5px;
                                 text-align:center;}
            #spalten         {   display: table; }
            nav              {   background-color: #BBBBBB;
                                 padding: 5px;
                                 width: 160px;
                                 display: table-cell; }
            nav ul           {   list-style-type: none; }
            aside            {   background-color: #AAAAAA;
                                 padding: 5px;
                                 width: 180px;
                                 display: table-cell;  }
            main             {   padding:  5px;
                                 display: table-cell; }
            li               {   margin: 5px 15px  }
        </style>
    </head>
    <body>
        <header>Dreispaltiges Layout</header>
        <div id="spalten">
            <nav>
                <ul>
                    <li><a href="#">Home</a></li>
                    <li><a href="#">Impressum</a></li>
                    <li><a href="#">Kontakt</a></li>
                </ul>
            </nav>
            <main>
                <h1>Inhalt</h1>
                <p>Hier steht der Text. … Hier steht der Text. Hier
steht der Text. Hier steht der Text.</p>
            </main>
            <aside>
                <h3>Seitenleiste</h3>
                <p>z.B. Links oder Werbung</p>
                <ul>
                    <li><a href="#">Link 1</a></li>
                </ul>
            </aside>
        </div>
        <footer>Fußzeile</footer>
    </body>
</html>
```

Wird hier das Wort „auto“ ergänzt (vor dem ;), ist das Layout immer im Fenster zentriert.

Die minimale Größe des Layouts ist 720px. Ist das Fenster kleiner, gibt es Scrollbalken.

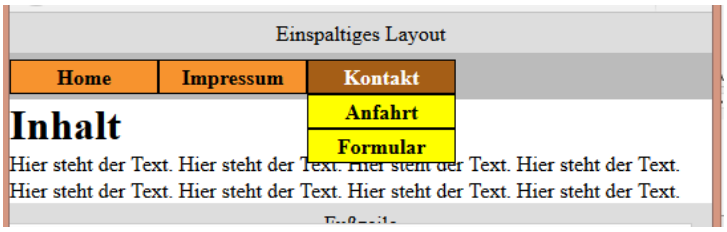
Die maximale Größe sind 60 Zeichen. Dies dient der besseren Lesbarkeit.



Übung: Menü mit Buttons

Bisher besteht das Menü aus Wörtern. Sollen sie einen Button-Effekt bekommen, muss im Style-Element folgendes ergänzt werden:

```
nav ul li a          {   border: 1px solid #000000;
                         background-color: #F7932E;
                         padding: 2px 5px 2px 5px;
                         color: #000000;
                         text-decoration: none;
                         font-weight: bold; }
nav ul li a:hover    {   background-color: #A55E16;
                         color: #ffffff; }
```



Zunächst müssen die Untermenüpunkte als verschachtelte Liste im Body angegeben werden. Später wird die übergeordnete Liste relativ positioniert und die untergeordnete Liste mit weiteren Links (Anfahrt; Formular) absolut positioniert. Auf diese Weise orientiert sich das Untermenü immer an der Position des Menüpunkts und es überdeckt durch die absolute Positionierung andere HTML-Inhalte (siehe oberes Bild):

```
<nav>
    <ul>
        <li><a href="#">Home</a></li>
        <li><a href="#">Impressum</a></li>
        <li>
            <a href="#">Kontakt</a>
            <ul>
                <li><a href="#">Anfahrt</a></li>
                <li><a href="#">Formular</a></li>
            </ul>
        </li>
    </ul>
</nav>
```

Achte auf die schliessenden End-Tags und die Formatierung: Der 3. Listenpunkt <li> der ungeordneten Menüliste <ul> enthält in Zeile 7 eineweitere untergeordnete Liste <ul> mit zwei weiteren Punkten bzw. Links <a>. Dies ist das Dropdownmenü!

Damit das Untermenü nur bei Bedarf sichtbar wird, muss im CSS das Display-Attribut (display: none / display: block) verwendet werden. Auf der Webseite findest du unter Projekte HTML/CSS hierzu Erklärungen mit Beispielen. Du kannst aber auch einfach mit dem unternstehenden CSS experimentieren und selbst herausfinden, was die Werte (none; block; inline-block) bewirken.

Die entscheidenden Erweiterungen des CSS Stylesheets für ein Dropdown-Menü sind im Folgenden fett gedruckt. Der Zustand „hover“ tritt ein, wenn sich die Maus im Bereich eines Links oder eines Listeneintrags befindet. Nur in diesem Fall wird das Untermenü sichtbar:

```
<style type="text/css">
    *                           {   margin:0px;
                                    padding:0px;
                                    border: 0px; }
    header, footer              {   background-color: #DDDDDD;
                                    padding: 5px;
                                    text-align: center; }
    nav                         {   background-color: #BBBBBB;
                                    height: 30px;
                                    padding: 5px 0px 0px 0px;
                                    text-align: center;
                                    font-weight: bold; }
    nav ul                      {   list-style-type: none;  }
    nav ul li                   {   display: inline-block
                                    position: relative; }
    nav ul li a                 {   display: block;
                                    width: 100px;
                                    border: 1px solid #000000;
                                    background-color: #F7932E;
                                    padding: 2px 5px 2px 5px;
                                    color: #000000;
                                    text-decoration: none;
                                    font-weight: bold; }
    nav ul li a:hover           {   background-color: #A55E16;
                                    color: #ffffff; }
    nav ul li ul                {   display: none;
                                    position: absolute; }
    nav ul li:hover ul          {   display: block; }
    nav ul li:hover ul a        {   display: block;
                                    width: 100px;
                                    color: black;
                                    text-align: center;
                                    background-color: yellow; }
    nav ul li ul li a:hover     {   background-color: silver; }
</style>
```

Theorie: Webseite mit mehreren Seiten

Will man eine Webseite aus mehreren Seiten erstellen, ist es sinnvoll, zuerst die Startseite mit der kompletten Navigation fertig zu stellen. (Hier spricht viel für externes CSS!)

Diese Startseite wird dann die Vorlage für alle weiteren Seiten. Hier wird dann nur noch der Inhalt ausgetauscht.

Nachteil: Will man die Navigation verändern, muss man dies in jeder einzelnen Datei tun. Mit PHP werden wir dies später verbessern.

Für die Erstellung einer verlinkten Webpräsenz mit mehreren Seiten und eigenen Inhalten empfiehlt sich ein Offline-Webseite (d.h. alle Dateien liegen in einem oder mehreren Ordnern und sind untereinander verlinkt). Alternativ kann man nach kostenlosem Webspace Ausschau halten. Diese Angebote sind aber oftmals eingeschränkt nutzbar. Auf w3schools gibt es beispielsweise eine Limitierung der Zugriffe/Monat.

Kapitel 5 - Projekt

Webseite – Bewertung	Herr Rose	Informatik 9
Name:	Thema:	

1. Allgemein

- ☐ Quelltext übersichtlich und fachgerecht eingerückt.
- ☐ „Neue“ CSS-Elemente werden kommentiert.
- ☐ Eigenleistung klar erkennbar.
- ☐ Keine störenden Darstellungsfehler („Glitches“)
- ☐ _____ Fehler und Redundanzen im Quelltext (Grundgerüst, Zeichen, Syntax)

2. Technische Umsetzung (aufeinander aufbauend, ohne direkten Notenanspruch)

ausreichend	☐ Mindestens 4 verlinkte Inhaltsseiten (funktionierende Links) ☐ Mindestens 2 weitere Elemente: Bild, Text, Tabelle, Liste
befriedigend	☐ direkte Zeichenformatierungen mit CSS ☐ globale CSS-Formatierung ☐ einheitliches Menü auf allen Seiten
gut	☐ Menü mit Dropdownfunktion ☐ CSS-Formatierungen mit Klassen oder IDs ☐ „Neue“ CSS-Elemente (nicht aus der Mappe)
sehr gut	☐ Durchgehende (globale) CSS-Formatierung mit Klassen/IDs in externer Datei ☐ komplexe „neue“ CSS-Elemente (z.B. Slideshow, Animationen, etc.) im Code kommentiert und vor der Klasse vorgestellt.

4. Inhalt

- ☐ Persönliche Beziehung zum Thema
- ☐ Eigene oder lizensfreie Fotos
- ☐ Texte, die etwas aussagen

5. Layout

- ☐ saubere Menüführung
- ☐ einheitliches Design (z.B. Farbwahl, Bilder, Schrift, wiederkehrende Elemente)
- ☐ bearbeitete Fotos (angepasste Bild- und Dateigröße)

6. Arbeitshaltung

- ☐ Zeitmanagement
- ☐ Selbständiges Arbeiten
- ☐ Nutzen von W3Schools, Codepen.io, SelfHTML und anderen Tutorials

7. Sonstiges (Besonderheiten)

- ☐

Kapitel 6 - Überleitung zu JavaScript: Eine Bildergallerie mit HTML und CSS ?

HMTL und CSS bieten nur wenige Möglichkeiten, um den Inhalt einer Seite mit der Maus oder mit der Tastatur zu verändern.

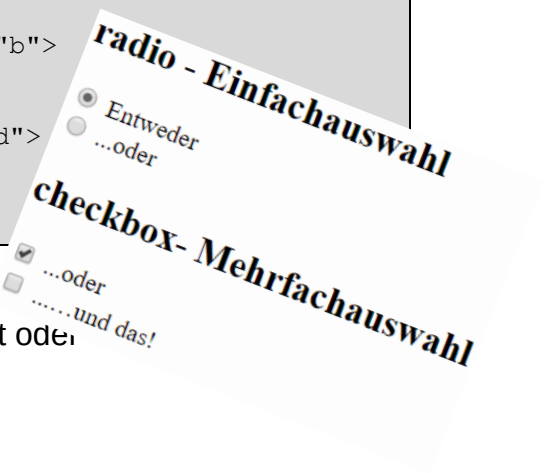
:hover Selektor (Mausposition)

Eine dieser Möglichkeiten ist der :hover Selektor. Bewegt sich die Maus über einen Link, kann ein Blockelement, dass zunächst unsichtbar war, sichtbar werden. Allein der :hover Selektor macht unser Dropdown-Menü möglich.

:checked Selektor (Mausklick)

Der :checked Selektor reagiert erst, wenn wir einen Bereich durch einen Mausklick auswählen. Er gehört zum <input> Element, welches eigentlich für Auswahlmöglichkeiten in <form> Formularen gedacht ist. Probier's aus:

```
<h2>radio - Einfachauswahl</h2>
  <input type="radio" name="Einfachauswahl" id="a" checked="checked">
  <label for="a">Entweder</label><br>
  <input type="radio" name="Einfachauswahl" id="b">
  <label for="b">...oder</label>
<h2>checkbox- Mehrfachauswahl</h2>
  <input type="checkbox" id="c" checked="checked">
  <label for="c">...oder</label><br>
  <input type="checkbox" id="d">
  <label for="d">....…und das!</label>
```



Die <label> Elemente sind duch das id-Attribut gekop

Es ist somit egal ob der User auf das <input> Element oder

auf das Text - <label> clickt.

„The checkbox hack"

Eine verbreitete Zweckentfremdung der oben beschriebenen HTML-Formularelemente ist der sogenannte „checkbox hack". Er wird beispielsweise für Bildergalerien verwendet.

Hierzu werden 3 HTML - Tags mit beliebigen "IDs und Klassen" kombiniert:

a) <label for="button">

b) <input type="checkbox" id="button" class="invisible">

c)

Mit CSS styled man nun...

a) das Label der Checkbox beispielsweise als Button oder Pfeil

b) die Checkbox unsichtbar -> display:none

c) den - Container, so dass er mit dem label verknüpft ist.

Der Pfeil ist somit ein beliebig großes, unsichtbares Kontrollkästchen. Ein Klick auf den Pfeil verändert gelichzeitig das Aussehen des -Elements, so dass hier ein neues Bild erscheint. Schau die die folgenden Beispiele an (ggfls. funktionieren müssen Links zu Bildern anderer Seiten aktualisiert werden. Viel Spaß beim klicken!

- https://codepen.io/WRose/pen/zYxemzq („Toggle“)
- https://codepen.io/WRose/pen/NWPJLyv (Bildergalerie)
- https://codepen.io/WRose/pen/NWPJLyv (Bildergalerie)

Die Bildergallerie funktioniert zwar einwandfrei, aber der Checkbox-Hack hat doch erhebliche Nachteile. Füge der Gallerie ein Bild hinzu und erläutere, warum dies ziemlich umständlich ist.

Im nächsten Skript erfährst Du, warum der Einsatz von Javascript hier mehr Sinn macht!


```
var n, bildNr = 1;
var x = document.getElementsByClassName("meineBilder");
function weiter(n)
  {zeigeBild(bildNr += n);}

function zeigeBild(n) {
  if (n > x.length) {bildNr = 1}
  if (n < 1) {bildNr = x.length}
  for (i = 0; i < x.length; i++)
  {x[i].style.display = "none";}
  x[bildNr-1].style.display = "block";
}

zeigeBild(bildNr);
```