code

<!-- -->

concept for open
digital education

CODE Skript 2016 / 2017

HTML, CSS, Javascript, PHP, MySQL

Praktischer Einstieg in die Sprachen des Web.

http://**code.arnoldbodeschule.de**

Die Daten, Editoren und erklärende Videos sind
auf unserer kostenlosen Plattform nutzbar.

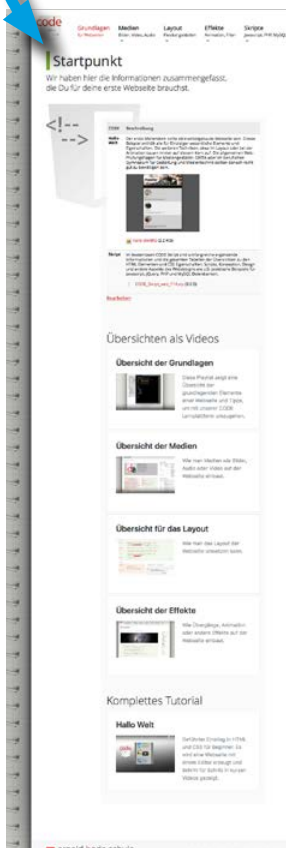
arnold bode schule
handwerk technik gestaltung

Berufliche Schule der Stadt Kassel
Schillerstr. 16
34126 Kassel

Norman Seeliger
seeliger@absks.de
Druck- und Medientechnik
Technischer Qualitätsmanagement-Beauftrager

Inhaltsverzeichnis

Reihenfolge der Beispiele

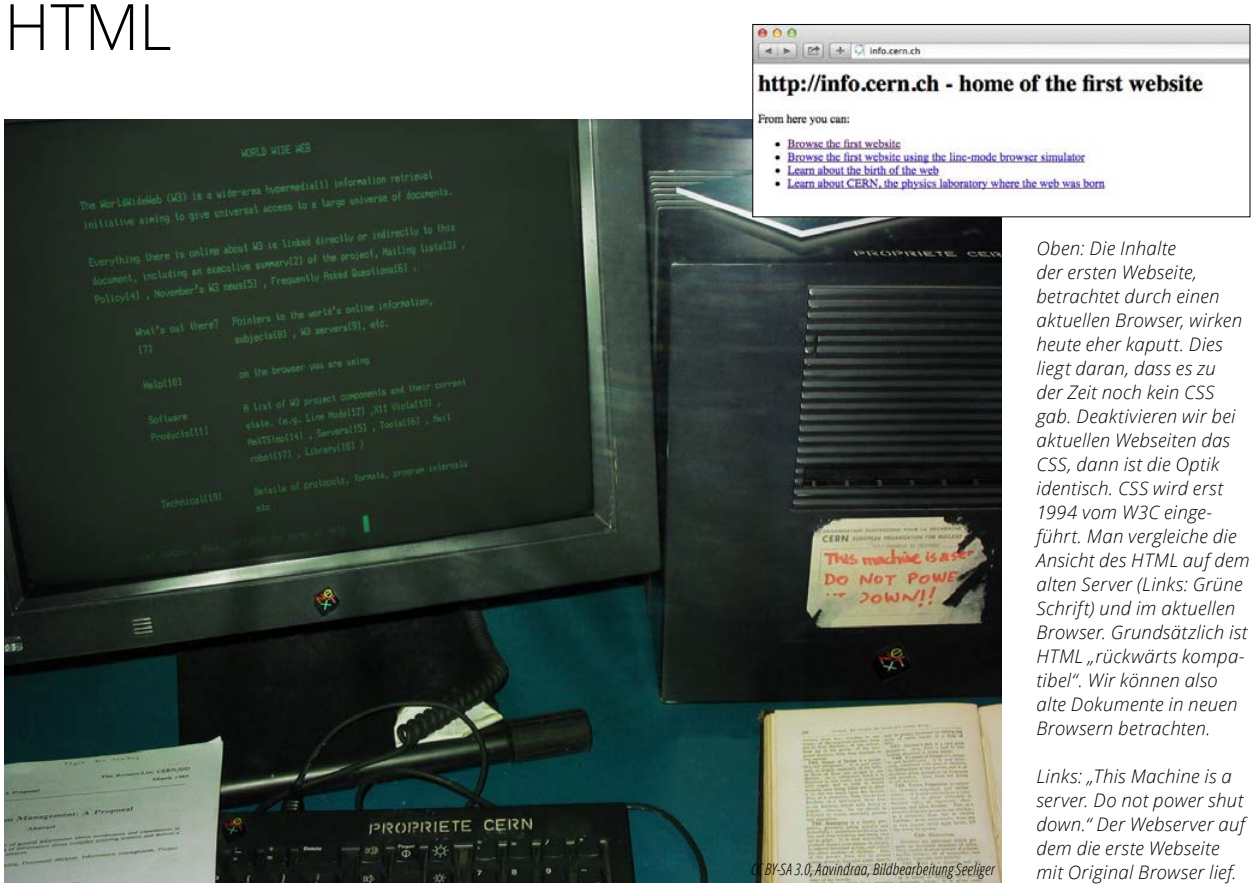


Die Beispiele bauen aufeinander auf. Man sollte „oben links" beginnen und sich dann nach „rechts unten" vorarbeiten.

code.arnoldbodeschule.de/startpunkt/

Du findest auf der CODE Plattform einen guten Anfang.

- Als ersten Meilenstein eine komplette Webseite. Die darin eingebauten Elemente und Eigenschaften sind eine sehr gute Grundlage für alles was sonst noch möglich ist. Für die allgemeinen Prüfungsfragen der Mediengestalter, GMTA und auch im Abitur für das Berufliche Gymnasium sollten diese Inhalte das Wesentliche abdecken.
- Auf den Meilenstein abgestimmte Lernkarten
- Viele erklärende Videos zu den CODE Abschnitte
- Das komplette Hallo-Welt Video-Tutorial.



Oben: Die Inhalte der ersten Webseite, betrachtet durch einen aktuellen Browser, wirken heute eher kaputt. Dies liegt daran, dass es zu der Zeit noch kein CSS gab. Deaktivieren wir bei aktuellen Webseiten das CSS, dann ist die Optik identisch. CSS wird erst 1994 vom W3C eingeführt. Man vergleiche die Ansicht des HTML auf dem alten Server (Links: Grüne Schrift) und im aktuellen Browser. Grundsätzlich ist HTML „rückwärts kompatibel". Wir können also alte Dokumente in neuen Browsern betrachten.

Links: „This Machine is a server. Do not power shut down." Der Webserver auf dem die erste Webseite mit Original Browser lief.

Die Geschichte von HTML

1989 schlug Berners-Lee seinem Arbeitgeber CERN ein Projekt vor. Hier wollte er das Prinzip der Hyperlinks nutzen um Informationen weltweit zu verlinken. Damit löste er ein bestehendes Problem, nämlich dass die Computernetzwerke des einen Standortes völlig inkompatibel zu anderen Standorten waren. Im Sommer 1991 jedoch startet die erste Webseite mit der Technologie von heute: Das Internet. Die erste Webseite beschreibt, was das World Wide Web sein sollte, wie man an einen Webbrowser kommt, was HTML und wie man einen Webserver aufsetzt. Sicherlich gibt es wesentliche Ursprünge bei Militär und Informatik, da hilft Wiki weiter.

Links: Tim-Berners Lee im Jahr 1994. Zusammen mit einem kleinen Team hat er die noch heute genutzte Web-Standards geschaffen: HTML, HTTP, HTML. Außerdem den ersten Web-Browser für HTML und einen öffentlichen Server mit Anleitung für HTML. Tim-Berner Lee ist heute Director des W3C. Quelle des Bildes: CERN. Mit freundlicher Genehmigung für die Nutzung zu Bildungszwecken.

CERN-GE-9407011-31

Zum Gebrauch der <tags> in HTML

Im Original steht auf der ersten HTML Webseite zu Tags diese immer noch gültige Anleitung:

```
HTML Tags: This is a list of tags used in the HTML language.
Each tag starts with a tag opener (a less than sign)
and ends with a tag closer (a greater than sign).
Many tags have corresponding closing tags which
identical except for a slash after the tag opener. (For example, the TITLE tag).

Some tags take parameters, called attributes.
The attributes are given after the tag, separated by spaces.
Certain attributes have an effect simply by their presence,
others are followed by an equals sign and a value. (See the Anchor tag,
for example). The names of tags and attributes are not case sensitive:
they may be in lower, upper, or mixed case with exactly the same meaning
(In this document they are generally represented in upper case.)
```

Liste der ersten HTML Elemente (<tags>)

Tag	Beschreibung
<TITLE>	Titel der Webseite im Browserfenster
<NEXTID 27>	Veraltete Info für den NEXT Browser
<a href>	Anker-Tag für Links
<ISINDEX>	Index einer Seite
<PLAINTEXT>	So gekennzeichneter Text wird unverändert angezeigt
<LISTING>	Eine Liste mit 132 Zeichen pro Zeile
<p>	Paragraph: Ein Text-Absatz
<H1>, <H2>, <H3> ...	Überschriften der Kategorie 1 bis 6
<ADDRESS>	Adresse einer Webseite
<DL><DT>	Definition-List und Definition-Term (Glossar einer Webseite)
	Ungeordnete Liste z.B. mit dem • (Punkt) als Aufzählungszeichen
	Geordnete Liste z.B. mit 1. 2. 3. oder I, II, III als Aufzählungszeichen
	Listenelement für und

a) Wann warst Du zum ersten Mal im Internet?

b) Kennst Du einige der ersten HTML Elemente?

c) Welche der ersten HTML Tags sind heute noch in Gebrauch?

d) Kannst Du den englischen Text zum Gebrauch von HTML übersetzen? (zumindest in Stichsätzen)

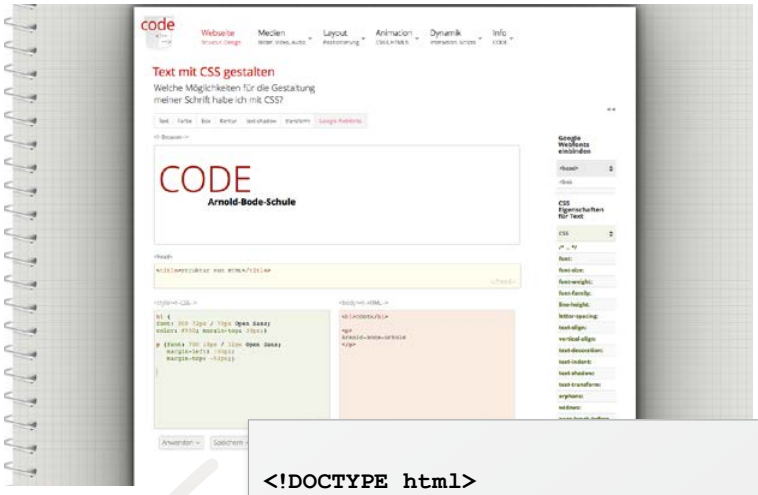
e) Markiere bei diesem Link (Anker-Element):
Den öffnenen Teil des Tags.
Das Attribut.
Den schließenden Teil des Tags.

```
<a href="hallo-welt.html„>Link</a>
```

Aufbau von HTML

HTML, mittlerweile in der Version 5, kümmert sich nur um die Inhalte. Die Gestaltung erfolgt über CSS. Prinzipiell ist ein HTML Dokument ein Textdokument mit Suffix .html und baut auf einer immer gleichen Struktur der auf: *<html>*, *<head>*, *<style>* und *<body>*.

Die möglichen HTML <tags> für *<html>*, *<head>* und *<body>* sind in der Übersichten der nächsten Seiten aufgeführt.



Speichern

```
<!DOCTYPE html>
  <head>
    <style>
      /*CSS*/
    </style>
  </head>
  <body>
    <!--HTML-->
  </body>
</html>
```

Die Struktur der HTML Dokumente ist spartanisch und wirksam. Eine handvoll Tags reichen aus für viel Action.

Bei den Beispielen auf CODE sind meist nur HTML und CSS aktiv. Die Fenster dort zeigen dann die Ausschnitte für HMTL (<body>) und CSS (<style>). Bei einigen Beispielen ist auch ein Editor für den HEAD (<head>) offen. Im Hintergrund sind die getrennten Elemente eine Einheit. Wenn Du bei den Beispielen auf Speichern klickst, dann siehst Du es.

Klickst Du bei den Beispielen auf CODE auf Speichern, dann wird der tatsächliche HTML und CSS Code angezeigt. In diesem Beispiel sind die Bereiche für head, style und body farblich markiert. Auf der Plattform bekommst Du den reinen Quellcode als Text angezeigt. Den Text kannst Du dir dann speichern. In einem Editor für Quellcode wird der Text dann farblich markiert.

```
<!DOCTYPE html>
  <head>
      <meta charset="UTF-8">
      <link href="http://fonts.googleapis.com/css
                  ?family=Open+Sans:300,700" rel="stylesheet"
                  type="text/css">

      <title>Struktur von HTML</title>

      <style>
          h1 {
          font: 300 72px / 70px Open Sans;
          color: #900; margin-top: 30px;}

          p {font: 700 18px / 32px Open Sans;
             margin-left: 100px;
             margin-top: -52px;}

      </style>

  </head>

  <body>
      <h1>CODE</h1>
      <p>Arnold-Bode-Schule</p>
  </body>

</html>
```

HTML Elemente und Attribute

Grundbestandteile von einem HTML Dokument

HTML	Bedeutung	Semantik	Beispiel
<!DOCTYPE html>	Einleitende Dokument-Deklaration (Prolog) im HTML Dokument. Markiert gemäß HTML5 Standard den folgenden Inhalt als HTML.	HTML Prolog	<!DOCTYPE html>
<body>	Enthält alle Elemente die im Browser angezeigt werden. Der BODY kann mittels CSS direkt gestaltet werden. Der BODY darf nur einmal pro HTML Dokument eingesetzt werden. Der BODY ist neben dem HEAD ein direktes Kind von dem HTML Element. Der BODY hat als Standard display: Block und nimmt 100% der Breite ein und ist so hoch wie sein Inhalt. Die Höhe muss bei Bedarf (z.B. Für einen Verlauf) im CSS definiert werden.	Körper der Webseite. Sichtbarer Bereich.	<body><main> ... </body>
<head>	Der <head> beinhaltet <title>, <style>, <style>, <meta>, <link>, <script>, <base>. Man kann in HTML5 allerdings den <head> weglassen und die genannten Inhalte einfach so vor dem <body> angeben. Wird dann aber bon dem Browser wieder sortiert und deshalb sind die gezeigten Beispiele mit HEAD.	Nichtsichtbare Informationen	<head><title> ...</head>
<html>	Das HTML-Element kommt direkt nach der <!DOCTYPE html>. Es beinhaltet HEAD und BODY sowie alle anderen Elemente. Gibt man es nicht an, dann wird es automatisch von dem Browser eingesetzt. Man kann das HTML-Element mit CSS gestalten. Das HTML-Element hat als Standard display: Block und nimmt 100% der Breite ein und ist so hoch wie sein Inhalt. Die Höhe muss bei Bedarf (z.B. Für einen Verlauf) im CSS definiert werden.	Indizier-Hilfe	<html lang="de">

Kommentare

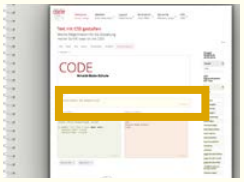
Element	Semantik	Beschreibung
<!--...-->	**HTML Kommentar**	Kommentar für HTML der im Quellcode sichtbar ist, aber nicht im Browser angezeigt wird. Wichtig für die eigene Übersicht der Arbeit. Sie können im HTML jedoch nicht im CSS genutzt werden.

```
<!DOCTYPE html>
<html lang="de">
    <head>
      <style></style>
    </head>
    <body></body>
</html>
```

<!DOCTYPE html> und </html> umschliessen (taggen) alle Elemente des HTML Dokuments.

```
<!DOCTYPE html>
<html>
<head>
  <meta charset='UTF-8'>
  <title>Basiselemente des HEAD</title>

  <link href='style.css' rel='stylesheet'>
  <style>
   h1 {font: 300 24px/16px Open Sans; letter-spacing: 2px;}
   p  {font: 300 14px/16px Open Sans;}
  </style>
</head>
```



Im <head> werden die (Meta) Informationen über die Seite gespeichert und die CSS wie Elemente. Außerdem können <scripts> geladen werden. Auf der CODE Plattform haben nicht alle Beispiele einen Editor für den <head> und damit die Möglichkeit dort Änderungen vorznehmen.

<head>

Elemente im <head>

code.arnoldbodeschule.de/**aufbau-und-inhalt-des-head/**

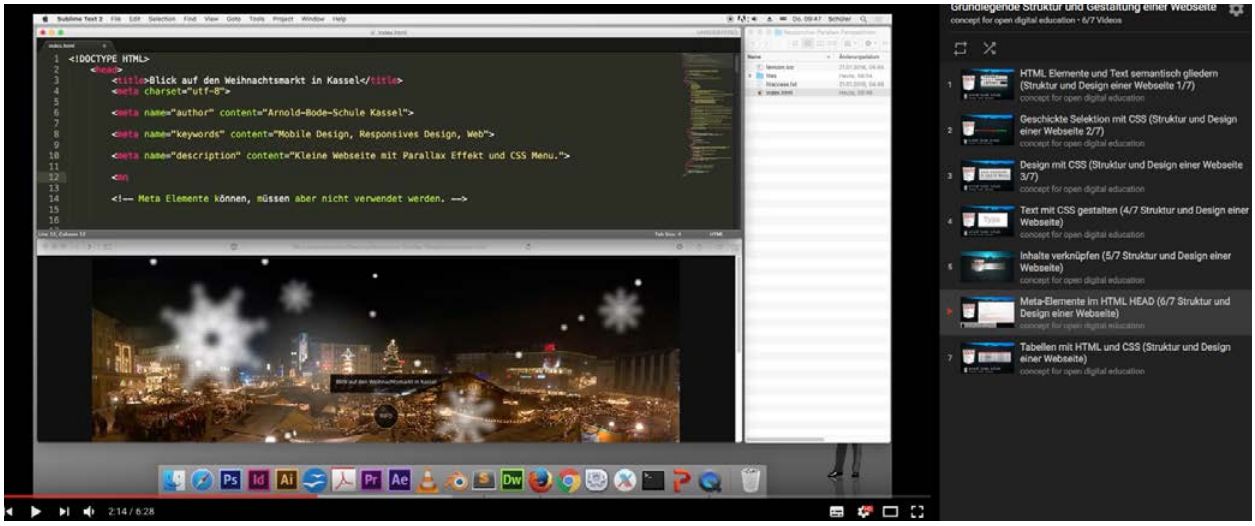
HTML	Bedeutung	Beispiel
<link>	Link auf externe Inhalte wie z.B. CSS Dokumente	`<link rel="stylesheet" type="text/css" href="formate.css">`
<meta>	Metainformationen wie Suchwörter, Zusammenfassung etc. des HTML Dokuments. Den META Elementen ist eine eigene Übersicht gewidmet.	`<meta charset="UTF-8">`
<script>	Container für client-seitigen Script und seine FUNCTIONS. Scripts liegen überwiegend in der Sprachfamilie Javascript vor. Ein weiteres populäres Beispiel für Inhalt im <script> ist das Javascript Framework jQuery. jQuery basiert auf Javascript und bietet bereits fertige FUNCTIONS. Der <script> wird je nach Bedarf im <head> der Seite oder direkt vor den </body> platziert. Das Laden von <script> sollten das erste Rendern der Seite nicht blockieren.	`<script src="jquery-1.11.3.min.js"></script>`
<style>	Im Style Bereich können die CSS-Eigenschaften des HMTL Dokuments eingetragen werden. Im <style> gilt die CSS Syntax. Für die Arbeit bei komplexeren Dokumenten ist der Verweis mittels <link src="name.css"> praktischer. (Minimierte)CSS Angaben im <style> sind bei HTTP und HTTPS jedoch schneller zu laden.	`<style> h1 {color: red;}`
<title>	Titel der Seite. Wird im TAB und der Titelleiste des Browsers angezeigt. Erscheint jedoch nicht im <body> der Webseite. Wird der <title> auf der Webseite und nicht im TAB angezeigt, dann hat man meist irgendwo einen Tippfehler.	`<title>Hallo Welt</title>`

Attribute für <meta>

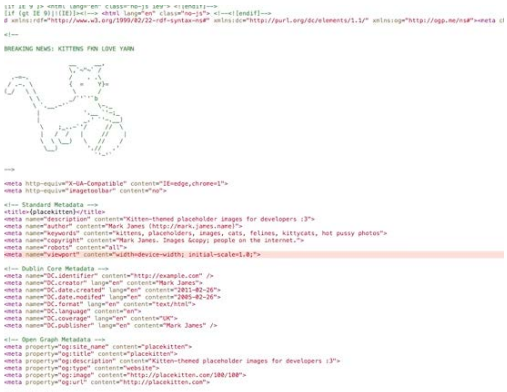
Attribut	Bedeutung	Beispiel
"revisit-after" content = "5 days"	Lädt die Suchmaschine ein, nach 5 Tagen erneut die Seite zu indizieren. Wobei Suchmaschinen meist nach eigenen Kriterien vorgehen.	`<meta "revisit-after" content="5 days">`
charset = "UTF-8"	Zeichencodierung der Webseite. Sehr wichtige Angabe für die korrekte Darstellung der Umlaute wie ä oder ü. UTF-8 ist für die meisten Webseiten zu empfehlen.	`<meta charset="UTF-8">`
content = "..."	Der eigentliche Wert bei Metadaten oder die Pragma-Directive bei http-equiv. z.B. bei description die Zusammenfassung der Seite oder bei Stichworten das konkrete Stichwort. Das wichtige Attribut content=" ... " ergänzt viele andere Meta-Attribute wie name, description, author etc.	`<meta name="keyword" content='Wolken'>`
Dublin Core Metadate	Alte Sammlung von Metainformationen, die sich nicht wirklich durchgesetzt haben. Dabei wird zu DC.identifier, DC.creator, DC.date.created, DC.date.modifed, DC.format, DC.coverage, DC.publisher mit content = Angaben gemacht.	`<meta name="DC.creator" lang="de" content="Norman Seeliger">`
href="..."	Angabe auf welche Datei mittels relativem oder absoluten Pfad verlinkt wird.	`<a href="hallo.html">Hallo Welt</a>`
http-equiv =	http-equiv ist die Kurzschreibweise für "Gleich-bedeutend mit dem HTTP-Header-Feld. Sie sind "Pragma-Directiven". Sie legen fest wie der Besucher die Seite verarbeitet. Genutzt werden können pragma, content-type, default-style und refresh (Siehe jeweilige Karte). In HTML5 sind set-cookie und content-language kein Standard mehr, daher gibt es für diese 2 auch keine Karte.	`<meta http-equiv="pragma" content="no-cache">`
http-equiv = "content-type" content="utf-8"	Hat die gleichen Auswirkungen wie charset="utf-8". Sollte nur benutzt werden wenn XHTML immigriert wird.	`<meta http-equiv="content-type" content="utf-8">`
http-equiv = "imagetoolbar" content = "false"	Verhindert im Internetexplorer das Rechtsklick-Kontext Menu um z.B. Bilder zu speichern	`<meta http-equiv = "image-toolbar" content = "false">`
http-equiv = "pragma" content = "no-cache"	Der Zusatz content="no-cache" verhindert dass die Seite auf Proxy-Servern in den Cache gespeichert wird.	`<meta http-equiv="pragma" content="no-cache">`
http-equiv = "refresh" content = "5; URL = http:...">	http-equiv="Refresh" veranlasst den Browser die Seite oder eine andere Seite neu zu laden. Mit content="5" z.B. wird nach 5 Sekunden das HTML Dokument neu geladen. Ist mit dem URL= eine andere Seite bestimmt, dann wird diese nach der Zeit geladen.	`http-equiv="refresh" content="5; URL=http://code.arnoldbodeschule.de/">`
http-equiv = "refresh" content = "600"	http-equiv="Refresh" ohne URL veranlasst den Browser diese Seite neu zu laden. Mit content="600" z.B. wird alle 600 Sekunden das HTML Dokument neu vom Server geladen. Es gibt heutzutage mit Javascript und AJAX elegantere Möglichkeiten aktuellen Content auf der Seite zu haben. Dieses zwanghafte Laden über http-equiv = "refresh" kann Benutzer irritieren und z.B. halbausgefüllte Formulare auf Null setzen.	`<meta http-equiv="Refresh" content="600">`
http-equiv = "X-UA-Compatible" content="..."	Der Kompatibilitätsmodus des Internet Explorers. Hier kann definiert werden, welche Version des IE unterstützt wird. Spielt für Webdesigner eine größere Rolle, die auch ältere Versionen des IE unterstützen müssen.	`<meta http-equiv="X-UA-Compa-tible" content="IE=8" />`
media="screen"	(orientation: portrait) and (min-width: 1px) Screen, Print etc.	`<link >`
name = "..."	Es können eigene Gruppen für Metadaten angelegt werden. Als Name können alle beliebigen Metainformationen gespeichert werden. Als Standard-Meta-Namen gelten: author, generator, keywords, description. Zu dem Attribut name gehört immer der Wert der mit dem Attribut content= eingeleitet wird.	`<meta name="keyword" content='Wolken'>`

Attribut	Bedeutung	Beispiel
name = "author" content="..."	Angabe zu dem Autor des Dokuments.	`<meta name="author" content='Hans Maulwurf'>`
name = "date" content = "2016-02-01"	Angaben der letzten Änderung. Dies sollte automatisiert geschehen, da es sonst leicht vergessen wird.	`<meta name="date" content="2016-05-15T08:49:37+02:00">`
name = "description" content="..."	Zusammenfassung der Seite, z.B. bei der Google Suche	`<meta name="description" content='Webseite von Döner24'>`
name = "google" content="no sitelinkssearchbox"	Verbietet Google ein Suchfeld direkt für die Webseite anzuzeigen in der Google Suche anzuzeigen.	`<meta name= "google" content= "nositelinkssearchbox">`
name = "keywords" content= "..."	Stichwörter für die Suchmaschine. Aufgrund des enormen Missbrauchs in der Vergangenheit werden diese heute oft ignoriert. Echter Content ist King.	`<meta name="keyword" content='Wolken, Natur'>`
name = "robots" content = "index, follow"	Legt fest ob Suchmaschinenroboter die Seite indizieren sollen. Index: Seite indixieren. NoIndex: Seite nicht indixieren. Follow: Links folgen. NoFollow: Links nicht folgen. Index, NoFollow: Seite indixieren, Links nicht folgen. NoIndex, Follow: Seite nicht indixieren, Links folgen All: Seite indizieren, Links folgen. Wenn man möchte dass die Seite und alle ihre Links indiziert wird wählt man die mit Komma getrennte Kombination aus index und follow: "robots" content="index, follow".	`<meta name="robots" content="index, follow">`
name = "viewport" content = "minimum-scale: 1, maximum-scale: 3">	Bestimmt die möglichen Vergrößerungsstufen beim Zoomen des Nutzers. Diese Kontrolle, laut Nutzer-Tests, beeinträchtigt die Zugänglichkeit und sollte im Allgemeinen vermieden werden.	`<meta name="viewport" content="minimum-scale: 1, maximum-scale: 3">`
name = "viewport" content = "user-scalable: no">	Der Viewport enthält wesentliche Angaben für mobile Seiten. Man könnte z.B. das Zoomen mit user-scalable: no verbieten. Dies ist allerdings meist nicht zu empfehlen, da es für eine viele Nutzer eine unangenehme Einschränkung darstellt. user-scalable könnte mit width=device-width, initial-scale und minimum-scale bzw. maximum-scale: in einen meta Tag zusammgenfasst werden.	`<meta name="viewport" content="user-scalable: no">`
name = "viewport" content = "width = 1200px"	Man kann den Viewport strikt definieren. Dies dürfte bei den meisten Anwendungen jedoch erhebliche Nachteile mit sich führen. Es passt nämlich nur selten. Die flexible Angabe mit content="width=device-width ist daher zu empfehlen.	`<meta name="viewport" content="width=1200px">`
name = "viewport" content = "width = device-width, initial-scale = 1.0"	Praktischer Hinweis für mobile Browser. Die Breite wird auf die Ausgabebreite des jeweiligen Gerätes gesetzt und die Zoomstufe bei der ersten Ansicht ist 1. Durch das Komma im Content werden zwei Angaben, zu width und dem initial-zoom, zusammengefasst. Wichtig ist dabei width = device-width. Nur damit ist die Webseite flexibel und geräteunabhängig. Durch die definierte Zoomstufe auf 1 ist die Schriftgröße auf dem mobilen Gerät gleich groß, egal ob im Hochformat (Porträt) oder Querformat (Landscape) gehalten. Das Drehen des Gerät würde ansonsten meistens zu einer Änderung der Optik führen. Wobei man allerdings diesen Effekt durch definierte @media Angaben im CSS auch ausschalten kann. Entscheidend ist width = device-width.	`<meta name="viewport" content="width=device-width, initial-scale=1.0">`
name = generator content="..."	Angabe des Editors der für die Erstellung genutzt wurde. Manche Software schreibt sich so in den head.	`<meta name=generator content="Adobe Dreamweaver">`
property= "og:description" content=" ... "	Zusammenfassung der Seite. Bestandteil des Open Graph Standards. Wird z.B. von Google + und Facebook genutzt.	`<meta property= "og:description" content="Zusammenfassung für den Open Graph Standard" />`
property= "og:image" content=" http:// ... "	Vorschaubild der Seite. Bestandteil des Open Graph Standards. Wird z.B. von Google + und Facebook genutzt.	`<meta property= "og:image" content=" http://http://code.arnold-bodeschule.de/preview.jpg" />`

Attribut	Bedeutung	Beispiel
property= “og:title“ content=“ ...“	Titel der Seite in der Vorschau von Social Media Diensten. Bestandteil des Open Graph Standards. Wird z.B. von Google + und Facebook genutzt.	`<meta property= “og:title“ content=“ CODE“ />`
property= “og:type“ content=“website“	Bestandteil des Open Graph Standards. Wird z.B. von Google + und Facebook genutzt.	`<meta property= “og:type“ content=“website“ />`
property= “og:url“ content=“http://...“	Link zu der Seite für Social Media Dienste. Bestandteil des Open Graph Standards. Wird z.B. von Google + und Facebook genutzt.	`<meta property= “og:url“ content=“http://code.arnold-bodeschule.de“ />`
rel="stylesheet"	Zeigt die Beziehung zu der mit href="..." eingeleiteten Quelle.	`<link rel="stylesheet" type="text/css" href="design.css">`
Twitter Cards	Vergleichbar zum Open Graph Standard hat Twitter ein Set von eigenen Meta Daten, die Twitter Cards. twitter:card, twitter:site, twitter:creator, twitter:title, twitter:description, twitter:image:src	`<meta name="twitter:card" content="Zusammenfassung der Seite auf einer Twitter Card" />`
type="text/css"	Typ des Mediums. Z.B. ein externes CSS um mehrere HTML Dokumente mit einer CSS Definition zu versorgen. Liste aller Typen ist ist bei iana.org definiert. ACHTUNG: Bei Javascript und bei CSS muss der Type in HTML 5 nicht mehr angegeben werden.	`<link rel="stylesheet" type="text/css" href="design.css">`



Meta-Elemente im HTML HEAD (6/7 Struktur und Design einer Webseite)
Das Meta Element dient für die weitere Auszeichnung der Webseite. Es können für die Suchmaschine relevante Angaben wie eine Zusammenfassung oder über den Autor gemacht werden. Somit spielen sie für „Search-Engine-Optimazation“ (SEO) eine Rolle. Für Social Dienste wie Facebook gibt es mit dem Open Graph Standard noch mehr Möglichkeiten. Mit dem META Element kann man auch Inhalte in HTTP Header Feldern überschreiben und so z.B. eine Weiterleitung für die Webseite realisieren. Diese Meta-Informationen kann man auch nutzen um z.B. ein Menu zu realisieren.



Elemente im <body>

Strukturelle Elemente

code.arnoldbodeschule.de/**elemente-semantisch-gliedern-und-selektieren/**

HTML	Bedeutung	Semantik	Beispiel
<article>	Umfasst Elemente die zu einem Artikel gehören. Z.B. Überschriften, Text, Tabellen und Bilder.	**Artikel**	`<article><h1> ...</article>`
<aside>	Umfasst ergänzende Elemente. ASIDE bedeutet "Nebenbemerkung". Verhält sich optisch wie alle anderen Block-Elemente. Muss also mit CSS an seine Position gebracht werden.	**Nebenbemerkung**	`<aside>Ach übrigens ...</ aside>`
** **	DIV ist die Abkürzung für Divison: Bereich. Er ist semantisch nicht näher definiert. Er kann beliebige Inhalte haben kann. DIV sind häufig benutzte Elemente. Allerdings sollten zunächst die semantischen Elemente wie <section> <article>, <nav> <button> etc. benutzen.	**Beliebiger Block-Bereich**	`<div>Beliebige Inhalte</div>`
<embed>	Eingebettete Application. Wie z.B. ein Flash oder auch Videos von Youtube oder Vimeo. Wobei der Trend in Richtung <iframe> geht.	**Eingebetteter Code**	`<embed src="helloworld. swf">`
<figure>	Container Element für alle Inhalte (FIGCAPTION) die zu einer Abbildung (IMG) gehören.	**Abbildung**	`<figure> <figcaption> ....<img href="... </figure>`
<footer>	Fußbereich der Webseite für dort geeignete Informationen.	**Abschluss**	`<footer>Imressum</ footer>`
<header>	Sichtbarer HEADER (Kopfbereich) der Webseite. Nicht mit dem unsichtbaren HEAD zu verwechseln.	**Kopfbereich**	`<header>Logo</ header>`
<iframe>	Bettet einen Bereich aus einer anderen Quelle, z.B. HTML, Webseite, Google-Maps oder Videohoster wie Youtube, ein.	**Intelligenter Rahmen**	`<iframe src="katzen. htm"></iframe>`
<main>	Hauptteil der Webseite. Darf in einem validen HTML Dokument nur einmal vorkommen und kein Kind-Element der anderen Elemente, bis auf HTML und BODY, sein. Leider wird <main> NICHT vom Internet Explorer unterstützt. Unglaublich. Insofern ist davon in der Produktion abzuraten.	**Hauptbereich**	`<main><nav> ...</main>`
<nav>	Navigation der Webseite. Enthält automatisch die ARIA-Roles die einem Screenreader die Navigation signalisieren.	**Navigation**	`<nav><a href="1. html">Menu</ a></nav>`
<script>	Container für client-seitigen Script und seine FUNCTIONS. Scripts liegen überwiegend in der Sprachfamilie Javascript vor. Ein weiteres populäres Beispiel für Inhalt im <script> ist das Javascript Framework jQuery. jQuery basiert auf Javascript und bietet bereits fertige FUNCTIONS. Der <script> wird je nach Bedarf im <head> der Seite oder direkt vor den </body> platziert. Das Laden von <script> sollten das erste Rendern der Seite nicht blockieren.	**Eingebetteter Script**	`<script src="jquery- 1.11.3.min. js"></script>`
<section>	Abschnitt (Sektion) der Webseite. Man kann mehrere SECTIONS in einem HTML Dokument benutzen. Eine SECTION kann weitere SECTION als Kind haben. Eine SECTION umfasst mehrere <article>. Die SECTION ist sozusagen die Kategorie.	**Sektion**	`<section> <article></section`

```
<section class='wrapper'>
  <header>
    <h1>Header</h1>
  </header>
  <nav class="black--box">Navigation</nav>
  <section>
    <article>
      <h2 id="art1">Article 1</h2>
    </article>
    <article>
      <h2>Article 2</h2>
    </article>
  </section>
  <aside>
      <small>Aside: Ergänzende Informationen</small>
  </aside>
  <footer>
      <address>Footer: Adresse, Impressum, Sitemap</address>
  </footer>
</section>
```

Alle HTML-Elemente werden in den <body> der Seite geschrieben. Der große Teil der <tags> hat auch einen schließenden Teil </tag>.

Jedes HTML Element hat eine definierte Optik im CSS die aktiv wird, so lange wir keine eigene CSS Eigenschaften dem Element zuweisen.

Neben ihren Kern-Funktionen und Inhalten können die HTML-Elemente weitere Attribute erhalten.

Die allgemeinen HTML Attribute können praktisch allen HTML Elementen der unterschiedlichen Arten wie textbezogen, oder flächenbezogen zugeordnet werden.

Allgemeine Attribute für Elemente im <body>

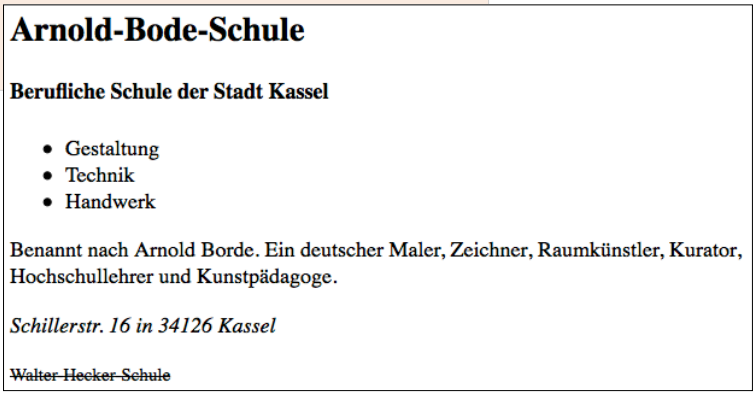
Attribut	Bedeutung	Semantik	Beispiel
id="..."	ID definiert und identifiziert ein Element exakt und kann deswegen auf der Webseite nur einmal benutzt werden. Kann auf jedes Element angewendet werden. ID können direkt verlinkt werden.	**Identification (ID)**	`<p id="start"> Startpunkt </p>`
class="..."	Klasse definiert Eigenschaften die mehrfach genutzt werden. Kann auf jedes Element angewendet werden. Es können auch mehrere Klassen gleichzeitig bei einem Element zugeordnet sein. Gestaltung sollte über Klassen und nicht über ID umgesetzt werden.	**Klasse (class)**	`<p class="box red shadow"> Start-punkt </p>`
style ="..."	Alle CSS Eigenschaften können auch direkt einem Element als Attribut zugeordnet werden. Dies sollte man jedoch zugunsten einer globalen Gestaltung mit einer CSS möglichst vermeiden. Einige Javascript Effekte, z. B. Parallax Effekte, basieren jedoch auf der Echtzeit-Veränderung des einzelnen STYLE Attribut.	**Style**	`<p style="color: red"> Roter Text </p>`
src="..."	Gibt eine Quelle, z.B. bei einem Bild an	**Quelle (Source)**	`<img src="kitten.jpg">`
title ="..."	Kleine Info-Tex-Box die als Mouse-Over angezeigt wird	**Mouse-Over-Information**	`<a title="navigation" href="hallo.html">Hallo Welt</a>`

Textbezogene Elemente

code.arnoldbodeschule.de/**elemente-semantisch-gliedern-und-selektieren/**

HTML	Bedeutung	Semantic / Übersetzung	CSS-Standard für Display:
<adress>	Markiert die Kontakt Informationen.	Adress Adresse	block
<b>	Fetter Text. Wichtiger Text sollte mit hervorgehoben werden. Die Optik wird mit der CSS Eigenschaft FONT-WEIGHT (z.B. Bold) bestimmt.	bold fett	inline
** **	BREAK. Zeilenumbruch. Der Text im Absatz bricht an dieser Stelle um.	Break Zeilenumbruch	block
<cite>	Zitat.	Cite Zitat	inline
<dd>	Erklärender Text in der Definitionsliste. Entspricht in der Verwendung dem	Definition Definition Element der Definitionsliste	block
<del>	Durchgestrichen.	Delete gelöscht	inline
<dl>	Anfang und das Ende einer Definitionsliste. Ähnelt definiert jedoch mittels <dt> und <ddl> ein konkretes Element.	Definition List Definitionsliste	block
<dt>	Zu definierender Begriff in der Definitionsliste.	Definition Tag Definitionsobjekt	block
****	Hervorgehobener Text. Steuert im semantischen Kontext die Bedeutung.	emphasis Hervorgehoben	inline
<h1> <h6>	Überschriften der 1. bis 6. Ebene. H1 (Head1) sollte pro Dokument nur einmal vorkommen.	Head Überschrift	block
<hr>	Horizontale Linie für einen thematischen Wechsel. Man kann übrigens selbst die HR mit Angaben für eine Höhe in ein Rechteck umbauen. Eine HR kann aber kein Text beinhalten. Sie ist zum gliedern von Text gedacht.	Horizontal Rule Horizontale Linie	block
<i>	Zeigt es im semantischen Kontext Fremdwörter an. Die Optik wird mit der CSS Eigenschaft FONT-FAMILY oder FONT-STYLE bestimmt.	italic italienisch	inline
<ins>	Neu eingefügter und deswegen hervorgehobener Text.	insertet eingefügt	inline
** **	Markiert in und die einzelnen Listen-Elemente.	List-Element Listen Element	block
** **	Anfang und das Ende einer nummerierten Liste. Per CSS-Standard bekommen LI Elemente in einer OL ein laufende Nummer.	Ordererd List Nummerierte Liste	block
<p>	Kompletter Absatz (Paragraph) im Text. Entspricht einem Absatzformat im InDesign	Paragraph Absatz	block
<q>	Container für ein Zitat (Quote) samt <cite>.	Quote Zitat-Container	inline
<small>	Kleine Schriftgröße. Wird mit CSS FONT-SIZE näher eingestellt.	small klein	inline
****	Fetter Text. Im semantischen Kontext betont es Wichtigkeit. Die Optik wird mit der CSS Eigenschaft FONT-WEIGHT (z.B. Bold) bestimmt.	strong stark	inline
<sub>	Tiefgestellter Text.	subscript tiefstellen	inline
<sup>	Hochgestellter Text.	superscript hochstellen	inline
** **	Anfang und das Ende einer Liste mit beliebigen Aufzählungszeichen. UL steht für Unordered List, also eine Liste ohne Zahlen. Per CSS-Standard bekommen LI Elemente in einer UL einen Aufzählungspunkt.	Unordered List Ungeordnete Liste (Aufzählungszeichen)	block
<var>	Kennzeichnet semantisch eine Variable. HTML kann aber keine Variablen ausführen. Es bezieht sich nur auf die Dokumentation von Variablen.	variable Variable	inline

```html
<!-- markup mit HTML -->
<article>
  <h1>Arnold-Bode-Schule</h1>
  <h4>Berufliche Schule der Stadt <strong>Kassel</strong></h4>
  <ul>
    <li>Gestaltung</li>
    <li>Technik</li>
    <li>Handwerk</li>
  </ul>
  <p>Benannt nach Arnold Borde. Ein deutscher Maler, Zeichner, Raumkünstler,
Kurator, Hochschullehrer und Kunstpädagoge.</p>
  <p><address>Schillerstr. 16 in 34126 Kassel</address></p>
  <p><small><del>Walter-Hecker-Schule</del></small></p>
</article>
```



Alle HTML-Elemente bringen von Haus aus ein Standard-Aussehen mit. Diese Standard-CSS Eigenschaften sind aktiv, so lange man keine eigenen Angaben in der CSS macht. Während die flächenhaften HTML-Elemente (<main>, <nav> oder etc.) identisch aussehen, unterscheiden sich die textbezogenen Elemente in der Optik auffälliger.

Es ist praktisch immer notwendig diese Standard-Optik an die eigene Gestaltung mit CSS anzupassen.

```css
/* style mit CSS */

* {
  font-family: arial;}

h1, h4 strong {
  font-size: 24px;}

strong {
  color: red;
  font-weight: bold;}

small {
  font-size: 9px}

p {
  font-size: 12px;
  max-width: 350px;}
```

Anchor (Anker) Element

<a>

code.arnoldbodeschule.de/**inhalte-verlinken/**

HTML	Bedeutung	Semantik	Beispiel
<a>	Anker Element zum Verknüpfen von Inhalten. Das Verknüpfen geschieht über das Attribut href="..." (Hypertext-Referenz). Das Element was mit <a>ELEMENT getagt wurde fungiert dann als Link. Als Link können praktisch alle Elemente dienen.	**Link (Anker)**	`<a class="menu-link" href="Katze.html">Test</a>`

```
<a href='http://www.arnoldbodeschule.de' target='_blank'>Arnold-Bode-Schule</a>
</body>
```

HTML Attribute für <a>

Attribut	Bedeutung	Übersetzung	Beispiel
href=""	Link (Hypertextreferenz) mit einem absoluten oder relativen Pfad zu einer URL (Domain, Internetadresse), Datei (.CSS, .JPG, .PDF etc,) oder mit einer ID gekennzeichneten Inhaltspunkt in dem HTML Dokument (Siehe Inhaltsverzeichnis bei Wikipedia.). Absolute Pfade sind vom Standort des eigenen Dokument unabhängig. Relative Pfade sind abhängig. Relative Pfade wie z. B. "css/mobile.css" sind oftmals flexibler (gehe in den Ordner css und lade dort mobile.css). Letztlich werden beide Pfadkonzepte benötigt. Das Protokoll der Pfade kann neben HTTP z.B. auch FTP://, MAILTO:// oder HTTPS:// sein.	**Hypertextreferenz**	`<img href="1a.jpg">`
target=""	target="_blank" öffnet den Link in einem neuen Browser-Fenster. Das Attribut "_self" öffnet den Link im gleichen Browser-Fenster.	**Ziel**	`<a href="hallo.html" target="_blank">Hallo</a>`
title=""	Mouseover Information mit Angaben zu dem Inhalt. Dies spielt auch bei Suchmaschinen eine Rolle. Wobei die Suchmaschine den Verbund aus Dateiname, TITLE, ALT und umgebenden Text interpretiert.	**Tool-Tip**	`<img title="Blauer Himmel in den Alpen" ...`

Arnold-Bode-Schule

Bei der Gestaltung von <a> sind auch besondere :pseudo-classes, z.B. Mouse-Hover, sehr interessant. Schau Dir dazu am Besten das Kapitel „Inhalte verknüpfen" und die Beispiele auf CODE an.

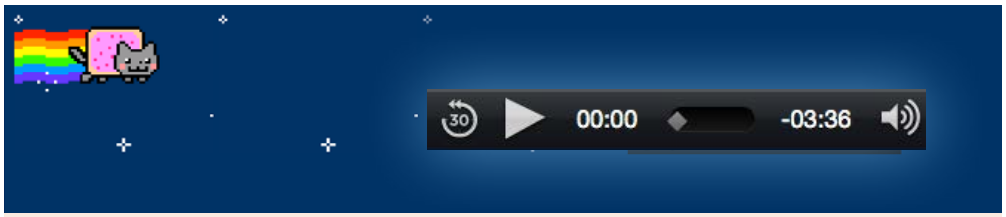
Grundsätzlich können alle denkbaren HTML Elemente, z.B. Bilder, zum Link werden, indem man sie mit <a> taggt.

Briefmarke Arnold Bode link.tiff, Gemeingut (Public Domain), CC-0

Medienbezogene Elemente

code.arnoldbodeschule.de**/bilder/**
code.arnoldbodeschule.de**/video/**
code.arnoldbodeschule.de**/audio/**

HTML	Typische Attribute	Bedeutung	Beispiel
<img>	**title=" Titel" alt=" Alternativtext" class="img-big"**	Lädt ein Bild und bietet dafür weitere Funktionen die über die Attribute zugewiesen werden. IMG steht für Image. Die wesentlichen Attribute sind: SRC, TITLE, ALT und CLASS.	**<img src="kitten.jpg">**
<video>	**controls autoplay**	Lädt ein Video und bietet dafür weitere Funktionen. Es ist der Container für die mit SOURCE angegebenen Dateien. Mit CLASS und CSS kann VIDEO Element als Box gestaltet werden. Zusätzlich kann die Kontrollleiste (controls) eingeblendet werden. Die Optik dieser Kontrollleiste ist vom Browser abhängig. Sie kann mit CSS und Javascript angepasst werden.	**<video><source ...>**
<audio>	**controls loop autoplay**	Lädt ein Audio und bietet dafür weitere Funktionen. Es ist der Container für die mit SOURCE angegebenen Dateien. Das <audio> ist unsichtbar und kann nicht mit CSS gestaltet werden. Die Kontrollleiste wird auf Wunsch angezeigt.	**<audio><source ...>**
<picture>		Container für Respnsive Bilder	
<source>	**src="music.mp3" type="audio/mpeg"**	Quelle für Audio-Dateien im <audio>.	**<source src="banzai.mp3" type="audio/mpeg">**
<source>	**src="movie.mp4" type="video/mpeg"**	Quelle für Video-Dateien im <video>	**<source src="lol.mp4" type="video/mpeg"**




```
<IMG SRC='bilder/NyanCat.gif'>
<IMG SRC='bilder/NyanCatStars.gif'>
<IMG SRC='bilder/NyanCatStars.gif' style='Opacity: 0.5;'>
<audio controls loop>
  <source src='audio/NyanCatoriginal.mp3' type='audio/mpeg'><!--webkit,moz,ie -->
  <source src='audio/NyanCatoriginal.ogg' type='audio/ogg'> <!--opera-->
</audio>
```

Aufgrund von Lizenzbestimmungen spielen nicht alle Browser die <source> im <video> und <audio> ab. Es ist besser dort noch Lizenzfreie Open Source Alternativen anzubieten.

Bei Bildern ist es hingegen unkritisch. JPEG (Fotos), PNG (Freigestelle Bilder), GIF (Grafiken und Animation) oder SVG (Vektor, Animation) sind werden flächendeckend unterstützt.

Ber Nyan-Cat wird die Audio-Control angezeigt weil das entsprechende Attribut angegeben ist, bei dem Video nicht.
Es ist auch ratsam zu prüfen ob ein ungefragtes abspielen von Musik für die Benutzer angenehm ist.


```
<video autoplay loop
   title='Spektatkuläres Mehl'>

   <source src='video/Parti.mp4'
           type='video/mp4'>

   <source src='video/Parti.ogv'
           type='video/ogg'>

</video>

<main>Particles #14</main>
```

```
video {width: 100%; height: auto;}

main { width: 280px; height: 80px;
       left: 50%; margin-left: -140px;
       background:
         rgba(5,100,5,0.5);
       position: absolute; top: 20px;
       text-align: center;
       border: 1px solid #aaa;
       color: #fff;
       font: 300 32px/70px Open Sans;}

body { background: #222;
       Overflow: hidden;}
```

Nyan Cat, Pop Tart Cat animation (Nyan Cat), Illustrator Chris Torres (prguitarman), Music von DANIWELL-AIDN, Video von SARAJOON

Particles von Andrew Cramer, Frei verwendbar, Particle Tutorial

Medienbezogene Attribute

Attribut	Bedeutung	Übersetzung	Beispiel
controls	Steuerfeld des Browsers. Die Optik der Kontroll-leiste unterscheided sich je nach Browser.	Steuerung	`<audio controls> ...`
autoplay	Startet automatisch. Es ist ratsam zu prüfen ob ein Autoplay für die Besucher angenehm ist.	Automatischer Start	`<audio autoplay> ...`
loop	Automatische Wiederholung,	Wiederholung (Schleife)	`<audio loop autoplay> ...`
src: "dateiname.mp3" type="audio/mpeg"	Dateityp MP3. Funktioniert in allen Browsern.	Audio-Quelle	`<source src='audio/ NyanCat.mp3'>`
src: "video.mp4"	Verknüpft ein MP4 Video im Source Tag.	Video-Quelle	`<source src='bloop.mp4'>`
type="video/mp4"	Definiert den Type der Source, z.B. MP4. Mp4 erhebt Lizenzgebühren für die Browserhersteller und wird deshalb mitunter nicht Supporten.	Video-Typ MP4	`<source src='bloop.mp4' type="video/mp4">`
type="video/ogg"	Definiert den Type der Source, z.B. OGG Vorbis für Firefox. OGG erhebt keine Lizenzgebühren.	Video-Typ OGG	`<source src='bloop.ogg' type="video/ogg">`
src: "dateiname.ogg" type="audio/ogg"	Dateityp OGG funktioniert in Firefox, Chrome und Opera	Quelle	`<source src='bloop.ogg'>`
src: "dateiname.wav" type="audio/wav"	Dateityp Wave. Funktioniert in Safari, Firefox, Chrome und Opera. Erzeugt große Dateien	Quelle	`<source src='bing. wav'>`
preload: "auto" "metadata" "none"	Bestimmt ob das Medium vorab geladen wird.	Vorab-Laden	`<audio preload: "auto"> ...`
muted	Stumme Ausgabe.	Tonlos	`<video muted> ...`
src=""	Pfadangabe für das dargestellte Element. Die SRC (Source, Quellenangabe) kann mit einem absoluten oder relativen Pfad zu der Datei (z.B. .MP3, .OGG, .MP4 etc,) verweisen. Absolute Pfade sind vom Standort des eigenen Dokument unabhängig. Relative Pfade sind abhängig. Relative Pfade wie z. B. "sound/beep.mp3" sind oftmals flexibler (gehe in den Ordner sound und nimm dort beep.mp3). Letztlich werden beide Pfadkonzepte benötigt. Das Protokoll der Pfade kann neben HTTP z.B. auch HTTPS:// sein.	Quelle	`<source src='bloop.ogg'>`
alt=""	Alternativ Text der den Inhalt besch-reibt, z.B. für einen Blinden-Screen-reader oder als Fallback-Text, wenn ein Medium nicht geladen werden kann.	Alternativ Text	`<img alt="Blauer Himmel" ...`
title=""	Mouseover Information mit Angaben zu dem Inhalt. Dies spielt auch bei Suchmaschinen eine Rolle. Wobei die Suchmaschine den Verbund aus Dateiname, TITLE, ALT und umgebenden Text interpretiert.	Tool-Tip	`<img title="Blauer Himmel in den Alpen" ...`
width=""	Breite des Elements. Man sollte über CSS, z.B. mit einer CLASS, das Element gestalten. Die Verwendung des Attributs width ist nicht zu empfehlen.	Breite	`<img width="100px" href="1a.jpg">`
height=""	Höhe des Elements. Man sollte über CSS, z.B. mit einer Class, das Element gestalten. Die Verwendung des Attributs height ist nicht zu empfehlen.	Höhe	`<img height="100px" href="1a.jpg">`
ismap=""	Definiert das Bild als serverseitige Image-Map. Koordinaten werden als URL übergeben und funktioniert nur in Kombination mit <a>	Imagemap	`<img ismap>`
usemap=""	Definiert das Bild als clienseitige Image-Map	Imagemap	`<img src="modor. png" usemap="#map">`

Tabellenbezogene Elemente

Tabelle

code.arnoldbodeschule.de/**tabellen-gestalten/**

HTML	Bedeutung	Standardwert der CSS-Eigenschaft Display
<table>	Definiert die Tabelle. TABLE enthält die weiteren Tabellen-Elemente.	**display: table;**
<caption>	Optionale Beschreibung der Tabelle. Die Tabelle erscheint per Standard automatisch überhalb der Tabelle.	**display: table-caption**
<thead>	Definiert den Kopf der Tabelle. Tabellen-Kopf. Der THEAD der Tabelle wird per Standard immer über dem TBODY angezeigt.	**display: table-header-group;**
<th>	Tabellen-Kopf-Zelle im Tabellen-Kopf. Mit der Eigenschaft VERTIKAL-ALIGN kann Text und INLINE-Inhalt auch vertikal zentriert werden.	**display: table-cell;**
<tbody>	Tabellen-Körper, der Hauptteil einer Tabelle. Per Standard erscheint er TBODY unterhalb von THEAD und überhalb von TFOOT.	**display: table-row-group;**
<tr>	Tabellen-Zeile (Table Row). Mit TR werden Zeilen sowohl in THEAD, TBODY und TFOOT markiert. Eine Zelle (TD oder TH) muss in einer Zeile (TR) liegen.	**display: table-row;**
<td>	Tabellen-Daten (Table Data) im Hauptteil der Tabelle. Mit der Eigenschaft VERTIKAL-ALIGN kann Text und INLINE-Inhalt auch vertikal zentriert werden.	**display: table-cell;**
<tfoot>	Definiert den Fuß der Tabelle. Per Standard liegt der TFOOT unterhalb von TBODY.	**display: table-footer-group;**

Tabellenbezogene Attribute

Attribut	Bedeutung	Semantik	Beispiel
colspan="..."	Attribut für <th> und <td&gt. Überbrückt Spalten bzw. Säulen in der Tabelle um Zellen horizontal von links nach rechts zu Verbinden. Es wird ein Zahlenwert zugeordnet und entsprechend viele Zellen verbunden.	**Säulen-Bereich (Table Column)**	**<td colspan="2">**
rowspan="..."	Attribut für <th> und <td>. Überbrückt Zeilen in der Tabelle um Zellen vertikal von oben nach unten zu Verbinden. Es wird ein Zahlenwert zugeordnet und entsprechend viele Zellen verbunden	**Zeilen-Bereich**	**<td rowpan="3">**

A	B	C	D	E
1A	1B	1C	1D	1E
2A	2B	2C	2D	2E
3A	3B	3C	3D	3E
4A		4C	4D	4E
5A	5B	5C	5D	5E
6A	6B	6C	6D	6E
7A	7B	7C	7D	7E
8A	8B		8D	8E

Im HTML der Tabelle sollte der Kopf separat mit dem <th> getaggt sein. <tr> kennzeichnet jeweils eine Zeile.

Spannend sind dann die Möglichkeiten der CSS Pseudoklassen. Unterschiedliche Färbungen der Spalten und Zeilen oder eine farbige Markierung leerer Zellen sind leicht zu realsieren.

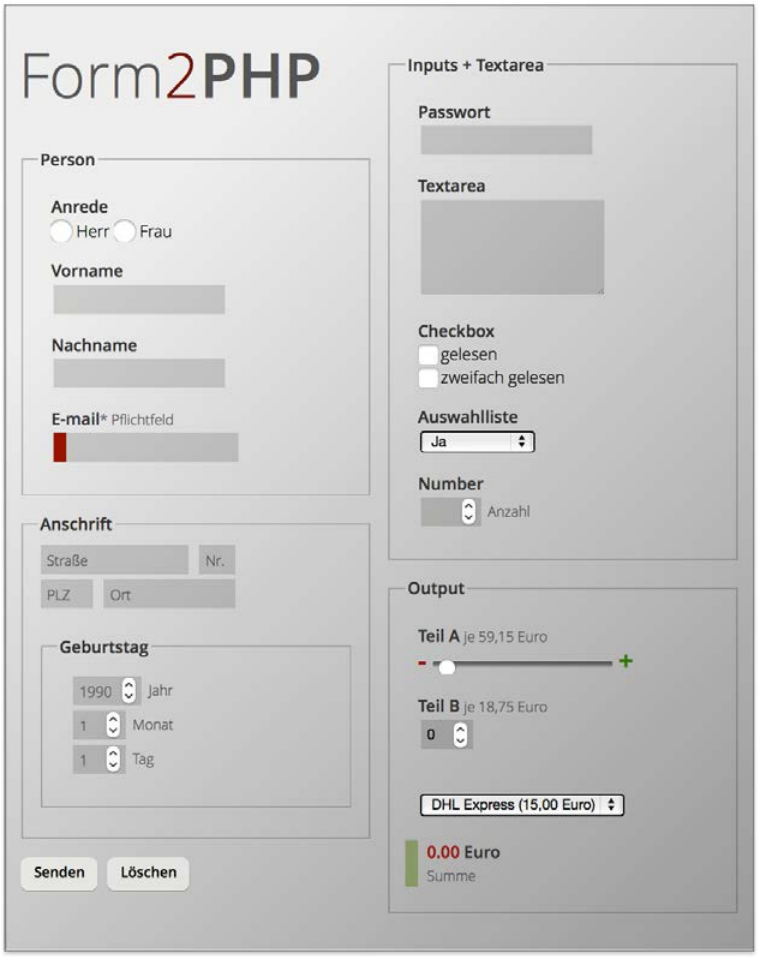
```
<table>
<thead>
<tr><th>A</th><th>B</th><th>C</th><th>D</th><th>E</th></tr>
</thead>
<tbody>
<tr><td>1A</td><td>1B</td><td>1C</td><td>1D</td><td>1E</td></tr>
<tr><td>2A</td><td>2B</td><td>2C</td><td>2D</td><td>2E</td></tr>
<tr><td>3A</td><td>3B</td><td>3C</td><td>3D</td><td>3E</td></tr>
<tr><td>4A</td><td></td><td>4C</td><td>4D</td><td>4E</td></tr>
<tr><td>5A</td><td>5B</td><td>5C</td><td>5D</td><td>5E</td></tr>
<tr><td>6A</td><td>6B</td><td>6C</td><td>6D</td><td>6E</td></tr>
<tr><td>7A</td><td>7B</td><td>7C</td><td>7D</td><td>7E</td></tr>
<tr><td>8A</td><td>8B</td><td></td><td>8D</td><td>8E</td></tr>
</tbody>
</table>
```

Formularbezogene Elemente

Formular

code.arnoldbodeschule.de**/formular-elemente/**

Element	Semantik	Beschreibung
<form>	**Formular-Container**	Definiert das Formular
<input>	**Eingabe**	Eingabe-Element das über das Attribut type="..." näher spezifiziert werden kann.
<textarea>	**Mehrzeilige Texteingabe**	Mehrzeiliges Textfeld
<button>	**Button**	Button zum Klicken. Kann Eigenschaften wie „Senden" oder „Löschen" erhalten.
<select>	**Auswahllisten-Container**	Container für die Auswahlliste. Umschließt die <option>.
<option>	**Auswahllisten-Element**	Element in <select>. Vergleichbar zu
<optgroup>	**Auswahllisten-Element-Gruppe**	Gruppiert ähnliche <option>
<label>	**Element-Beschriftung**	Beschreibt ein Formular-Element. Dabei wird mit dem Attribut for='...' auf die gewünschte ID des Ziels verwiesen.
<fieldset>	**Gruppe von Elementen**	Gruppiert verwandte Elemente im Formular
<legend>	**Gruppen-Beschreibung**	Beschreibung eines <fieldset>
<output>	**Kalkulierte Ausgabe**	Berechnet Ergebnisse im Formular. Muss im <form> eingeleitet werden. Basiert auf Javascript.
<keygen>	**Schlüsselgenerator**	Schlüsselgenerator z.B. für Passwörter der nicht vollständig supportet ist.
<datalist>	**Eingabe-Stichwörter**	Nicht vollständig supportete Ausfüllhilfe. Dabei werden Stichwörter vorgeschlagen.



Formulare können mit HMTL grundsätzlich aufgebaut werden. Mit CSS werden sie gestaltet.

Die Interaktion geschieht dann jedoch auf Basis von Javascript, PHP oder MySQL.

Mehr dazu gibt es in dem Kapitel „Form2SQL"-

Beispiel Formulare können für die Übergabe an PHP und dem Versenden von Mails mitsamt Links für die nötige Software sind:

http://code.arnoldbodeschule.de/formular-action/

Formularbezogene Attribute

Attribute für <form>

Attribut	Bedeutung	Beispiel
.toFixed(2)	Gibt die Werte immer auf 2 Kommas-tellen gerundet aus.	`<form oninput='summe.value = (parseFloat(stueck.value) * 18.75 + parseFloat(versand.value)).toFixed(2)'>`
oninput= "x.value= parse...(a.value) + parse...(b.value)"	Bereitet die Ausgabe im <output> vor. Man definiert den Namen für das <output> und die gewünschte Berechnung. Wenn mit Zahlen gearbeitet wird kommt entweder parseInt für ganze Zahlen oder parseFloat für die Ausgabe mit Kommastellen in Frage. Es müssen außerdem der NAME der Inputs angegeben werden, mit deren Werten gerechnet wird. Dies ist Javascript. Im später verwendeten <output> wird nichts berechnet. <output> ist lediglich der semantisch richtige Ort für die Ausgabe dieser Kalkulation. Vorsicht: Wenn Javascript deaktiviert ist, dann wird auch nichts kalkuliert.	`<form oninput='summe.value = (parseFloat(stueck.value) * 18.75 + parseFloat(versand.value)).toFixed(2)'>`
accept-charset="..."	Zeichensatz für die Datenweitergabe. UTF-8 ist empfohlen.	`<form action="send.php" accept-charset="utf-8">`
autocomplete="..."	Erlaubt dem Browser die Daten des Formular automatisch auszufüllen.	`<form action=" send.php" autocom-plete=" none">`
enctype="..."	x-www-form-urlencoded wandelt die Leerzeichen in ein + um und Sonderzeichen in HTML. form-data ändert nichts und ist gut für Dateien. plain wandelt nur Leerzeichen in + um.	`<form action="send.php" enctype=" plain">`
method="..."	method="get" zeigt Informationen in der URL an. Get ist praktisch im Webshop da Links weitergeschickt werden können. method="post" sendet diese vertraulich und ist Praktisch im Kontaktformular.	`<form action="send.php" method=" get">`
novalidate	Deaktiviert eine Validitätsprüfung (Pattern) vor dem Senden.	`<form action="send.php" novalidate>`
target="..."	Öffnet den über action="" definierte Inhalt im gleichen oder anderen Browserfenster.	`<form action="send.php" target=" _ self">`
action="..."	Ist ein Attribut der <form>. Definiert den nächsten Schritt nach dem Klicken des Senden-Buttons. Z.B. die PHP-Funktion die dann aufgerufen wird. Aus Sicherheitsgründen sind die Beispiele als Download verfügbar.	`<form action="send.php">`
name="..."	Der Name kennzeichnet die Elemente des Formular und das Formular selber. Name spielt eine wichtige Rolle bei der Weitergabe der Daten des Formulars.	`<textarea name="nachricht" id="Nachricht">`
label=""	Beschriftung für <optgroup>	`<optgroup label="Tiere">`

Eingebabefeldbezogene Attribute im Formular

Attribut	Bedeutung	Beispiel
type="button"	Button	`<input type="button">`
type="reset"	Button mit der Funktion "Formular-Inhalte löschen"	`<input type="reset">`
type="submit"	Button mit der Funktion senden. Führt die bei <form action="..." definierte Aktion aus.	`<input type="submit">`
type="color"	Farbwert	`<input type="color">`
type="month"	Monatsangabe in der Schreibweise YYYY-MM (2012-12)	`<input type="month">`
type="time"	Zeitangabe in der Schreibweise hh-mm-ss (16:24:56)	`<input type="time">`
type="week"	Wochenangabe in der Schreibweise YYYY-w00 (2014-w16)	`<input type="week">`
type="date"	Datum in der Schreibweise YYYY-MM-TT (1980-09-12)	`<input type="date">`
multiple	Erlaubt die Eingabe mehrerer Werte. Z.B. mehrere E-Mail Adressen oder die Auswahl mehrere Dateien. Funktioniert auch nur bei E-Mail und File. Bei E-Mail müssen diese mit einem Komma (,) getrennt werden.	`<input type="email" multiple>`
type="email"	Email Adresse. In Verbindung mit required auch ein Test. In Safari, Chrome und Firefox *@*.*. hingegen in Opera *@*. Eine echte Prüfung ist das nicht, aber schon mal ein Schritt in die richtige Richtung.	`<input type="email">`
accept	Erlaubt bei dem type ="file" z.B. bestimmte Datei-typen (pdf, audio/*, video/*, image/*, media_type,)	`<input type="file" type ="pdf">`
autofocus	Element bekommt die pseudoklasse :focus beim Laden der Seite	`<input type="text" required autofocus>`
disabled	Ausgegrauter Wert ohne Eingabemöglichkeit	`<input type="checkbox" disabled>`
type="file"	Upload von Dateien.	`<input type="file">`
required	Pflichtfeld. <input> mit dem Attribut required werden jedoch automatisch mit der Pseudoklasse :invalid markiert, insofern sie leer sind. In Verbindung mit Pattern kann die Eingabe schon recht gut gelenkt werden. Es verhindert allerdings nicht das Senden wenn das Formular ohne dieses Feld abgeschickt wird. Für solche echten Checks können sehr gut Javascript eingesetzt werden.	`<input type="text" required>`
label for="#ID"	Ordnet einem Label die ID des gemeinten Element eindeutig zu. Besonders wichtig für Screenreader.	`<label for="anrede"> Anrede</label>`
checked	Vorauswahl für Checkbox	`<input type="checkbox" checked>`
value=""	Zugewiesener Wert der tatsächlich übertragen wird. Wichtig für die Typen checkbox, radio und range.	`<input type="checkbox" value="Herr" placeholder=" Herr">`
type="checkbox"	Checkbox mit Häkchen. Checkboxen werden für die Bestätigung genommen und es können auch mehrere Checkboxen gleichzeitig gewählt sein.	`<input type="checkbox">`
type="radio"	Wählknopf im Kreis. Radio-Button werden gewählt, wenn man sich zwischen mehreren Wahlmöglichkeiten für eine Wahl entscheiden muss. Der Radio-Button entspricht "entweder-oder". Der Name kommt von den alten amerikanischen Auto-Radio-Knöpfen. Bei denen konnte immer nur ein Knopf gedrückt sein. So wie bei alten Walkman-Kassetenspielern. Falls die Teile heute noch jemand kennt.	`<input type="radio">`
max="10"	Maximaler Wert	`<input type="range" max="100">`
min="1"	Mindestwert	`<input type="range" min="1">`

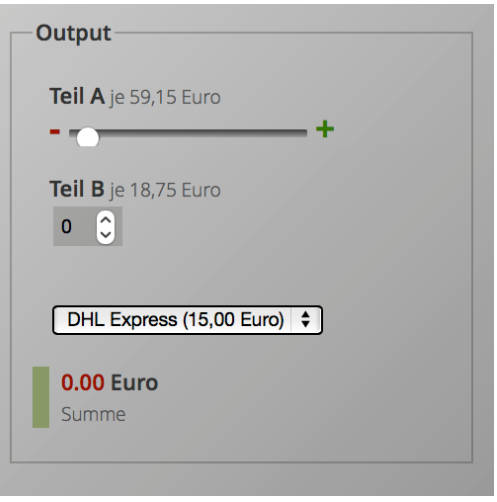
Attribut	Bedeutung	Beispiel
step=""	Schritte in denen die Werte erhöht werden können für type="number".	`<input type="range" min="1" step="1">`
type="number"	Liste um eine Anzahl einzugeben	`<input type="number">`
type="range"	Skala um innerhalb eines definierten Wert zu wählen	`<input type="range">`
pattern="[1-9]{5}"	Definition der Eingabemöglichkeit. Es gelten für das Pattern die "Regular Expressions" (RegEx.) [A-X]{3} z.B. erlaubt genau drei Großbuchstaben bis auf Y und Z. [A-Za-z]{3} z.B. erlaubt genau drei beliebige Groß und Klein-Buchstaben. [1-9]{5} erlaubt genau 5 beliebige Zahlen. [1-9]{1-3} erlaubt beliebige Zahlen mit einer, zwei oder drei Dezimal-Stellen.	`<input type="number" pattern="[1-9]{5}" name="postleitzahl">`
placeholder=""	Platzhalter der nicht übertragen wird.	`<input type="checkbox" value="Herr" name="Anrede">`
readonly	Kann nur gelesen werden	`<input type="text" readonly>`
cols="200"	Breite der Textarea	`<textarea cols="200">`
form="ID"	Definiert das Formular (ID) dem die Textarea zugehört.	`<textarea form="kontakt">`
rows="5"	Zeilen der Textarea	`<textarea rows="5">`
wrap="hard"	Zeilenumbruch der Textarea beim Senden. "Hard" ist mit Zeilenumbruch und "Soft" ist ohne Zeilenumbruch.	`<textarea wrap="hard">`
maxlength="100"	Maximale Anzahl von Zeichen	`<input type="text">`
size=""	Stellt bei textbezogenen Input Elementen, außer bei Textarea, die Breite ein. Diese Eigenschaft sollte über CSS gelöst werden.	`<input type="text" size="40">`
type="password"	Textfeld mit einer Zeile bei dem die eingegebenen Zeichen als • oder * angezeigt werden.	`<input type="password">`
type="text"	Textfeld mit einer Zeile	`<input type="text">`
type="url"	Internetadresse	`<input type="url">`

Attribute für <output>

Attribut	Bedeutung	Beispiel
for="a b"	Im <output> müssen die NAMEN der zu verarbeiteten Inputs angegeben sein.	`<output name='summe' for='stueck versand' >`
name='x'	Der Name des <output> muss dem im oninput= Attribut des <form> entsprechen.	`<output name='summe' for='stueck versand' >`

```
<form action="send.php" method="POST" name="Form2Mail" target="_self"
      accept-charset="UTF-8"
      onsubmit="return validateForm()"
      autocomplete='off'
      oninput='summe.value =   (parseFloat(TeilA.value) * 59.15
                              + parseFloat(TeilB.value) * 18.75
                              + parseFloat(versand.value)).toFixed(2)'>

<fieldset id='Output'>
    <legend>Output</legend>
    <p>
        <label for='range'>Teil A</label> <small>je 59,15 Euro</small><br />
        <strong class='minus'>-</strong>
        <input type='range' id='TeilA' name='TeilA' min='0' max='10' step='1' value='0'>
        <strong class='plus'>+</strong>
    </p>
    <p>
        <label for='Output'>Teil B</label> <small>je 18,75 Euro</small><br />
        <input type='number' min='0' max='20' step='1' id='TeilB' name='TeilB' value='0'>
        <br/><br/>
    </p>
    <p>
        <select id='versand' name='versand'>
            <option value='15.00'>DHL Express (15,00 Euro)</option>
            <option value='4.25'>DHL normal (4,25 Euro)</option>
            <option value='0.00'>Selbstabholer (0,00 Euro) </option>
        </select>
    </p>
    <h4><output for='TeilA TeilB versand' name='summe'>0.00</output> Euro<br/>
    <small>Summe</small></h4>
    </p>
</fieldset>
```



LS 01 **Hallo Welt mit HTML**

Die folgenden Elemente sollen auf der Webseite erscheinen. Schreibe den Code zuerst auf Papier. Danach am Computer testen und zum Abschluss mit der Lösung auf der nächsten Seite vergleichen.

So soll die Webseite später im Browser aussehen.

Element	Inhalt
Link zu der externen design.css	Die design.css ist noch ohne Inhalt.
Titel	Simple Site
Überschrift 1. Kategorie	Hallo Welt
Text	Das ist eine einfache Webseite mit der üblichen Struktur:
Geordnete Liste	HTML, Head, body
2 horizontale Linien	

Beispielhafte Lösung

a) Unten ist eine mögliche Lösung aufgeführt.
Welche Unterschiede im Code hast Du im Vergleich zu dieser Lösung?

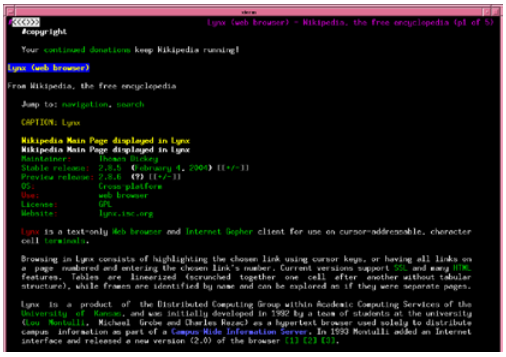
```html
<!DOCTYPE html>
<head>
    <title>Simple Site</title>
    <link rel="stylesheet" type="text/css" href="design.css">
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
</head>
<body>
    <h1>Hallo Welt</h1>
    <p>Das ist eine einfache Webseite mit der üblichen Struktur:</p>
    <ol>
    <li>HTML</li>
    <li>Head</li>
    <li>body</li>
    </ol>
    <hr>
    <hr>
    <body>
</html>
```

webmontag-kassel.de

Web Montag ist ein informelles, nicht-kommerzielles, dezentral organisiertes Treffen, das zum Ziel hat, all diejenigen miteinander zu verbinden, die die Zukunft des Internet gestalten.

In der Zeit vor CSS

Zu Beginn von HTML gab es KEINE Möglichkeit für die Gestaltung. Im Laufe der ersten Jahre Zeit gab durchaus Alternative Versuche: RRP, PWP, FOSI, DSSSL, PSL, CHSS, and JSSS. Die haben sich jedoch nicht durchgesetzt.



RPP in einem LINX Browser

Beispiel für RRP, ein Vorläufer von CSS.
Setzt den Body in die Schriftfamilie Helvetice
und 24 Pixel Schriftgröße.

```
@BODY fo(fa=he,si=24)
```

Geschichte von CSS

1994 hat Håkon Wium Lie bei der W3C das Konzept CSS vorgestellt und seitdem wird die Gestaltung von Webseiten mittels Cascading Style Sheets (CSS) gemacht. Er war Mitarbeiter von Tim Berners-Lee am CERN.

Håkon Wium Lie CTO of Opera & Founder of CSS, CC 2.0

CSS kann mit „Kaskadierende Style Blätter“ oder, meiner Meinung nach besser, frei mit „Stufenweise aufgebaute Design Formate“ übersetzt werden. CSS ist der Standard für Gestaltung und wird ständig weiterentwickelt. Neue Funktionen von CSS werden ständig veröffentlicht. Jeder kann an der Entwicklung teilnehmen. Große Firmen wie Google, Apple, Firefox oder Adobe prägen die Neuerungen von CSS enorm.

Lie fand Unterstützung beim wenige Wochen zuvor gegründeten W3C. Lie brachte die typische Vererbung per „Kaskade“ ein. Die Entwicklung entstand in Kooperation mit anderen Kollegen. Die frühen Entwürfe von CSS unterschieden sich deutlich von dem heutigen CSSS. Z. B. gab es Punkte statt Leerzeichen und komplexe „Einfluss“-Berechnungen. Diese „Einfluss“-Berechnung ist mittlerweile durch die Spezifizität der Kaskade praktischer geklärt.

Microsoft brachte CSS als Standard deutlich voran. Jedoch zum dem Preis von sehr unterschiedlichen Interpretationenen. CSS ist seit 20 Jahren unverzichtbar

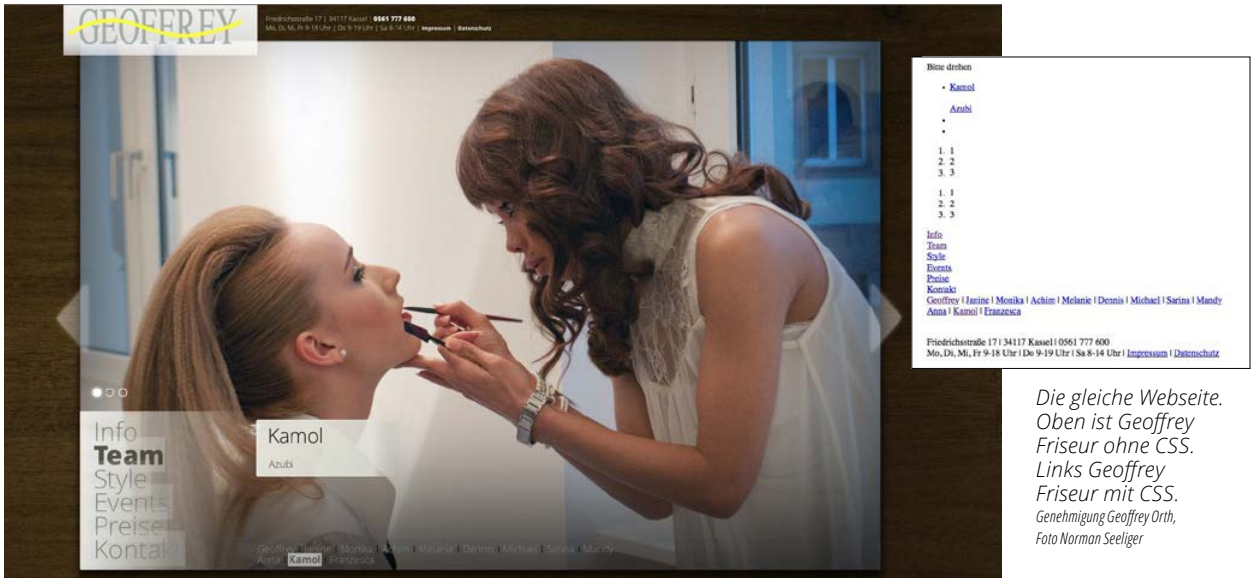
Beispiel für „historisches“ CSS
mit Punkt und Einfluss-Berechnung:

```
h1.font.size = 24pt 100%
```

```
h2.font.size = 20pt 40%
```

Beispiel für „aktuelles“ CSS

```
h1 {font-size: 24px;
    font-family: helvetica;}
```



Die gleiche Webseite. Oben ist Geoffrey Friseur ohne CSS. Links Geoffrey Friseur mit CSS.
Genehmigung Geoffrey Orth, Foto Norman Seeliger

Einbindung von CSS

Besonders wertvoll ist das verweisen auf eine externe CSS. Im *<head>* einer HTML Datei wird mit *<link>* auf die externe CSS verwiesen, z.B:

```
<link rel="stylesheet" href="formate.css">
```

Sollen Webseiten für mehrere Geräte optimiert sein, dann werden auch mehrere spezielle CSS Dokumente verwendet. Z.B. indem Mindestangaben für Breite oder Format definiert sind.

```
<link rel="stylesheet" media="screen and (max-height: 1024px) href="1.css">
<link rel="stylesheet" media="screen and (min-height: 1025px) href="2.css">
```

Die CSS Angaben können auch in den *<head>* der HTML Dokumente geschrieben werden. Ich empfehle das nicht für das Entwickeln einer Seite, weil es schnell zu unübersichtlichen HTML Dokumenten führen kann. Bei kleinen Projekten ist es wiederum praktisch. Allerdings ist anzumerken dass CSS aus dem <style> am Schnellsten geladen werden kann. Wir unterscheiden zwischen den Enwickeln der Seite und dem Betreiben der Seite.

```
<head>
  <style>
    h1 { color:#ff0000; }
  </style>
```

CSS Eigenschaften können auch im *<body>* bei den HTML-Tags geschrieben werden. Dies wird häufig bei Bildern gemacht. Mit Angaben von der Breite, der Höhe oder dem Rahmen wird der Browser bei der Darstellung eines Bildes unterstützt, weil die Dimensionen des Bildes bekannt sind, bevor ein Bild vollständig geladen wird.

```
<img src="LolCat.jpg" width="360" height="900" border="0">
```

Auch wird dies häufig bei Slidern oder sonstigen moderneren Effekten genutzt, wie z.B. bei dem Unslider:

```
 <li style="background-image: url(,img-slide/1aAnfahrtKarte.jpg');">
```

Aufbau von CSS

Selektor { Eigenschaft: Wert; }

Der Aufbau erfolgt über einen Selektor. Als Selektor können generelle HTML Funktionen wie *<a>*, *<img>*, *<h1>* oder *<p>* benutzt werden und darüber hinaus können diese noch mittels Klassen oder ID beliebig benannt werden. Nach dem Selektor kommt eine öffnende geschweifte Klammer, dann eine oder mehrere konkrete CSS Eigenschaft(en) und ein Wert mit Semikolon sowie am Ende eine abschließende geschweifte Klammer.

```
h1       {color: #ff0000;}
.border  {border:1px solid #000;}
```

Selektoren kombinieren

Gemeinsame Werte können zusammengefasst definiert werden.

```
h1, h2, h3, h4 {font-family: Verdana, font-weight: bold; color: #555}
```

Besondere Kombinationen von Selektoren können auch definiert sein. Z.B. die Farbe von *<h2>* ist nur dann Schwarz, insofern das Tag *<h2>* sich in dem Div mit der ID *navi* befindet.

```
#navi h2 { color: #000; }
```

Natürlich geht dies auch mit einer Kombination aus Klasse und Selektor. Z.B. dass die Farbe von *<h2>* Rot wird, sobald es sich in einer Div mit der Klasse navi befindet.

```
.navi h2 { color: #F00; }
```

Eltern-Kind Prinzip

Eltern geben ihre Eigenschaften an ihre Kinder weiter. Genauso ist es in HTML und CSS. Mit Eltern-Elementen sind in HTML alle übergeordneten Elemente gemeint. Alle darin enthaltenden Elemente sind jeweils Kinder. Einmal definierte CSS Eigenschaften der Eltern vererben sich an die Kinder. Allerdings haben die Kinder praktisch alle Freiheiten und können, so es denn gewünscht ist, auch eigene Wege gehen. Es ist immer die zuletzt definierte Eigenschaft dominant.

```
<html>
  <head> <!-- html ist Elternelement von head -->
  <body>                  <!-- erbt Eigenschaften von html -->
    <div>                 <!-- erbt Eigenschaften von html, head, body -->
      <p>Kind von div</p> <!-- erbt Eigenschaften von html, head, body, div -->
    </div>
    <p>Kind von body</p>  <!-- erbt Eigenschaften von head, html, body -->
  </body>
</html>
```

Stufenweise vererben

Das Eltern-Kind Prinzip sorgt dafür, dass alle definierten Eigenschaften vererbt werden. Im folgenden Beispiel werden die Schriftart Arial und die Schriftgröße 15px vererbt. Sobald die Maus über den Link fährt (*Hover)* ändert sich die Farbe in Rot und der Text wird unterstrichen. Die Vererbung sorgt dafür, dass die Schrift immer noch Arial und die Schriftgröße auch weiterhin 15px ist.

```
a {font-family: Arial; color: #fff; font-size: 15px; text-decoration: none;}
a:hover {color: #f00; text-decoration: underline;}
```

Universalselektor *

Mit * können alle Elemente einer Webseite gleichzeitig erreicht werden. So sorgt das folgende Beispiel dafür, dass in jedem Element die Schriftfamilie Arial oder als Ersatz die Schriftfamilie Verdana benutzt wird.

```
* {font-family: arial, verdana;}
```

Geschickt mit CSS selektieren

Die geschickte Kombination stimmiger Selektoren, pseudo-classes und -elements und des Eltern-Kind-Prinzips kann viel Arbeit ersparen. Aber man sollte seine Werkzeuge kennen.

Bei der Vielzahl an Möglichkeiten nur für die Auswahl (Selektion) gilt es sicherlich erst mal die Ruhe behalten und die Übersicht zu gewinnen.

Durch geschickte Nutzung der Selektoren im CSS und die Berücksichtigung des Eltern-Kind Prinzip kannst Du dir viel dann auf Dauer sehr viel Arbeit ersparen. Auch wenn Du z.B. Themes für Projekte einkaufst, die diese Kombinationen sind essentiell für das tiefere Verständnis.

Selektoren

code.arnoldbodeschule.de/**geschickt-mit-css-selektieren/**

Selektor	Bedeutung	Übersetzung	CSS Beispiel
.class	CSS Klassen bündeln beliebige CSS Eigenschaften und können mehrfach benutzt werden. Die im CSS definierte Class muss im HTML als Attribut zugeordnet werden. Im Sinn einer modulorientierten Konzeption ist das Styling ausschließlich über CLASS empfohlen. Die Namensgebung sollte aussagekräftig sein. Auf ein Element können mehrere CLASS angewendet werden.	Klasse	`.first {...}`
#id	ID identifizieren Elemente eindeutig. Eine ID kann im HTML Dokument nur einmal benutzt werden. Die ID wird im HTML als Attribut zugeordnet. ID bündeln zwar beliebige CSS Eigenschaften aber im Sinn einer modulorientierten Konzeption ist das Styling ausschließlich über CLASS empfohlen. Die ID fungiert bei Bedarf als Identifikator.	ID	`#first {...}`
div>p {...}	Wählt die entsprechenden Elemente aus, welche direkte Kinder-Elemente des ersteren sind. Z.B. span die direkte Kinder von h1 sind. Durch das Größer als Zeichen ">" werden nur die direkten Kinder und nicht die Kindes-Kinder ausgewählt.	Direkter Kinder-Seelektor	`nav>a {...}`
h1 span {...}	Wählt die entsprechenden Elemente aus, welche Kinder-Elemente des ersteren sind. Z.B. span die Kinder und Kindeskinder von h1 sind. Durch das Leerzeichen werden alle Kindeskinder mit ausgewählt.	Kind- und Kindeskind-Seelektor	`main p {...}`
h1, h2, h3 {...}	Wählt die mit dem Komma zusammengefassten Elemente aus. Alle bekommen gleichzeitig die bestimmten Eigenschaften.	Selektoren-Kombination	`h1 h2 p ul {...}`
h1+p {...}	Selektiert nur die Geschwister-Elemente die direkt nach dem ersteren kommen. Z.B. nur die p Elemente, welche direkt nach h1 kommen.	Geschwister-Selektor	`h2+h3 {...}`
h1~p {...}	Wählt alle Geschwister-Elemente aus, welche folgen. Z.B. alle p die nach einem h1 kommen.	Folge-Seelektor	`p~ul {...}`
nav * {...}	Der Stern (*) wählt alle Elemente aus. Kann auf die Kinder-Elemente angewendet werden. Z.B. wähle alle Elemente im NAV aus.	Universalselektor	`* {...}`
p {...}	Das Element wird ausgewählt und bekommt die Eigenschaften.	Selektor	`a {...}`
p.red {...}	Werden 2 Elemente ohne Leerzeichen zusammengeschrieben, dann werden sie somit näher bestimmt. Z.B. p MIT der Klasse red. Nur diese bekommen die Eigenschaften.	Spezifischer Selektor	`li.green {...}`

Kommentare in CSS

/** ... **/	**CSS Kommentar**	Kommentare im CSS, also <style> oder externer CSS.

Pseudoclass für <a>

code.arnoldbodeschule.de/**geschickt-mit-css-selektieren/**

pseudoclass	Bedeutung	CSS Beispiel
:active	Aktiver Link, kann nur auf <a> angewendet werden. Wird erst aktiv, wenn bei <a&g das Attribut HREF gesetzt ist UND der Link geklickt wird, also währen die Maustaste gedrückt ist. Es gilt folgende Reihenfolge für die Gestaltung des Anker-Elements: <a> {...;} :LINK {...;} :VISITED {...;} :HOVER {...;} :ACTIVE {...;}	`a:active {...}`
:hover	Veränderung bei Kontakt mit dem Mauszeiger. Es gilt folgende Reihenfolge für die Gestaltung des Anker-Elements: <a> {...;} :LINK {...;} :VISITED {...;} :HOVER {...;} :ACTIVE {...;}. :HOVER kann im Gegensatz zu :LINK :VISITED und :ACTIVE auf jedes sichtbare Element angewendet werden. :HOVER spielt eine wichtige Rolle in der Benutzerfreundlichkeit. Mit :HOVER wird dem Nutzer eine Möglichkeit für Interaktion signalisiert.	`a:hover {...}`
:link	Unbesuchte Links, kann nur auf <a> angewendet werden. Wird erst aktiv, wenn bei <a&g das Attribut HREF gesetzt ist. Es gilt folgende Reihenfolge für die Gestaltung des Anker-Elements: <a> {...;} :LINK {...;} :VISITED {...;} :HOVER {...;} :ACTIVE {...;}.	`a:link {...}`
:target	Wählt alle diese Elemente, wenn Sie mittels eines Links auf ihre ID verlinkt und angeklickt sind. Klicke ich auf einen Link der auf eine ID verweist, dann erscheint diese ID als Anhang mit # in der Internetadresse oben im Browserfenster. Dieses so gewählte Element kann mit :TARGET gestaltet werden.	`h1:target {...}`

Pseudoclass

code.arnoldbodeschule.de/**geschickt-mit-css-selektieren/**

pseudoclass	Bedeutung	CSS Beispiel
:first-child	Das erste Kinder-Element wird ausgewählt.	`article:first-child {...}`
:first-of-type	Dass erste Element seiner Art wird ausgewählt.	`article:first-of-type {...}`
:lang(...)	Auswahl von Elementen die mit der definierten Sprache markiert sind.	`p:lang(it) {...}`
:last-child	Das letzte Kind-Element.	`article:last-child {...}`
:last-of-type	Das letzte Element seiner Art.	`p:last-of-type {...}`
:not(selector)	Wählt alle Elemente aus, welche nicht zutreffen.	`:not(p) {...}`
:nth-child(n)	Wählt das n-te Kind-Element aus. Für n kann eine beliebige Zahl eingesetzt werden. Es können mit dem Wert ODD (Ungerade) und EVEN (Gerade) für n außerdem leicht wiederholende Muster von Eigenschaften erzeugt werden.	`td:nth-child(3) {...}`
:nth-last-child(n)	Wählt das n-te-letzte Kind seiner Art aus. Für n kann eine beliebige Zahl eingesetzt werden. Es können mit dem Wert ODD (Ungerade) und EVEN (Gerade) für n außerdem leicht wiederholende Muster von Eigenschaften erzeugt werden.	`p:nth-last-of-child(2) {...}`
:nth-last-of-type(n)	Wählt das n-te-letzte Element seiner Art aus. Für n kann eine beliebige Zahl eingesetzt werden. Es können mit dem Wert ODD (Ungerade) und EVEN (Gerade) für n außerdem leicht wiederholende Muster von Eigenschaften erzeugt werden.	`p:nth-last-of-type(3) {...}`
:nth-of-type(n)	Wählt das n-te Element seiner Art aus. Für n kann eine beliebige Zahl eingesetzt werden. Es können mit dem Wert ODD (Ungerade) und EVEN (Gerade) für n außerdem leicht wiederholende Muster von Eigenschaften erzeugt werden.	`p:nth-of-type(even) {...}`
:only-child	Wählt alle Einzelkinder aus.	`p:only-child {...}`
:only-of-type	Wählt alle Einzel-Elemente dieser Art aus.	`p:only-of-type {...}`
:root	Wählt das erste Eltern-Element aus. Das ist das HTML Element.	`p:root {...}`

Pseudo-Elemente

code.arnoldbodeschule.de/**geschickt-mit-css-selektieren/**

Pseudo-Elemente	Bedeutung	Übersetzung	CSS Beispiel
::after	Fügt Inhalte nach dem Element ein.	Danach	`h2::after {content: 'alt';}`
::before	Fügt Inhalte vor dem Element ein. Z.B. ein Bild mit url("/bild.png"). Es können auch Attribute ausgelesen werden. Kann nicht auf leere Elemente wie IMG (Bild) oder BR (break) angewendet werden.	Davor	`h2::before {content: 'Neu';}`
::first-letter	Wählt den ersten Buchstaben des Elements aus.	Erster Buchstabe	`p::first-letter {...}`
::first-line	Wählt die erste Zeile aus.	Erste Zeile	`p::first-line {...}`
::input-placeholder	Platzhalter Text in Formular-Elementen.	Platzhalter	`{...}`
::selection	Wählt die vom Benutzer durchgeführte Markierung mit der Maus.	Auswahl	`p::selection {...}`

Formularbezogene Pseudoclass

code.arnoldbodeschule.de/**geschickt-mit-css-selektieren/**

Pseudoclass	Bedeutung	CSS Beispiel
:checked	Ausgewählte Eingabeelemente im Formular. Z.B. bei einer Checkbox.	`input:checked {...}`
:disabled	Reaktivierte Eingabeelemente.	`input:disabled {...}`
:empty	Elemente ohne Inhalt.	`div:empty {...}`
:enabled	Aktivierte Eingabeelemente.	`input:enabled {...}`
:focus	Fokussierte Eingabeelement, also gerade aktive Elemente.	`input:focus {...}`
:in-range	Verhalten bei Eingabe bestimmter Werte.	`input:in-range {...}`
:invalid	Aussehen von Eingabeelementen falschen Werten.	`input:invalid {...}`
:optional	Markiert optionale Eingabeelement.	`input:optional {...}`
:out-of-range	Markiert Eingabeelement die außerhalb des definierten Bereichs liegen.	`input:out-of-range {...}`
:read-only	Eingabeelemente die nur gelesen werden können.	`input:read-only {...}`
:read-write	Wählt Eingabeelemente mit Lese-Schreib Mode.	`input:read-write {...}`
:required	Wichtige Eingabeelemente.	`input:required {...}`
:valid	Wählt valide Eingabeelemente an.	`input:valid`

Ein Feature im CSS ist die Möglichkeit Eigenschaften innerhalb Kaskade zu überschreiben. Dies erfolgt über den allgemeinen Fluss von Oben nach Unten und wird außerdem durch die sogenannte Spezifizität bestimmt. Eine höhere Spezifizität setzt sich durch.

Spezifizität der Kaskade

code.arnoldbodeschule.de/**geschickt-mit-css-selektieren/**

Spezi	Bedeutung	CSS Beispiel
Browser-Einstellung	Nimmt man keine Einstellungen vor, dann gelten die Standardwerte. Z.B. Times New Roman für eine H1. Je nach Browser sind diese unterschiedlich. Diese Browser-Einstellung haben eine niedrige Spezifiziert.	`h1 {}`
Universelle Selektoren und Vererbung	Universelle Selektor, *, und Vererbung haben eine sehr niedrige Spezifiziert. Sie werden sofort überschrieben. 1 Punkt.	`* {color: red} body {font-family: verdana}`
Typ-Selektoren	Die reinen HTML Elemente haben eine niedrige Spezifizität. 1 Punkt.	`p {color: red}`
Klassen-Selektoren	CSS Klassen haben eine mittlere Spezifität. 10 Punkte.	`.red {color: red}`
Attribut-Selektoren	Attribute Selektoren haben eine mittlere Spezifität. 10 Punkte.	`[href$="menu"] {color: red}`
Pseudoklassen	CSS Pseudoklassen haben eine mittlere Spezifität. 10 Punkte.	`.menu-link:nth-of-type {odd {color: red}}`
ID-Selektoren	CSS Pseudoklassen haben eine hohe Spezifität. 100 Punkte. Man sollte mit ID die Selektion vornehmen, der ID jedoch keine Styles zuweisen. Eine Kombination ist also empfohlen: ID und Klasse.	`#button.red {color: darker}`
Kombination	Bei Kombinationen werden die Punkte addiert und die Eigenschaft Spezifischer. So hat ein Link mit Pseudoklasse a:hover einen Wert von 11 Punkten und überschreibt damit auch später definierte einzelne Definitionen von A. Man sollte allerdings davon absehen den kompletten HTML-DOM im CSS nachzubauen. Im Modulorientierter Konzeptzion ist ein Ziel mit 1-2, maximal 3 Ebenen auszukommen.	`#form .red {color: red;} a:visited {color: #000}`
Reihenfolge	"Die Letzten werden die Ersten sein". Das CSS wird von Oben nach Unten verarbeitet. Bei gleicher Spezifizät überschreiben spätere Eigenschaften die vorherigen. Das ist ein Feature von CSS.	
Inlinestile	Inlinestile werden im HTML zugefügt und entsprechen 1000 Punkten. Sie sind allerdings schwierig zu verwalten. Viele Javascript-Effekte arbeiten mit Inline-Stilen.	`<button style="color: red">Button</button>`
!important	Wird die ein Wert um den Zusatz !important ergänzt, so steigt die Spezifizität. !important kann nur von einer späteren !important Eigenschaft überschrieben werden.	`.red {color: red !important;}`

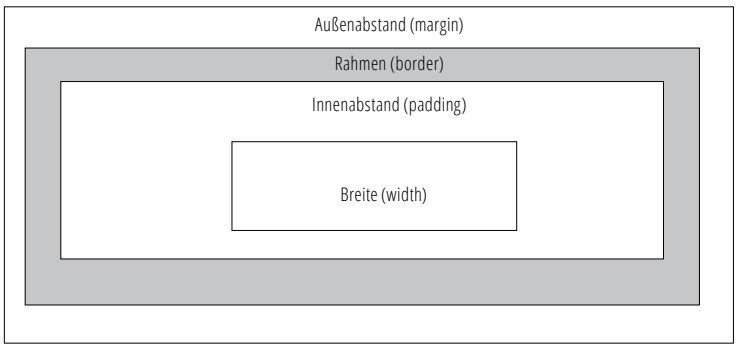
CSS Eigenschaften und typischen Werte

Box-Eigenschaften

code.arnoldbodeschule.de/**design-mit-css/**

Eigenschaft	Beispiel-Werte	Bedeutung
margin:	`5px 3px 8px 5px;`	Aussen-Abstand zum nächsten Element. Im Uhrzeigersinn: 12 Uhr (oben), 3 Uhr (rechts), 6 Uhr (unten), 9 Uhr (links).
margin-top:	`50%; 10px`	Margin kann auch mit -top, -left, -bottom oder-right direkt angesprochen werden.
border-spacing:	`10px;`	Abstand zwischen den Zellen. Die Eigenschaft kann nur auf <table> angewendet werden
padding:	`5px 3px 8px 5px;`	Innen-Abstand des Inhalts zum Rand. Wie bei Margin: Im Uhrzeigersinn: 12 Uhr (oben), 3 Uhr (rechts), 6 Uhr (unten), 9 Uhr (links)
padding-top:	`30%; 20px`	Padding kann auch mit -top, -left, -bottom oder-right direkt angesprochen werden.
width:	`200px;`	Die Breite von Text bestimmen. Hier besteht eine Wechselwirkung mit der Eigenschaft display. Diese ist Standard display: block. Block-Elemente können in der Breite eingestellt werden. Padding, margin und border-width werden darauf addiert.
height:	`200px;`	Die Höhe von Text bestimmen
min-width:	`150px;`	Für Breite und Höhe lässt sich ein minimaler Wert definieren.
max-width:	`400px;`	Für Breite und Höhe lässt sich ein maximaler Wert definieren.
vertical-align:	`top; baseline; middle, sub; super; (...)`	Vertikale Ausrichtung des Inhalts von einem Element mit der Eigenschaft display: inline
box-shadow:	`6px 4px 5px 0px #000;`	Schlagschatten bei dem horizontaler und vertikaler Versatz, Weichzeichner, Intensität und Farbe eingestellt werden können. Empfehlenswert ist dabei die Benutzung von rgba Farben.
column-count:		Anzahl der Spalten
column-width:	`200px;`	Breite der Spalten
columns:	`200px 4`	Kurzschreibweise für column-width und column-count.
column-fill:	`balance, auto`	Art der Füllung der Spalten. Sie kann ausgeglichen (Balance) oder sequenziell (auto) sein.
column-gap:	`15px`	Abstand (Steg) zwischen den Spalten.
column-rule:	`2px dashed #333;`	Kurzschreibweise der folgenden Eigenschaften für die Trennlinie.
column-rule-color:	`rgba (3,3,3,0.5)`	Farbe der Trennlinie zwischen den Spalten.
column-rule-style:	`dashed, solid ...`	Art der Trennlinie zwischen den Spalten.
column-rule-width:	`4px,`	Stärke der Trennlinie des Stegs zwischen den Spalten.

Traditionelle Berechnung der tatsächlichen Breite oder Höhe im CSS Boxmodell



CSS-Eigenschaft	Beispiel-Wert
width	720 px
+ padding-right:	5px
+ padding-left:	5px
+ border-right:	15px
+ border-left	15px
+ margin-left	10px
+margin-right	10px
Tatsächliche Breite	**780px**

CSS Maßeinheiten

Einheit	Bedeutung	Beispiel
px	Der PIXEL ist ein Kofferwort aus Picture und Element. Er ist die kleinste darstellbare auf dem digitalen Ausgabegerät. Je nach Geräteauflösung ist die Ausgagröße unterschiedlich. So hat z.B. ein Smartphone gegenüber einem älteren Monitor eine deutlich höhere Auflösung. PX steht für Pixel.	`font-size: 16px`
%	% bezieht sich je nach Kontext relativ auf das Eltern-Element. % steht für Prozent.	`Width: 60%`
em	Ein EM entspricht der FONT-SIZE (Schriftgröße) des Elements, auf das eine Eigenschaft mit einem EM-Wert angewendet wird. Wird die Schriftgröße selbst (font-size) in EM festgelegt, bezieht sich die Einheit auf die Schriftgröße des Elternelements. Es gilt immer die letzte definierte FONT-SIZE eines Eltern-Elements. Historisch-Typographisch gesehen ist ein EM im Schriftsatz ein Geviert. Der Begriff EM leitet sich aus einer phonetisierung des Buchstaben M ab. Wobei anzumerken ist, dass ein M kein Geviert ist.	`margin: 2em;` `font-size: 2em;`
rem	REM entspricht der Schriftgröße, die für das HTML-Element (Wurzelelement) festgelegt ist. Es ist vererbungsunabhängig. Die FONT-SIZE für REM wird im HTML Element definert und kann mit REM genutzt werden. REM bedeutet Root-EM (Wurzel-EM).	`Font-size: 3rem;`
s	1S entspricht genau einer Sekunde. 1s sind 1000ms.	`transition-delay: 0.5s;`
ms	1MS entspricht dem 1000. Teil einer Sekunde. 1000Ms sind 1s.	`transition: all 400ms;`
deg	1DEG (degree, Grad) entspricht 1 von 360°. Positiver Wert im Uhrzeigersinn und negativer Wert entgegen dem Uhrzeigersinn. 360DEG sind 1TURN. 0.5TURN sind 180DEG.	`transform: rotate(270deg);`
turn	1TURN (Vollwinkel) entspricht einer Kreisumrundung. 360DEG sind 1TURN. 0.75TURN sind 270DEG.	`transform: rotate(.75turn);`
cm	ZM (Zentimeter) ist der hundertste Teil eines Meters. 1 ZM ist 10 Millimeter lang.	`margin-bottom: 2cm;`
mm	MM (Millimeter) ist der tausendste Teil eines Meters. 10MM sind 1 Zentimeter.	`margin: 5mm;`
in	IN (Inch englisch für Zoll) ist 2,54 cm lang.	`margin-left: 3in;`
pt	PT (DTP-Punkt) ist der 72ste Teil eines Inch (Zoll).	`font-size: 12pt;`
pc	PC (Pica) ist der sechste Teil eines Inch (Zoll). Es ist daher 12 Punkt lang.	`font-size: 1pc;`
vw	VW (Viehport-Breite) entspricht dem 100. Teil (1 Prozent) der Breite des Viewport (Anzeigebereichs). 100vw sind die komplette Breite des Viewports.	`Width: 30.5vw;`
vh	VH (Viewport-Höhe)entspricht dem 100. Teil der Höhe des Viewport (Anzeigebereichs). 100vh sind die komplette Höhe des Viewports.	`height: 40vh;`
vmin	Die Einheit VMIN (Viewport-Minimalabmessung) entspricht dem 100. Teil der KLEINEREN Größe bei Breite oder der Höhe des Viewport (Anzeigebereich). Viewport mit den Maßen 800px × 600px entspräche 1vmin also genau 6px. Die Höhe ist in dem Beispiel der kleinere Wert.	`max-width: 70vmin;`
vmax	Die Einheit VMAX (Viewport-Maximalabmessung) entspricht dem 100. Teil der GRÖSSEREN Größe bei Breite oder der Höhe des Viewport (Anzeigebereich). Viewport mit den Maßen 800px × 600px entspräche 1vmin also genau 8px. Die Breite ist in dem Beispiel der größere Wert.	`min-height: 40vmax;`
Fr	FR kann im display: GRID verwendet werden. Bedeutet soviel wie Bruchteil des verfügbaren Platzes. FR ist die Abkürzung von Fractional Space (Bruchteil).	`grid-template-rows: 2fr 1fr;`
ex	EX entspricht der Höhe des Kleinbuchstabens x. Dieses Größenmaß ist sehr interessant für Gestaltung von Drucksachen. Die Lesbarkeit beträgt erfahrungsgemäß den Faktor 500. Eine Schrift mit z.B. 1cm großen Kleinbuchstaben x ist aus 5m noch Lesbar. Kann man gut bei Plakatgestaltung einsetzen. Leider kann es nicht so eindeutig im InDesign definiert werden. EX ist im CSS außerdem noch relativ zu der definierten Font-Size vom Eltern-Element.	`Font-size: 2ex;`
ch	CH entspricht der Breite der Ziffer 0. CH steht für Character (Zeichen). Man kann mit CH relativ genau Lauf-Breiten für Texte bestimmen. Dies ist praktisch bei Eingabeformularen, z. B. für eine Begrenzung auf 20 Zeichen. Bei Mengentext und auch Überschriften ist es ebenso günstig wenn eine maximale Wortanzahl bestimmt werden soll. Z. B. sollte ein Mengentext mit linksbündigem Flattersatz mindestens 40 Zeichen breit und maximal 70-80 Zeichen breit sein. Die Breite hängt letztlich von der Schriftgröße ab. Die Laufweite für Überschriften werden eher geringer ausfallen. Mitunter sehen imLayout 3-4 Wörter pro Zeile einer Überschrift ansprechend aus. Grundsätzlich gilt: Das Lesen von Text sollte ohne seitliche Kopfbewegung und Augenbewegung geschehen können. Wenn für Screendesign Blocksatz eingesetzt wird, dann müssen die Zeilen breiter sein. Blocksatz funktioniert erst mit 8-10 Leerzeichen, also 9-12 Wörtern. Die Silbentrennung funktioniert in Browsern jedoch nicht gut, Blocksatz muss also jeweils getestet werden. Die Zeicheneinheit CH funktioniert am zuverlässigsten bei Schriften mit gleicher Breite (Monotype). CH für Breiten ist aber auch bei jeder anderen Schrift zu empfehlen. Die Grundlage ist die 0, ein Buchstabe M ist breiter und das Buchstabe i ist schmaler. In der Summe passt die Anzahl der Zeichen meistens in die mit CH bestimmte Breite.	`p {font-size: 13px; min-width: 40ch; max-width: 70ch;} input {font-size: 1rem; width: 20ch} h2 {font-size: 36px; min-width: 20ch; max-width: 40ch;}`


```
main        {background:#eee;
             width: 400px; margin: 0 auto;
             border: 4px solid #000;}

header      {background:#999; height:50px;}
nav         {background:#333; height:20px;}
footer      {background:#333; height:30px;}

section     {background: linear-gradient(to top, #aaa,#eee);}

article     {height: 34px;
             border-bottom:
             4px solid rgba(210,40,40,0.9);}
```

```
h1   { border: solid transparent;
       border-width: 42px 38px 48px 39px;
       border-image-source: url('bilder/rahmen.png');
       border-image-slice: 98 96 97 100 fill;
       border-image-repeat: round;
       border-image-outset: 5px;

       font: 700 38px/30px Open Sans;
       color: #220; padding: 12px; margin: 20px;
       display: inline-block;}

h1 em {font: 300 16px/30px Open Sans; color: #320;}
body  {background: #eee;}
```

Border-Eigenschaften

code.arnoldbodeschule.de/**design-mit-css/**

Eigenschaft	Werte	Bedeutung	Beispiel
border	`2px solid #0f0;`	Praktische Kurzschreibweise die border-width, border-color und border-style zusammenfasst	`border: 5px dashed rgba(128, 128, 128,0.6);`
border-collapse:	`separate; collapse`	Steuert ob Rahmen zeilenweise durchgezogen (collapse) oder jede Zelle einzeln (separate) umrandet ist	`border-collapse: collapse;`
border-color:	`#ff0000; red`	Farbe kann mit den CSS-Color-Definitionen bestimmt werden: (color-keyword; #hex; rgb; rgba; hsl; hsla;)	`border-color:`
border-radius:	`2px;`	Stärke der abgerundeten Ecken in Pixel. Verträgt sich in Tabellen nicht mit border-collapse	`border-radius:`
border-style:	`none; solid; dotted; dashed; groove;`	Art der Kontur kann beeinflusst werden: none; solid; dotted; dashed; groove; double; ridge; inset; outset;	`border-style:`
border-top-width:	`6px;`	Mit top, left, bottom und right kann die Kontur unabhängig voneinander adressiert werden.	`border-top-width:`
border-width:	`1px; 5px 2px 8px 2px;`	Stärke der gesamten Kontur. Einzelne Rahmen ansprechen.	`border-width:`
outline:	`1px solid red;`	Erzeugt weitere Kontur allerdings unabhängig vom Box-Modell.	`outline: 5px dotted #fff;`
outline-offset:	`15px;`	Bestimmt den Abstand der Outline-Kontur	`outline-offset: 20px;`
border-image-source:	`url('rahmen.png');`	Lädt den Border-Image Sprite	`border-image-source: url('border.png');`
border-image-slice:	`98 96 97 100 fill;`	Zerschneidet den Sprite in Slices. Innenfläche kann mit fill gefüllt werden. Es gibt praktische Editoren um dieses Zerschnippeln vorzunehmen.	`border-image-slice: 100 100 105 100 fill;`
border-image-repeat:	`repeat, round, stretch; space;`	Der Rahmen-Slices wiederholen sich (repeat), wiederholen sich mit Anpassung (round), werden gestreckt (stretch) oder wiederholt und ergänzt (space).	`border-image-repeat: no-repeat`
border-image-outset:	`5px`	Verschiebung des Rahmens	`border-image-outset: 5px`
-VENDOR-border-image:	`url(rahmen.jpg) 30 round;`	Kurzschreibweise die allerdings aufgrund der notwendigen Vendor Prefixe keine nennenswerten Vorteile in der Zeitersparnis bietet.	`-webkit-border-image: url(rahmen.jpg) 30 round; -moz-border-image: url(rahmen.jpg) 30 round;`

Farb-Eigenschaften

Eigenschaft	Werte	Bedeutung
color:	#00ff00; (...)	Zeichenfarbe
background-color:	rgba(240, 240, 240, 0.5); #ff0; (...)	Hintergrundfarbe
opacity:	0.9; 1; 0.05;	Deckkraft. Je höher die Opazität, desto niedriger die Transparenz.
background:	linear-gradient(to right, red , blue);	Kurzschreibweise für die Backgroundeigenschaften. Ein Farbverlauf ist möglich. Verläufe sollten mittels Editor komponiert werden. http://www.colorzilla.com/gradient-editor/

Farb-Werte

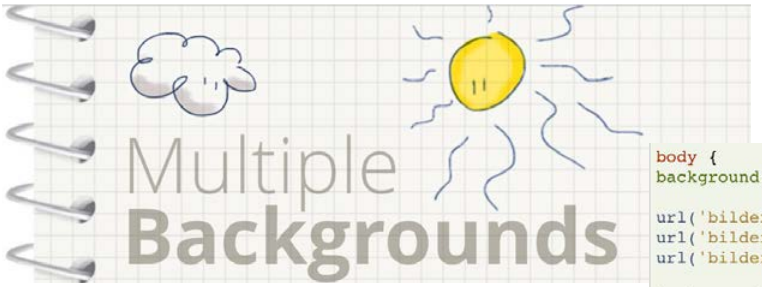
Eigenschaft	Name	Bedeutung
red;	Color-Key Word	Definierte Farbnamen
#ff0000; #0f0;	Hex-RGB	Hexadezimale RGB
rgb(255, 0, 0);	RGB	Farbwert als RGB, hier Rot
rgba(255, 0, 0, 0.4);	RGBA	Farbwert als RGB mit Angabe der Opazität (Alpha), hier Rot das zu 40% sichtbar ist
transparent	transparent	Durchsichtig
currentColor	currentColor	Bezieht sich auf die Farbe die bei dem Element über Color eingestellt ist.
Hsl(0,100%,50%)	HSL	Hue (Farbton in Grad) Saturation (Sättigung) Lightness (Helligkeit), hier Rot
Hsla(0,100%,50%, 0.1)	HSLA	Hue (Farbton in Grad) Saturation (Sättigung) Lightness (Helligkeit) Alpha (Opazität)
linear-gradient(to top, #f00 , #00f);	Verlauf	Der Eigenschaft Background kann ein linearer oder radialer Verlauf zugeordnet werden.

Text-Eigenschaften

code.arnoldbodeschule.de/**text-mit-css-gestalten/**

Eigenschaft	Werte	Beschreibung	Beispiel
font:	`italic bold 12px/30px Georgia serif;`	Kurzschreibweise für die Font Eigenschaften: font-weight font-size/line-height font-family.	`font: 300 24px/30px Open Sans;`
font-size:	`14px;`	Schriftgröße mit absoluten Werten z.B. wie Pixel oder auch relative Größe wie REM.	`font-size: 5em:`
font-weight:	`bold; 800; 300;`	Schriftschnitt z.B. Bold oder Light. Bei Google Webfonts sind die Schnitte meist Zahlen. Je niedriger desto dünner. 400 ist dann Dünn. 700 ist Fett.	`font-weight: 700;`
font-family:	`verdana; sans-serif;`	Schriftfamilie. Man sollte den Systemschriften wie Verdana oder Arial Vorzug gegenüber anderen Schriften geben. Diese sind schneller geladen. Man kann auch gut damit auskommen die Schrift grob über sans-serif (Serifenlos) oder serif (Serifenschrift) zu definieren. Je nach Gerät wird dann eine typische genommen. Wenn man andere Schriften verwendet, dann am Besten als WOFF bzw. WOFF2 Schrift.	`font-family: Open Sans`
line-height:	`18px;`	Zeilenabstand. Je kleiner die Schrift, desto größer sollte das Verhältnis von Schrift-größe und Zeilenabstand sein. Bei großen Schriften kann es 1:1 oder sogar kleiner sein.	`line-height: 24px`
list-style-type:	`none, circle, square, disc, decimal, lower-roman, upper-roman, lower-greek, decimal-leading-zero, armenian, georgian, inherit`	none = Kein Aufzählungszeichen, circle = Kreis, nur Rahmen, square = Quadrat, disc = Gefüllter Kreis, decimal = Dezimalzahlen (1. ,2. , 3. , ...), lower-roman = Kleine römische Zahlen (i. ,ii. ,iii. , ...), upper-roman = Grosse römische Zahlen (I. ,II. , III., ...), decimal-leading-zero = Dezimalzahlen mit führender 0 (01. ,02. , 03. , ...), lower-greek = Kleine griechische Nummerierung (alpha, beta, gamma,...), lower-latin = Kleines ABC (a. ,b., c. , ...), upper-latin = Große ABC (A., B. ,C. , ...), armenian = Armenische Nummerierung, georgian = Georgische Nummerierung, Inherit = erbt vom letzten definierten Elternelement	`ul li {list-style-type: square;}`

Eigenschaft	Werte	Beschreibung	Beispiel
letter-spacing:	`4px;`	Spationieren bzw. Zeichenabstand ist das Erhöhen des Buchstabenabstandes. Die muß mitunter unbedingt gemacht werden. Z.B. liest sich je nach Schrift der Wort click wie dick. In so einem Fall wird das letter-spacing für das Wort etwas erhöht. Das habe ich mal bei einer Werbung von Apple gesehen. :)	`letter-spacing: 1px;`
text-align:	`left; right; center; justify;`	Absatzausrichtung: linksbündig, zentriert, Blocksatz etc. Text-align greift nur bei Text und INLINE-Elementen. Damit können keine BLOCK-Elemente ausgerichtet werden.	`text-align: left;`
vertical-align:	`sub; super; baseline (Standard); top; bottom; middle; text-top; text-bottom; super; baseline; top; bottom; middle; text-top; text-bottom;`	Vertikale Ausrichtung bei Elementen mit der Eigenschaft display: TABEL-DATA. Ist für Tabellen gedacht.	`vertical-align: bottom`
text-decoration:	`none; overline; underline; line-through;`	Unterstreichen und andere Auszeichnungen. Ist ein Standardwert bei dem Link (a).	`text-decoration: underline;`
text-indent:	`15px;`	Texteinzug der ersten Zeile.	`text-indent: 9px;`
text-shadow:	`2px 2px 4px #ff0000;`	Schlagschatten. Verschiebung auf der X- und Y-Achse sowie Weichzeichner und Farbe. Vergleichbar zu Box-Shadow. Kann mit Komma getrennt mehrfach angewendet werden. RGBA Werte ergeben meist ein weicheres Ergebnis.	`text-shadow: 2px 2px 1px #000, 5px 5px 9px #333;`
text-transform:	`uppercase; lowercase; capitalize;`	Großbuchstaben, Kleinbuchstaben, Kapitälchen	`text-transform: lowercase`
orphans:	`3; 5; (...)`	Verhindert einzelne Zeilen am Ende einer neuen Seite oder Spalte. Entspricht dem typographischen Begriff "Schusterjunge". "Der Schusterjunge weiß nicht wo er hin soll."	`orphans: 3;`
widows:	`3; 5; (...)`	Verhindert einzelne Zeilen zu Beginn einer neuen Seite oder Spalte. Entspricht dem typographischen Begriff "Hurenkind". "Das Hurenkind weiß nicht wo es herkommt."	`widow: 3`
page-break-before	`always; avoid; left; right; auto`	Erzeugt einen Spalten bzw. Seiten-umbruch vor dem Element.	`page-break-before: avoid;`
page-break-inside	`avoid; auto`	Verhindert den Umbruch innerhalb des Textes (avoid) oder erlaubt den Umbruch (auto).	`page-break-inside: avoid;`
page-break-after	`always; avoid; left; right; auto`	Erzeugt einen Spalten bzw. Seiten-umbruch nach dem Element.	`page-break-after: right`
-webkit-column-break-inside:	`avoid;`	Element wird bei Webkit Browsern (Safari, Chrome) nicht umbrochen sondern beginnt vollständig auf der neuen Seite oder Spalte.	`-webkit-column-break-inside: avoid`
column-span:	`2`	Elemente mit dieser Eigenschaft können in ihrer Breite über mehrere Spalten laufen.	`column-span: 2`




```
body {
background: url('bilder/spirale2.png')
            repeat-y scroll left,
url('bilder/karo2.png') repeat scroll left top,
url('bilder/sonne.png') no-repeat 310px -30px,
url('bilder/wolke.png') no-repeat 95px 8px / 120px;

background-color: rgb(247,246,242);}

h1 {font: 700 60px/55px Open Sans;
    margin: 95px 0 0 90px; color: rgba(75,70,50,0.4);}

h1 span {font-weight: 300;}
```

Background-Eigenschaften

code.arnoldbodeschule.de/**hintergrund-bilder/**

Eigenschaft	Werte	Bedeutung	Semantic	Beispiel
background:	`#000 url("*.*") no-repeat fixed center;`	Praktische Kurzschreibweise der Eigenschaften background -color, -image, -position, -size, -repeat, -origin, -clip, -attachment	Hintergrund	`background: #000 url("back.jpg") repeat fixed top;`
background-color:	`#fff;`	Farbe des Hintergrundes mittels der CSS üblichen Farbwerte.	Hintergrund-Farbe	`background-color:`
background-image:	`url("bild.jpg");`	Pfadangabe des Hintergrundbildes	Hintergrund-Bild	`background-image: url("kittens.jpg");`
background-position:	`left, top, right, center (...), 20% 30%, 50px 200px;`	Position im Element	Hintergrund-Position	`background-position: left`
background-size:	`auto, 50px, 50%, cover, contain;`	Größe des Hintergrund unverändert (auto), in Pixel oder Prozent. Cover füllt die Box aus wobei Bereiche nicht sichtbar werden, contain füllt den Bereich aus wobei alle Bereiche sichtbar bleiben.	Hintergrund-Größe	`background-size: 100% auto;`
background-repeat:	`repeat, repeat-x,`	repeat-y; no-repeat;	Der Hintergrund wiederholt sich horizontal und vertikal, nur auf der X- oder Y-Achse oder überhaupt nicht.	`background-repeat: repeat-x;`
background-origin:	`border-box, padding-box, content-box;`	Der Ursprung des Hintergrundbildes kann entsprechend des allgemeinen Box Modells an der Box des Rahmens, des Paddings oder des Inhalts festgelegt werden.	Hintergrund-Bezugspunkt (Quelle)	`background-origin: content-box;`
background-clip:	`border-box, padding-box, content-box;`	Hintergrundbild kann entsprechend des allgemeinen Box Modells an der Box des Rahmens, des Paddings oder des Inhalts abgeschnitten werden.	Hintergrund-Begrenzung	`background-clip: content-box;`
background-attachment:	`fixed, scroll;`	Hintergrundbild scrollt mit oder ist fixiert. Scroll ist Standard.	Hintergrund-Verhalten	`background-attachment: fixed;`

Transform und Perspective

code.arnoldbodeschule.de**/design-mit-css/**
/3d-transform-perspective/

CSS	Werte	Bedeutung
transform:	`rotateY(5deg) scale(2);`	TRANSFORM kann über diverse Werte das Element manipulieren. Rotieren, Skalieren, Kippen und eine ganze Reihe mehr Veränderungen sind möglich. Es können mehrere Werte mit Leerzeichen aneinandergereiht werden. Die neuen Browser unterstützen TRANSFORM in einfacher Schreibweise. Ältere Versionen der Browser benötigen die Vendor-Prefixe (-moz-, -ms-, -webkit, -o-). TRANSFORM ist bei Animationen besonders gut und damit schnell zu berechnen und daher ausdrücklich empfohlen. TRANSFORM legt die Elemente im Browser auf einen anderen Render-Layer. Dies kann bei dünner Schrift zu Unschärfen führen, da ein Pixel nicht ganz getroffen wurde. In so einem Fall sollte TRANSFORM bereits von Anfang an mit einem Wert zugewiesen werden. Z.B. TRANSFORM: translateZ(0) wird dafür gerne genommen.
perspective:	`none, 900px;`	Bestimmt Entfernung der Beobachter Perspektive für die KINDER des 3D Elements. Der Wert NONE zeigt die 3D-Kinder ohne Verzerrung. Ein niedriger Wert lässt die Kinder nah erscheinen.
perspective-origin:	`50% 50%;`	Quelle der 3D Kinder. Ein Wert von 50% 50% lässt sie genau in der Mitte entstehen.
backface-visibility:	`visible; hidden`	Regelt die Sichtbarkeit der Rückseite einer gedrehten Ebene.

Werte für Tranform

code.arnoldbodeschule.de**/design-mit-css/**
/3d-transform-perspective/

Wert	Einheit	Bedeutung
translate	`(x,y);`	Verschieben auf der X und Y Achse
translate3d	`(x,y,z);`	Verschieben auf der X, Y und Z Achse
translateX	`(x);`	Verschiebt nur auf der X-Achse
translateY	`(y);`	Verschiebt nur auf der Y-Achse
translateZ	`(z);`	Verschiebt nur auf der Z-Achse
scale	`(x,y);`	Skaliert Breite und Höhe
scale3d	`(x,y,z);`	Skaliert Breite und Höhe sowie Z Achse
scaleX	`(x);`	Skaliert nur die Breite
scaleY	`(y);`	Skaliert nur die Höhe(YAchse)
scaleZ	`(z);`	Skaliert nur die Z-Achse(Z Achse)
rotate	`(angle);`	Drehen
rotate3d	`(x,y,z,angle);`	3D Rotation
rotateX	`(angle);`	Dreht um die X-Achse
rotateY	`(angle);`	Dreht um die Y-Achse
rotateZ	`(angle);`	Dreht um die Z-Achse
skew	`(x-angle,y-angle)`	Kippen der X- und Y-Achse
skewX	`(angle)`	Kippen der X-Achse
skewY	`(angle)`	Kippen der Y-Achse
matrix	`(n,n,n,n,n,n)`	Zusammenfassung der sechs 2-D Werte
matrix3d	`(n,n,n,n,n,n, n,n,n,n,n,n,n,n,n,n)`	Zusammenfassung der 16 3-D Werte

display

Eigenschaft	Werte	Bedeutung
display:	**none;**	Das Element wird nicht angezeigt und ist praktisch nicht da. Es hat keine Auswirkungen auf das Layout. Das ist der Unterschied zu OPACITY: 0 (Deckkraft: 0) und VISIBILITY: HIDDEN. Diese beiden machen das Element transparent. Ein transparenter Link ist aber z.B. Immer noch klickbar. Transparenter Text ist auch auszahlbar. Mit display: NONE ist der Link hingegen nicht im Browser. Im HTML-Quelltext der Seite wird der Inhalt auf jeden Fall aufgeführt. NONE kann gut genutzt werden um z.B. Inhalte auszublenden die bei anderen Zuständen (:hover, media-querie etc.) eingeblendet werden.
display:	**inline;**	Das Element kann keine Breite und Höhe haben sondern übernimmt diese Werte von seinem Inhalt. Es erzwingt keine neue Zeile. Ist Standard bei SPAN und wird benutzt um z.B. Wörter in einem Absatz anders zu gestalten. Es entspricht einen Zeichenformat in InDesign.
display:	**block;**	Das Element kann Breite und Höhe, Innenabstand und Aussenabstand (Boxmodell) haben. Es erzwingt eine neue Zeile. Die meisten HTML Elemente haben display: BLOCK als Standard. Z.B. H1 bis H6, P oder LI usw.
display:	**inline-block;**	Das Element steht inline, kann aber Breite und Höhe haben, Innenabstand und Aussenabstand haben.(Boxmodell). Standard-Wert für die Breite ist die Breite des eigenen Inhalts.
display:	**list-item;**	Das Element steht inline, kann aber Breite und Höhe haben, Innenabstand und Aussenabstand haben (Boxmodell). Standard-Wert für die Breite ist die Breite des eigenen Inhalts.
display:	**inline-list-item;**	Verhält sich wie LIST-ITEM. Der Inhalt wird jedoch unter dem Aufzählungszeichen umbrochen und nicht unter dem Anfang des Textes.
display:	**inherit;**	Das Element erbt (INHERIT) den Display-Wert von seinem Eltern-Element
display:	**initial;**	Die Display Eigenschaft wird auf den initialen Standard zurückgesetzt.
display:	**flex;**	Das Element verhält sich wie ein Block-Element. Sein Inhalt reagiert auf die Flexbox Eigenschaften. Sehr praktisch zum dynamischen horizontalen und vertikalen Ausrichten von Elementen. Zentrieren geht mit FLEX sehr leicht.
display:	**inline-flex;**	Das Element verhält sich wie ein Inline-Element. Sein Inhalt reagiert auf die Flexbox Eigenschaften. Sehr praktisch zum dynamischen horizontalen und vertikalen Ausrichte von Elementen.
display:	**table;**	Das Element verhält sich wie TABLE. Es kann tabellenbezogene Inhalte, z.B. TBODY, TR, TD etc., haben. Es verhält sich wie ein Block-Element.
display:	**inline-table;**	Hat kein entsprechendes HTML Element. Verhält sich wie display: TABLE, aber ist ein INLINE Element und erzwingt somit keine neue Zeile.
display:	**table-caption;**	Entspricht Element CAPTION. Richtet innerhalb einer Tabelle die Beschreibung oberhalb der Tabelle aus.
display:	**table-cell;**	Entspricht dem Element TD. Einzelne Zelle innerhalb einer ZEILE (TR).
display:	**table-footer-group;**	Entspricht dem Element TFOOT. Fußzeile der Tabelle. Immer letztes Kind-Element von TABLE.
display:	**table-header-group;**	Entspricht dem Element THEAD. Kopfzeile der Tabelle. Immer erstes Kind-Element von TABLE.
display:	**table-row;**	Entspricht dem Element TR. Zeile (Table-Row) in THEAD, TBODY, TFOOT. Enthält die TD.
display:	**table-row-group;**	Entspricht dem Element TBODY. Hauptteil der Tabelle. Unterhalb von THEAD und oberhalb on TFOOT.
display:	**grid;**	Gestaltungsraster für die Kind-Elemente. Ermöglicht mehr Einstellungen im Vergleich zu FLEX, da die Elemente sowohl auf Ebene der Spalten als auch auf ebene der Zeilen verbunden werden können. Leider noch nicht vollständig unterstützt und ist nur über ein Polydill lauffähig. Verhält sich wie als BLOCK-Element. GRID wird näher über weitere Eigenschaften definiert.
display:	**inline-grid;**	Gestaltungsraster für die Kind-Elemente. Ermöglicht mehr Einstellungen als Flex. Leider noch nicht vollständig unterstützt. Verhält sich wie als Inline-Element.
display:	**run-in;**	Dynamisches Verhalten. Je nach Kontext verhält sich das Element als BLOCK, Inline oder Inline-Block. Der Kontext wird dabei vom Nachbarn bestimmt.
display:	**contents;**	Das eigene Box-Verhalten wird deaktiviert.

Position

Eigenschaft	Wert	Bedeutung	Übersetzung	Beispiel
position:	`static;`	Standard Verhalten. STATIC ist automatisch allen Elementen zugewiesen. Kein verschieben mit TOP: oder LEFT etc. ist möglich. Das Element ist im allgemeinen Fluss.	Statisch (Standard)	`position: static;`
position:	`relative;`	Ist Bezugspunkt für absolute Kinder-Elemente. Das Element ist im allgemeinen Fluss. Wird ein RELATIVE Element mit z.B. TOP: 0 verschoben, so nimmt es an seiner ursprünglichen Stelle noch Raum ein.	Relativer Bezugspunkt	`position: relative;`
position:	`absolute:`	Das Element wird aus dem allgemeine Fluss entfernt. Position bezieht sich auf relative-Eltern-Element. Verschoben mit z.B. LEFT: 0 nimmt das Element keinen Raum an seiner ursprünglichen Stelle ein. Das ABSOLUTE Elemente scrollen mit. Verschieben mit TOP, LEFT etc. ist möglich.	Absolut Positioniert	`position: absolute; left: 10px; bottom: 5px`
position:	`fixed;`	Verhalten analog zu absolute, Element scrollt jedoch nicht mit. Es bezieht sich auf den BODY als Bezugspunkt.	Fixiert	`position: fixed; top: 10px; right: 5px`
z-index:	`1; 5; 99; 108;`	ABSOLUTE und FIXED Elemente können übereinander liegen. Z-INDEX bestimmt die Reihenfolge der Schichtung. Die höchste Zahl liegt oben.	Z-Achse (Höhen-Achse)	`z-index: 5;`

Abstände für position: absolute und fixed

Eigenschaft	Werte	Bedeutung
top:	`-5px; 20%; 5em; 10cm (...)`	Oberer Abstand zum letzten relative-Eltern-Element
right:	`5px; -20%; (...)`	Rechter Abstand zum letzten relative-Eltern-Element
bottom:	`-5px; 20%; (...)`	Unterer Abstand zum letzten relative-Eltern-Element
left:	`-5px; 20%; (...)`	Linker Abstand zum letzten relative-Eltern-Element

float, clear und .clearfix

Eigenschaft	Zusammenfassung	Werte	Beispiel
float:	Nimmt das Element aus dem normalen Fluss heraus und platziert es entweder auf der linken oder rechten Seite des Eltern-Elements. Die anderen Inhalte umfließen es. Ursprünglich ist FLOAT für den Umfluss von Texte um Bilder gemacht worden. Im Laufe der Zeit hat es sich als Methode entwickelt komplexere Layouts zu gestalten. Je nachdem welche Browser-Bandbreite unterstützt werden soll, spielt es noch heute in dieser Hinsicht eine Rolle. Kann man für neuere Browser gestalten, dann bietet display: FLEX auf jeden Fall mehr Möglichkeiten. In der Mircro-Ebene (Text und Bilder) ist FLOAT immer noch ein wesentliches Gestaltungselement im Webdesign. FLOAT aktiviert den Fluss der folgenden Elemente nach rechts oder links. Technisch gesehen wird das "gefloatete" Element auf einen anderen Layer im Browser geschoben. Seine Art die Höheninformation auszugeben wird dadurch geändert. Die Höhe des Eltern-Elements kann zusammenbrechen (kollabieren). Texte und Bilder üben allerdings ihre tatsächliche Höhe aus insofern sie die Eigenschaft FLOAT haben. Während die Höhen von Flächen kollabieren, so bleibt der Text mitunter an der ursprünglichen Stelle. Dies führt zu kuriosen Situationen. Dieses Verhalten ist in der Geschichte von FLOAT begründet. FLOAT war halt niemals für moderne Layouts gedacht und geplant. Das Verhalten kann nur mit CLEAR kontrolliert werden. Benutzt man also FLOAT, benötigt man fast immer auch CLEAR. Anzumerken bei FLOAT ist, dass man sich neben CLEAR bei der Konzeption auch Gedanken zu den Höhen der Elemente machen muss. Gleiche Höhen funktionieren am Besten.	`left; right; none; inherit`	`float: left;` `float: right;`
clear:	Beendet den FLOAT. CLEAR beendet entweder den beidseitigen (both), linken (left) oder rechten (right) FLOAT. Clear verhindert das bei dem Eltern-Element mit gefloateten Kindern die Höhe kollabiert. Die Eigenschaft CLEAR bekommt immer das folgende Element, welches nicht mehr "floaten" soll.	`left; right; both; none; inherit;`	`clear: both;` `clear: right;`
.clearfix	CLEARFIX ist eine populäre Methode für das praktische Zuweisen von CLEAR mit Hilfe von des ::AFTER-Pseudo-Elements. Anstatt im HTML ein Element mit der Eigenschaft CLEAR einzubauen fügen wir das benötigte CLEAR mit einer CSS-Klasse ein. Der Begriff CLEARFIX hat sich dafür eingebürgert und wird in sehr vielen Projekten und Templates verwendet. Mittlerweile ist CLEARFIX mehr Container-Begriff für eine Vielzahl an Ansätzen für dieses Prinzip. Alle CLEARFIX-Methoden bauen auf dem Prinzip des Pseudo-Element auf. Es können ::BEFORE und ::AFTER zugewiesen werden. Wir nutzen in der traditionellen CLEARFIX Methode ::AFTER. Es wird so als Inhalt hinter dem Eltern-Element des FLOAT-Inhalt ein Punkt (.) eingefügt der eine neue Zeile erzwingt (display: block.) Der Punkt hat die Eigenschaft CLEAR. Damit man das Element nicht sieht bekommt er eine Höhe von 0 und die Farbe transparent.	`content: ".";` `display: block;` `clear: both;` `height: 0; color: transparent;`	`.clearfix::after {content: ".";` `display: block;` `clear: both;` `height: 0; color: transparent;}`

Flex-Container (display: flex)

code.arnoldbodeschule.de**/flex-im-layout/**
/flexcontainer-elternelement/

Eigenschaft	Bedeutung	Werte
flex-direction:	Positionierung der Kinder entweder auf einer Zeile nebeneinander oder als Säule übereinander. Die FLEX-DIRECTION hat Auswirkungen auf die Eigenschaften. Bei ROW entspricht die Hauptachse der X-Achse. Bei COLUMN entspricht die Hauptachse der Y-Achse.	`row; column; row-reverse; column-reverse;`
flex-wrap:	Erlaubt oder Verbietet den Umbruch in neue Zeile(n) (ROW) bzw. weitere Säulen (COLUMN.)	`nowrap; wrap; wrap-reverse;`
flex-flow:	Praktische Kurzschrift für eine Kombination der Werte von FLEX-DIRECTION und FLEX-WRAP.	`z.B. row wrap-reverse;`
justify-content:	Ausrichtung der Kinder auf der Hauptachse. Das ist bei ROW die X-Achse und bei COLUMN die Y-Achse.	`flex-start; flex-end; center; space-between; space-around;`
align-items:	Ausrichtung der Anfangszeile der Kinder auf der Kreuzachse. Die Kreuzachse ist bei ROW die Y-Achse und bei COLUMN die X-Achse.	`flex-start; flex-end; center; baseline; stretch;`
align-content:	Ausrichtung des Zwischenraum der Kinder auf der Kreuzachse. Die Kreuzachse ist bei ROW die Y-Achse und bei COLUMN die X-Achse.	`flex-start; flex-end; center; space-between; space-around; stretch;`

Flex-Kinder

code.arnoldbodeschule.de**/flex-im-layout/**
/flex-items-kinderelemente/

Eigenschaft	Bedeutung	Werte
order:	Reihenfolge der Flex-Kinder kann unabhängig vom HTML-Markup angepasst werden.	`1; 10; (etc.)`
flex-grow:	Vergrößert das Flex-Kind relativ zu den anderen Flex-Kindern. Standardwert ist 0.	`2;`
flex-basis:	Passt die Breite bzw. Höhe auf den "gewünschten" Wert an. Dabei wird das gewohnte CSS Boxmodell angewendet. Je nach flex-direction wird die Breite (row) bzw. die Höhe (column) angepasst. Dient als Bezugspunkt für flex-shrink. Es ist empfohlen mit der Kurzschreibweise flex: zu arbeiten um -grow, -shrink und -basis zusammenfassend anzugeben.	`40px; auto;`
flex-shrink:	Verkleinert das Flex-Kind relativ zu den anderen Flex-Kindern sobald der Platz nicht ausreicht. Standardwert ist 1.	`4;`
flex:	Praktische Kurzschreibweise für flex-grow, flex-shrink und flex-basis. Dabei sind flex-shrink und flex-basis optional. Der Standardwert ist 0 1 auto,	`3 1 auto;`
align-self:	Individuelles Verhalten des Flex-Kindes. Die möglichen Werte sind von align-items.	`auto; flex-start; flex-end; center; baseline; stretch;`

Flexbox Werte

Eigenschaft	Wert	Bedeutung
flex-direction:	`row;`	Die Flex-Kinder werden hübsch in einer Reihe angeordnet. Sieht auf den ersten Blick aus wie Float. Aber die Höhe kollabiert nicht! Das ist der Standardwert.
flex-direction:	`column;`	Die Flex-Kinder werden vertikal als Säule angeordnet. Entspricht isoliert betrachtet dem Verhalten von display:block Elementen.
flex-direction:	`row-reverse;`	Entspricht dem Wert row mit umgedrehter Reihenfolge.
flex-direction:	`column-reverse;`	Entspricht dem Wert column mit umgedrehter Reihenfolge.
flex-wrap:	`nowrap;`	Die Flex-Kinder werden in einer Zeile angeordnet. Standardwert.
flex-wrap:	`wrap;`	Die Flex-Kinder verteilen falls nötig sich auf mehreren Zeilen.
flex-wrap:	`wrap-reverse;`	Die Flex-Kinder verteilen falls nötig sich auf mehreren Zeilen und werden in umgedrehter Reihenfolge dargestellt.
justify-content: align-items: align-content:	`flex-start;`	Die Flex-Kinder werden oben links beginnend angeordnet. Standard Wert.
justify-content: align-items: align-content:	`flex-end;`	Die Flex-Kinder beginnen am Ende des Flex-Containers.
justify-content: align-items: align-content:	`center;`	Die Flex-Kinder werden im Flex-Container zentriert.
justify-content:	`space-between;`	Die Flex-Kinder werden im Flex-Container zentriert und der Raum zwischen den Kindern wird auf der X-Achseaufgeteilt.
justify-content:	`space-around;`	Die Flex-Kinder werden im Flex-Container zentriert und der Raum zwischen Rand und Kindern sowie zwischen ihnen wird auf der X-Achse aufgeteilt.
align-items:	`baseline;`	Die Flex-Kinder werden registerhaltig auf der Grundlinie ihrer ersten Zeile angeordnet..
align-items: align-content:	`stretch;`	Die Höhe der Flex-Kinder wird an die Höhe des Flex-Containers angepasst. Das ist der Standard. Stretch gilt also immer so lange Du keine andere Eigenschaft zuordnest.
align-items:	`space-between;`	Der Raum der Y-Achse zwischen den Zeilen der Flex-Kinder wird aufgeteilt.
align-items:	`space-around;`	Der Raum der Y-Achse zwischen den Zeilen der Flex-Kinder und zwischen dem Rand des Flex-Containers wird aufgeteilt.
align-content:	`space-between;`	Der Raum der Y-Achse zwischen den Zeilen der Flex-Kinder wird aufgeteilt.
align-content:	`space-around;`	Der Raum der Y-Achse zwischen den Zeilen der Flex-Kinder und zwischen dem Rand des Flex-Containers wird aufgeteilt.

Eigenschaft	Bedeutung	Zentriert	POSITION: (Selbst)	DISPLAY: (Selbst)	WIDTH: (Selbst)	HEIGHT: (Selbst)	POSITION: (Kinder)	DISPLAY: (Kinder)	WIDTH: (Kinder)
margin: 0 auto;	Ist ein Block-Element im Fluss kann es über den AUTO Wert bei MARGIN für links und rechts innerhalb seines Eltern-Elements zentriert werden. Man kann über diese Methode pro Zeile nur 1 Element zentrieren, da Display-Block folgende Elemente in eine neue Zeile zwingt. Damit die horizontale Zentrierung erfolgt, muss die WIDTH kleiner als 100% sein. Es ist nicht möglich vertikal zu Zentrieren.	Selbst	STATIC; RELATIVE;	BLOCK;	<100%	Egal	STATIC; RELATIVE;	Egal	Egal
text-align: center;	Der eigenen Text und alle INLINE* Kinder-Elemente wie z. B. A (Links), SPAN, IMG werden innerhalb von Elementen mit der Eigenschaft TEXT-ALIGN: CENTER; horizontal zentriert. Reicht der Platz in der Zeile nicht für alle Elemente aus, dann beginnt eine neue Zeile.	Eigener Text und INLINE*-Kinder	Egal	Egal	Nein	Egal	STATIC; RELATIVE;	Text; INLINE* (INLINE-BLOCK; INLINE-FLEX ... etc.)	Egal
left: 50%; transform: translateX(-50%);	Elemente, die in ihrer Position nicht im Fluss sind können nur relativ zu ihrem Bezugspunkt zentriert werden. Der Bezugspunkt ist bei ABSOLUTE das letzte Eltern-Element mit position: RELATIVE. Bei FIXED ist der Bezugspunkt immer der BODY. LEFT:50% schiebt das Element um die Hälfte seiner Breite von den linken Rand weg. Bei LEFT muss (!) ein Prozentwert eingegeben werden. Bezugspunkt für Left ist die linke obere Ecke. Daher muss das Element um die Hälfte seiner eigenen Breite "zurückgezogen" werden. Dies geschieht mit TRANSLATEX. Man kann über diese Methode nur 1 Element zentrieren. Eine Reihe von Elementen müssen in einen Container gepackt werden und dann wird dieser mit der Methode zentriert. Ein großer Vorteil dieser Methode ist, dass aufgrund von TRANSLATEX die Breite des Elements nicht bekannt sein muss. Vorsicht: TRANSLATEX ist ein Wert von TRANSFORM und kann somit leicht überschrieben werden. LEFT: auto "deaktivert" den Bezug zum linken Rand und führt, anders als bei MARGIN, nicht zum gewünschten Resultat.	Sich selber	ABSOLUTE; FIXED;	Egal	<100%	Egal	STATIC; RELATIVE; ABSOLUTE;	Egal	Egal
display: flex; justify-content: space-around; align-items: center;	Alle direkten Kinder von display: FLEX können nicht nur horizontal, sondern auch vertikal zentriert werden, unabhängig von ihren eigenen DISPLAY Werten. Sie müssen allerdings im Fluss sein. Außerdem können gegenüber allen anderen Methoden weitere Details angepasst werden. Z. B. die Reihenfolge der Kinder-Elemente kann bestimmt werden. Breiten, Höhen und Zwischenräume können ebenfalls flexibel bestimmt werden. Kinder von FLEX-Containern können für ihre eigenen Kinder ebenfalls FLEX-Containern werden (Nesting).	Kinder	Egal	Egal	Nein	Egal	STATIC; RELATIVE;	Egal	Egal
vertical-align: center;	Innerhalb von TD (Table-Data bzw. Tabellen-Zelle) kann leicht vertikal zentriert werden. Mit der Eigenschaft display: TABLE-CELL kann jedes Element für VERTICAL-ALIGN nutzbar gemacht werden. Jedoch bietet display: FLEX mehr Vorteile, z.B. zentrieren von BLOCK Elementen oder Feinabstimmungen wie das Anpassen des Zwischenraums.	Eigener Text und INLINE*-Kinder	Egal	display: table-cell; (Pflicht)	Nein	Egal	STATIC; RELATIVE;	Text; INLINE* (INLINE-BLOCK; INLINE-FLEX ... etc.)	Egal
top: 50%; transform: translateY(-50%);	Elemente, die in ihrer Position nicht im Fluss sind können auch horizontal relativ zu ihrem Bezugspunkt zentriert werden. Der Bezugspunkt ist bei ABSOLUTE das letzte Eltern-Element mit position: RELATIVE. Bei FIXED ist der Bezugspunkt immer der BODY. Ein großer Vorteil dieser Methode ist, dass aufgrund von TRANSLATEY die Höhe des Elements nicht bekannt sein muss. TRANSLATEY ist ein Wert von TRANSFORM und kann somit leicht überschrieben werden. Natürlich können eine vertikale und horizontale Zentrierung kombiniert werden.	Selbst	ABSOLUTE; FIXED;	Egal	Egal	<100%	STATIC; RELATIVE; ABSOLUTE;	Egal	Egal
left: 50%; margin-left: -200px;	Diese ältere Methode ist der Ursprung der Methode von LEFT:50% und TRANSFORM. Sie hat allerdings den großen Nachteil, dass die Breite der CSS-BOX (Margin, Border, Padding, Width) bekannt sein muss. Diese wird addiert und davon die Hälfte wird MARGIN-LEFT zugewiesen. Jede Anwendung muss also spezifisch konzipiert werden. Die Methode kann allerdings als Fallback für die Kombination mit TRANSFORM nützlich sein.	Selbst	ABSOLUTE; FIXED;	Egal	<100%	Egal	STATIC; RELATIVE; ABSOLUTE;	Egal	Egal

@media Queries

CSS	Bedeutung	Beispiel
@media (...) AND (...) { ... }	Enthält spezifische Angaben für Ausgabegeräte. In runder Klammer werden diese näher bestimmt und können mit Boolschen Attributen wie AND noch feiner definiert werden. Die geschweifte Klammer enthält die Elemente, welche sich ändern sollen. In der geschweiften Klammer werden die Elemente wie in CSS üblich notiert. Am Ende steht also eine doppelte geschweifte Klammer. Die @MEDIA Angaben sollten am Ende der CSS Angaben stehen. Es können beliebige Definitionen aneinandergereiht werden.	`@media (max-width: 400px) AND (orientation: portrait) { h1 {color:#900;}} @media (min-width: 401px) AND (orientation: portrait) { h1 {color:#090;}}`
all	Gilt für alle Geräte-Formen. Ist der Standardwert und muss nicht angegeben werden.	`@media all {...`
print	Gilt für Drucker und druckbezogene Ausgabeformate wie z.B. PDF.	`@media print {...`
screen	Computerbildschirme, Tabletts, Beamer, Smartphones etc.	`@media screen {...`
speech	Screenreader für barrierefreie Webseiten die eine Sprachausgabe unterstützen. Dieses Feature wird langsam umgesetzt.	`@media speech {...`
aspect-ratio	Das Verhältnis von Breite zu Höhe des Browsers.	`@media (aspect-ratio: 4/3) {`
device-aspect-ratio	Das Verhältnis von Breite zu Höhe des Browsers von dem Gerät.	`@media (device-aspect-ratio: 4/3) {`
device-height	Die Höhe von dem Gerät.	`@media (device-height: 600px) {`
device-width	Die Breite von dem Gerät.	`@media (device-width: 500px) {`
height	Die Höhe von dem Browser. Auf einem Geräten kann der Browser unabhängig von der eingestellten Auflösung verändert werden.	`@media (height: 600px) {`
max-aspect-ratio	Das maximale Verhältnis von Breite zu Höhe des Browsers.	`@media (max-aspect-ratio: 4/3) {`
max-device-aspect-ratio	Das maximale Verhältnis von Breite zu Höhe von dem Gerät.	`@media (max-device-aspect-ratio: 10/3) {`
max-device-height	Maximale Höhe von dem Gerät.	`@media (max-device-height 600px) {`
max-device-width	Maximale Breite von dem Gerät.	`@media (max-device-width: 950px) {`
max-height	Maximale Höhe des Browsers.	`@media (max-height: 900px) {`
max-resolution	Maximale Auflösung.	`@media (max-resolution 3) {`
max-width	Maximale Breite des Browsers.	`@media (max-width: 950px) {`
min-aspect-ratio	Das minimale Verhältnis von Breite zu Höhe des Browsers.	`@media (min-aspect-ratio: 2/3) {`
min-device-aspect-ratio	Das minimale Verhältnis von Breite zu Höhe von dem Gerät.	`@media (min-device-aspect-ratio: 2/4) {`
min-device-width	Die minimale Breite von dem Gerät.	`@media (min-device-width: 600px) {`
min-device-height	Die minimale Höhe von dem Gerät.	`@media (min-device-height: 600px) {`
min-height	Die minimale Höhe von dem Browser.	`@media (min-height: 600px) {`
min-resolution	Die minimale Auflösung.	`@media (min-resolution: 600px) {`
min-width	Die minimale Breite des Browsers.	`@media (min-width: 600px) {`
orientation	Landscape (Querformat) oder Portrait (Hochformat)	`@media (orientation: landscape) {`
width	Die Breite von dem Browser.	`@media (width: 600px) {`
AND	Es können beliebige Media-Eigenschaften mit AND aneinandergereiht werden.	`@media (min-width 400px) AND (max-height 950px) {`
NOT	Es können beliebige Media-Eigenschaften mit NOT ausgeschlossen werden.	`@media (min-width 400px) NOT (orientation: landscape) {`

Transitition

code.arnoldbodeschule.de/**transition/**

Eigenschaft	Werte	Bedeutung	Übersetzung	Beispiel
transition:	`all 1s width 2s ease, color 0.5s linear;`	Kurzschreibweise für delay duration property und timing function	Übergang	`a {transition: 0.5s color 3s; color: #000} a:hover {color: #fff}`
transition-delay:	`0.4s`	Verzögerung des Übergangs	Übergang-Verzögerung	`transition-delay:`
transition-duration:	`0.6s`	Dauer des Übergangs	Übergang-Dauer	`transition-duration:`
transition-property:	`all, color, background-color (...)`	Angabe auf welche CSS Eigenschaft der Übergang angewendet wird.	Übergang-Eigenschaft	`transition-property: line-height;`
transition-timing-function:	`ease, linear, ease-in, ease-out, ease-in-out, cubic-bezier(n,n,n,n)`	Tempokurve des Übergangs. Z.B. langsamer Start und schnelles Ende oder anders herum.	Übergang-Zeitdauer-Funktion	`transition-timing-function: ease-in`

Animation

code.arnoldbodeschule.de/**animation/**

Eigenschaft	Werte	Bedeutung
animation-name:	`Spinning-Arnold`	Name
animation-duration:	`24s; 0; 8s;`	Dauer
animation-timing-function:	`ease; linear, ease-in; ease-out; ease-in-out; cubic-bezier(n,n,n,n);`	Beschleunigung der Animation
animation-delay	`24s; 0; 8s;`	Verzögerung des Starts
animation-iteration-count:	`1; 5; infinite`	Durchgänge der Animation
animation-direction:	`normal; reverse; alternate; alternate-reverse`	Abspielrichtung, z.B. vorwärts und dann rückwärts etc.
@keyframes	`0% {css: value; ... } 100% {css: value; ...}; from {css: value; ... } to {css: value; ... };`	Beliebig viele Keyframes der Animation
animation:	`Spinning-Arnold 5s infinite;`	Kurzschreibweise für die Animations Eigenschaften. (name, duration, timing, delay, iteration, direction)
animation-play-state:	`running; paused;`	Abspielen oder Pause

Werte für timing-function

code.arnoldbodeschule.de/**animation/**

Eigenschaft	CSS-Werte	Bedeutung	Beispiel
timing-function:	`ease`	Langsamer Start, schneller Mittelteil, langsames Ende. Ist der Standardwert falls keine anderen Timing Functions genutzt werden.	`transition: all ease 1.0s`
timing-function:	`linear`	Durchgängig gleichmässiges Tempo.	`transition: all linear 1.2s`
timing-function:	`ease-in`	Langsamer Start und dann zunehmend schneller.	`transition: all ease-in 1.0s`
timing-function:	`ease-out`	Schneller Start und dann zunehmend langsamer.	`transition: all ease-out 1.0s`
timing-function:	`ease-in-out`	Wirkt insgesamt gegenüber ease langsamer. Verzögert Start und Endposition und beschleunigt im Mittelteil nicht so stark.	`transition: all ease-in-out 1.0s`
timing-function:	`cubic-bezier (n,n,n,n)`	Eigene Tempokurve des Übergangs.	`transition: all cubic-bezier(0.550, 0.055, 0.675, 0.190) 2.2s`

Wert	Bedeutung	Beispiel
blur(px),	Weichzeichner mit Stärke in einer beliebigen CSS Längeneinheit wie z.B. PX, CM, VMAX oder REM. Der Wert 0 ist die normale Darstellung. Die Vererbung von BLUR kann je nach Browser unterschiedlich statt finden. BLUR zeichnet in der Regel alle Kind-Elemente weich aber nicht immer. Es ist mitunter empfehlenswert das Element mit der Eigenschaft BLUR in einen zusätzlichen Container (Wrap) zu packen, damit die Rahmenkonturen scharf bleiben.	`filter: blur(4px),`
brightness(%),	Helligkeit des Element. 0 ist völlig schwarz. 1 ist normal. 2 ist doppelt so hell. Die Helligkeit wird linear berechnet.	`filter: brightness(2),`
contrast(%);	Kontrast des Elements. 0 ist ohne jeglichen Kontrast. 1 ist normal und 2 ist doppelter Kontrast.	`filter: contrast(1.5);`
drop-shadow(...);	Schlagschatten des Elements. Verhält sich auf den ersten Blick ähnlich wie Boxshadow. Allerdings werden Pfade und Alphakanäle erkannt. Es wird mit 3 Längenwerten (X,Y,Weichzeichner) und 1 Farbwert definiert.	`filter: (0 4px rgba(0,0,0,0.2));`
grayscale(%);	Endsättigung des Elements. 0 Ist normale Wiedergabe. 1 ist Graustufenversion.	`filter: grayscale(0.8);`
hue-rotate(deg);	Farbton-Rotation mit Gradangabe. Farbton wird entsprechend im Farbkreis verschoben.	`filter: hue-rotate(40deg);`
invert(%);	Negative Umkehrung der Farben des Elements. 0 Ist die normale Wiedergabe. 0.5 erzeugt eine graue Fläche und 1 ist negative Farbwiedergabe.	`filter: invert(0.9);`
opacity(%);	Deckkraft des Element. Verhält sich wie die CSS Eigenschaft Opacity.	`filter: opacity(0.2);`
saturate(%);	Farb-Sättigung des Elements. 1 ist normale Wiedergabe. 0 ist ein Graustufenbild. 2 ist doppelte Sättigung.	`filter: saturate(1.5);`
sepia(%);	Sepia-Farbeffekt auf das Bild. 0 ist normale Wiedergabe. 1 ist vollständiger Sepia-Effekt.	`filter: sepia(0.8);`
sepia(%) saturate(%) blur(px);	Es können beliebig viele Filter in Kombination hintereinander geschaltet werden. Dabei sollte die Performance des Wiedergabegerätes beachtet werden.	`filter: sepia(0.5) saturate(2) blur(10px);`
url();	Es können SVG Filter über die Angabe der URL angewendet werden.	`filter: url();`
none;	Filter wird deaktiviert.	`filter: none;`
initial;	Standard-Ausgangswert.	`filter: initial;`
inherit;	Erbt den Effekt des Eltern-Elements.	`filter: inherit;`

LS 02 **Hallo Welt mit HTML**

Verfasse den HTML und CSS Code zuerst auf dem Papier. Danach am Computer testen.

Element	Aussehen
Überschrift 1. Kategorie	Arial, 48px, fett, #f00;
Breite/Höhe/Farbe	500px/350px/#bbb
Hintergrund	#999
Text	Arial, 14px, #000;
Geordnete Liste	Arial, 18px, #646464;
2 horizontale Linien	5px hoch, #f00;

Beispielhafte Lösung

```html
<!DOCTYPE html>
<head>
<title>Simple Site</title>
<link rel="stylesheet" type="text/css" href="design.css">
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
</head>
<body>
    <main>
        <h1>Hallo Welt</h1>
        <p>Das ist eine einfache Webseite mit der üblichen Struktur:</p>
        <ol>
            <li>HTML</li>
            <li>Head</li>
            <li>body</li>
        </ol>
        <hr>
        <hr>
    </main>
<body>
</html>
```

```css
body    {background-color: #999;}

h1      {font-size: 48px; font-weight: bold; color: #f00;}

p       {font-size: 14px; color: #000;}

ol li   {font-size: 18px; color: #646464;}

hr      {height: 5px; color: #f00; background-color: #f00; border-style: solid;}

main    {width: 500px; height: 350px; background-color: #bbb; margin: 0 auto;}
```

Geschichte von Javascript

Brendan Eich, Mozilla Corporation
Taken by AcidJazzed cc SA Lizenz.

Javascript, nicht mit der Sprache Java zu verwechseln, wurde 1995 von Brendan Eich in 10 Tagen entwickelt. Diese kurze Entwicklung merkt man der Sprache noch heute an, meinen manche Kritiker.

Javascript wird für dynamische Interaktion im Browser eingesetzt. Dies kann auch in Echtzeit geschehen. Javacsript ist im Gegensatz zu den Auszeichnungs-Sprachen HTML und CSS eine echte Programmiersprache.

Javascript hat sich in in den letzten 10 Jahren rasant entwickelt und ist heute die am häufigsten genutzte Programmiersprache weltweit

Seit 1998 findet eine Standartisierung in Form von ECMAScript statt. Besonders Populär sind Javascript Frameworks bzw. Librarys wie JQuery, React oder Angular JS. Es gibt auch auf Javascript basierende Server wie Node JS.

HTML und Javascript

Javascript wird in HTML über den <srcipt> eingbettet. Die Funktionen von Javascript sind im Browser eingebaut. Die Verbindung ist hoch. So gibt es eine Reihe von HTML5 API (z.B. Video oder Audio), der <output>, <canvas> oder Attribute in <form> die direkt mit Javascript angesprochen werden.

Typische Anwendungsgebiete von JavaScript

- dynamische Manipulation von Webseiten über das Document Object Model. Z.B. das Hinzufügen von Klassen oder das Entfernen von Klassen. Sortieren von Tabellen. Austausch von Inhalten.
- Plausibilitätsprüfung von Formulareingaben vor der Übertragung zum Server
- Anzeige von Dialogfenstern
- AJAX: Senden und Empfangen von Daten in Echtzeit (ohne dass der Nutzer die Seite neu laden muss)
- Vorschlagen von Suchbegriffen während der Eingabe
- Schreib- und Lesezugriff auf Cookies und den Web Storage innerhalb des Browsers
- Produktion von Progressive Web-Apps
- Produktion nativer Apps

```
// ! Definition globale Variable
var button_menu = document.getElementById('button-menu');

function toggle_menu() {
  var menu = document.getElementById('menu');      // ! Definition lokale Variable
      button_menu.classList.toggle('close-menu'); // ! globale var
      button_menu.classList.toggle('click-menu');
      menu.classList.toggle('hide-menu');          // ! lokale var
      menu.classList.toggle('show-menu');
  }

window.onload = function() {
  button_menu.onclick = toggle_menu;
  }
// ! Event-Handling von Event (.onclick)
// ! mit Assigment (= (Übertrag))
// ! zu einer einer Function (toggle_menu).
// ! Nutzt globale Variable und Name der Function
```

http://code.arnoldbodeschule.de/einstieg-in-javascript/

Sonderfunktionen mit <script>

Element	Semantik	Beschreibung
<script>	Client-side script	Container für den client-seitigen script. Dieser liegt in einer anderen Sprache, z.B. javascript, vor. Ein weiteres populäres Beispiel ist das Javascript Framework J-Query. Die Scripte sind für Aktionen da, die mit HTML und CSS allein nicht möglich sind.

<script> und andere dynamische Elemente wie SQL oder PHP sind in diesem Skript nur als Überblick gedacht. Schwerpunkt soll Konzeption und Umsezzung von Webseiten mit HTML und CSS sein. Es ist allerdings gut über den Tellerrand zu blicken und zu wissen was es sonst so gibt.

Mit <script> lassen sich Funktionen ausführen, die sonst nicht möglich wären.

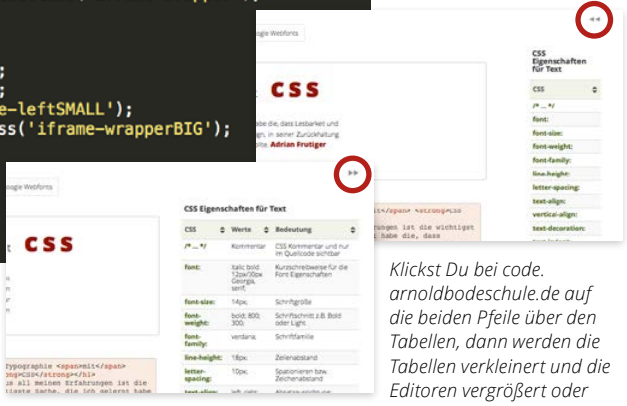
Besonders interssant sind Abfolgen die an Bedinungen knüpfen. „wenn das so ist, dann machst Du das. Sonst machst Du das ...".

Attribute für <script>

Attribut mit Beispiel	Bedeutung
src="..." <script src="jquery-1.11.3.min.js"></script> <script src="toggler.js"></script>	Lädt den script aus einer externen Quelle. Z.B. das Framework jQuery. Auf dieses Framework kann sich dann anschließend bezogen werden. Natürlich kann man den Javascript auch selber verfassen und in externen Quellen ablegen ohne sich auf Framworks zu beziehen.
type="..." <script type="text/javascript" src="toggler.js" ></script>	Es sollte der Typ des Scriptes angegeben werden.

Beispiel jQuery

```
<script type="text/javascript">
jQuery(document).ready(function($) {
  $(".ToggleInfo").click(function(){
    if($(this).hasClass('ToggleInfoSMALL')) {
      $('.awayC2').removeClass('awayC2').addClass('column-2');
      $('.awayC3').removeClass('awayC3').addClass('column-2');
      $('.table-leftSMALL').removeClass('table-leftSMALL').addClass('table-left');
      $('.iframe-wrapperBIG').removeClass('iframe-wrapperBIG').addClass('iframe-wrapper');
      $('.ToggleInfoSMALL').removeClass('ToggleInfoSMALL');
      $('.bubble-wrap').animate({left: '305px'});
    } else {
      $('.column-2').removeClass('column-2').addClass('awayC2');
      $('.column-3').removeClass('column-3').addClass('awayC3');
      $('.table-left').removeClass('table-left').addClass('table-leftSMALL');
      $('.iframe-wrapper').removeClass('iframe-wrapper').addClass('iframe-wrapperBIG');
      $('.ToggleInfo').addClass('ToggleInfoSMALL');
      $('.bubble-wrap').animate({left: '375px'});
    }
  });
});
</script>
```



Klickst Du bei code.arnoldbodeschule.de auf die beiden Pfeile über den Tabellen, dann werden die Tabellen verkleinert und die Editoren vergrößert oder wieder in die Ausgangsgröße gebracht. Das passiert mit jQuery.

Beispiel Javascript

Während z.B. Chrome bei Formularen aufgrund des HTML Attributes „required" von alleine eine Fehlermeldung anzeigt, muß z.B. SAFARI nachgeholfen werden. In dem Beispiel Form2Mail wird dafür dieser Javascript benutzt. Die Datei check.js wird geladen.

```
<head>
    <meta charset='UTF-8'>
    <title>Form2Mail</title>
    <link href='http://fonts.googleapis.com/css?family=Open+Sans:300,600' rel='stylesheet' type='text/css'>
    <link rel="stylesheet" href="style.css">

    <script language="javascript" type="text/javascript" src="check.js"></script>

</head>
```

Oben ist Chrome und unten ist Safari.

Zunächst wird x definiert: Es ist in dem Formular Form2Mail der Wert von email. Dann wird festgelgt dass wenn dieses Feld leer ist, dann kommt eine Fehlermeldung. Ansonsten passiert nix und das Formular wird normal weiterverarbeitet.

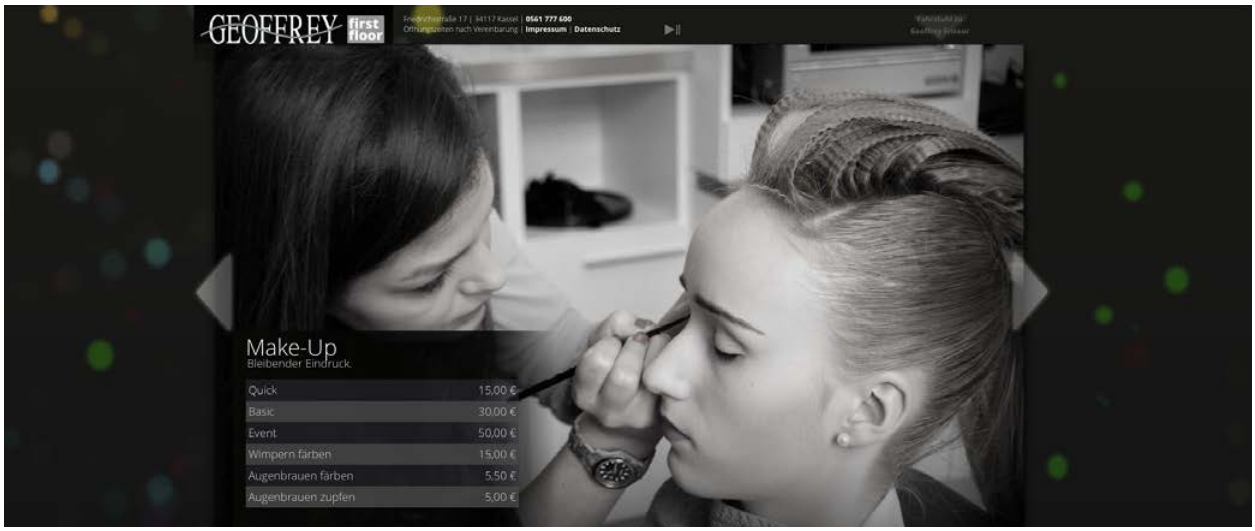
```
function validateForm()
{
    var x = document.forms["Form2Mail"]["email"].value;
        if (x == null || x == "") {
            alert("Bitte E-Mail Adresse angeben");
            return false;
    }
}
```

Die Leinwand mit <canvas>

Element	Semantik	Beschreibung
<canvas>	Leinwand	Container für die Ausgabe von Grafiken die mit Scripts, z.B. Javascript, verfasst werden.

Attribute für <canvas>

Attribut mit Beispiel	Bedeutung
id=“” **<canvas id=“Canvas1“></canvas>**	Die ID ist wichtig um dem Script zu sagen dass die Ausgabe der Grafik genau hier statt findet-
width=“...“ **<canvas id=“Canvas1“ width=“90px“ ></canvas>**	Die Breite kann entweder über CSS oder direkt als Attribut eingegeben werden.
heigth=“...“ **<canvas id=“Canvas1“ width=“90px“ heigth=“90px“ ></canvas>**	Die Höhe kann entweder über CSS oder direkt als Attribut eingegeben werden.



Geoffrey Orth, Seeliger, CC-BY-NC-ND-4

```
<div class="canvas">
<canvas id="mainCanvas"></canvas>
</div>
```

Auf der Landing-Page Geoffrey First Floor ist <canvas> eingesetzt. Jeder Klick erzeugt Effekte im Hintergrund die randomisiert gefärbt sind und heurmschwirren.

```
particleList = {};
recycleBin = {};

//random acceleration factors - causes some random motion
randAccelX = 0.1;
randAccelY = 0.1;
randAccelZ = 0.1;

gravity = 0; //try changing to a positive number (not too large, for example 0.3), or negative for floating upwards.

particleRad = 2.5;
sphereRad = 200;
sphereCenterX = 0;
sphereCenterY = 0;
sphereCenterZ = -3 - sphereRad;

//alpha values will lessen as particles move further back, causing depth-based darkening:
zeroAlphaDepth = -750;

turnSpeed = 2*Math.PI/1600; //the sphere will rotate at this speed (one complete rotation every 1600 frames).
turnAngle = 0; //initial angle

timer = setInterval(onTimer, 1000/24);

ction onTimer() {
//if enough time has elapsed, we will add new particles.
count++;
    if (count >= wait) {

    count = 0;
    for (i = 0; i < numToAddEachFrame; i++) {
        theta = Math.random()*2*Math.PI;
        phi = Math.acos(Math.random()*2-1);
        x0 = sphereRad*Math.sin(phi)*Math.cos(theta);
        y0 = sphereRad*Math.sin(phi)*Math.sin(theta);
        z0 = sphereRad*Math.cos(phi);

        //We use the addParticle function to add a new particle. The parameters set the position and velocity components.
        //Note that the velocity parameters will cause the particle to initially fly outwards away from the sphere center (a
        //it becomes unstuck).
        var p = addParticle(x0, sphereCenterY + y0, sphereCenterZ + z0, 0.002*x0, 0.002*y0, 0.002*z0);
```

Im <canvas> Element mit der ID mainCanvas wird der unten aufgeführte Javascript ausgeführt. Okay, zugegeben es ist jetzt vielleicht nicht selbsterklärend was dort passiert. Aber es ist nun einmal die einzige tasächliche Anwendung von <canvas> von mir überhaupt. Diese nicht minimierte Version basiert auf der Grundlage des Tutorial: „Dustysphere“ von http://www.rectangleworld.com.

Javascript	Bedeutung	Beispiel
function name(parameter) {aktion();};	Eine Javascript FUNCTION ist ein CODE-Abschnitt, der für das Ausführen einer Aufgabe konzipiert ist. Die Funktion wird ausgeführt, wenn sie von einem Event aufgerufen wird. Der Aufbau ist: Das Keyword FUNCTION kann gefolgt werden von dem individuellen Namen und hat eine runde Klammer (). Eine FUNCTION ohne Name wird anonyme Function genannt. In der runden Klammer können mit Komma getrennte Parameter der Funktion angegeben werden. Es folgt eine geschweifte Klammer mit den Bestand-teilen der FUNCTION. Diese Bestandteile werden mit einem Semikolon abgeschlossen. Eine FUNCTION kann weitere FUNCTIONS enthalten. Die Benennung von FUNCTIONS muss sich zum einen an die Regeln halten und zum andere sollte sie nach einem Projektübergreifenden Konzept geschehen. Idealerweise entwickelt man bei der Arbeit langsam eine Repertoire an FUNCTIONS für unterschiedliche Anwendungsfälle.	`window.onload = function() {document.getElementById('button'). onclick = start;}`
Event-Handler	Ein Javascript Event wie .onclick ist mit dem Assigment-Operator (=) einer FUNCTION zugeordnet. Dies registriert den so bestimmten Event-Handler und wenn dieses Event geschieht, dann feuert (startet) die FUNCTION. Event-Handler bestehen in dieser Form seit den ersten Javascript-Tagen und werden von jedem Browser verstanden. Es besteht eine 1:1 Beziehung zwischen Event und FUNCTION. Allerdings können in einer FUNCTION weitere FUNCTIONS gesammelt werden. Es ist wichtig die FUNCTION OHNE runde Klammer () im Event-Handler zu registrieren. Die FUNCTION wird nur OHNE runde Klammer bei dem Event ausgeführt. Mit runden Klammern entsteht ein Fehler: Die FUNCTION wird einmal ausgeführt und das Ergebnis registriert. Event-Handler sind keine Event-Listener. Event-Listener sind neuer, übernehmen vergleichbare Aufgaben und können in beliebiger Menge einem Event gleichzeitig zugeordnet werden.	`document.getElementById('image'). onclick = toggle _ mobile _ menu;`
var name = document. getElementById('id-name');	VAR ist eine Variable und wird als Platzhalter für die mit dem Gleichzeichen (=) gegenüber-gestellten Ausdruck in der FUNCTION genutzt. Mehrfach genutzte Elemente sollten mittels Variable bestimmt werden. VAR hilft auch FUNCTIONS für andere Anwendungsfällen anzupassen. Die FUNCTION wird dann nicht verändert, die VAR werden jedoch an das jeweils relevante DOM (Document Object Modell) angepasst. Eine Variable kann z.B. ein mit ID bestimmtes Element sein oder eine komplette FUNCTION. Eine Variable kann auch ein Teilstück einer mathematischen Operation sein. Es gibt globale Variablen und lokale Variablen.	`var Audio = document. getElementById('audio-player');`
Globale Variable	Auf die globalen Variablen kann jede FUNCTION zugreifen, sie können überall verwendet werden. Indem die Variable außerhalb von FUNCTIONS definiert wird, wird sie global. Die Variable muss in dem Fall nicht mit dem Zusatz "var" eingeleitet werden. Bei kleinen Projekten kann dies praktisch sein, jedoch können Namenkonflikte bei komplexeren Projekten oder im Zusammenspiel mit Frameworks auftreten. Insofern sind in einer FUNCTION lokal definierte Variablen zu empfehlen.	`variable1 = "1"; var variable2 = "Test"; // Außerhalb einer FUNCTION`

Javascript	Bedeutung	Beispiel
Lokale Variable	Lokale Variable wird innerhalb einer FUNCTION bestimmt und mit "var" eingeleitet. Obwohl es auf der einen Seite Wiederholung (redundanten) Quellcode bedeute, so ist die Gefahr von Namenkonflikten bei lokalen Variablen gegenüber globalen Variablen überschaubarer.	`var variable2 = "Test"; // Innerhalb einer FUNCTION`
// Javascript Kommentar	Innerhalb des Javascript können einzelne Zeilen mit doppelten Slash // auskommentiert werden.	`var Audio = document.getElementById('audio-player'); //ffooo!!!`
// Regeln und Tipps für Namen	Keine Leerzeichen, Buchstaben und Ziffern, das erste Zeichen muss ein Buchstabe sein, Groß- und Kleinschreibung werden unterschieden, keine deutschen Umlaute wie ß oder ä, das einzige erlaubte Sonderzeichen ist Unterstrich '_'. Hyphens sind demnach VERBOTEN: (button-menu). Underscore ist ERLAUBT: (button_menu). CamelCase ist ERLAUBT: (ButtonMenu, buttonMenu). Es dürfen keine Javascript-Keywords bzw. reservierte Wörter benutzt werden (z.B. var). Wichtig: Die Bedeutung sollte sich auch nach Monaten noch ergeben. Lieber etwas ausführlicher schreiben und sehr ausführlich den eigenen Code kommentieren! Am Wichtigsten ist sicherlich die schnelle Fortführung der Arbeit am Code. Das Prinzip der Namensgebung und der modularen Zusammensetzung sollte sich auch im CSS widerspiegeln. Längerer Quellcode auf Basis konsistenter Bezeichnungen lässt sich gut mit GZIP komprimieren und bewirkt letztlich eine schnelle Datenübertragung.	`// button _ menu, ButtonMenu oder buttonMenu`
if	Wenn die in runden Klammern definierte Bedingung erfüllt ist, dann wird der in der geschweiften Klammer aufgeführte Code ausgeführt. Es ist wichtig im Javascript IF (Wenn) in Kleinbuchstaben zu schreiben.	`if (Audio.paused) {Audio.play();}`
else	Gilt wenn keine IF Bedingung zutrifft. Es wird der bei ELSE (Ansonsten) in der geschweiften Klammer bestimmte Code ausgeführt. Auch der Befehl ELSE muss im Javascript in Kleinbuchstaben geschrieben werden.	`else {(Audio.pause);}`
window.onload	window.onload startet (feuert) die FUNKTION sobald das Laden aller Bestandteile, Z.B. CSS, Texte und Bilder abgeschlossen ist.	`window.onload = function Party _ ON() { ... }`
.onclick	ONCLICK ist der Mausklick-Event.	`document.getElementById('button-map').onclick = function Show _ Map() { ... }`
.onmouseover	Event des Massekontakt mit dem Element. Entspricht dem CSS Hover.	`document.getElementById('timebomb').onmouseover = function booom() { ... }`
.onkeyup	Event wenn eine Taste losgelassen wird	`<input type="text" onkeyup="coolFunction()">`
.onkeypress	Der Event bei einem Tastendruck. Gilt nicht für alle Tasten (z.B. ALT, CTRL, SHIFT, ESC).	`document.getElementById('input-name').onkeypress = function hello() { ... }`
.onkeydown	Event bei einem Tastendruck. Gilt für alle Tasten.	`<input type="text" onkeydown="coolFunction()">`
document.getElementById('id-name');	Das Element mit der in der runden Klammer bestimmten ID wird ausgewählt. Über eine ID kann ein Element eindeutig identifiziert werden.	`document.getElementById('button-map')`
.classList.toggle('class-name')	TOGGLE prüft ob die in der runden Klammer genannte CSS-CLASS vorhanden ist, dann wird die CLASS entfernt. Ist die CLASS nicht vorhanden, dann wird sie hinzugefügt.	`function beat _ animation _ toggle() {document.getElementById('beat-button').classList.toggle('beat-animation');}`
.classList.add('class-name')	Fügt eine CSS-Class hinzu.	`document.getElementById('header').classList.add('green');`
.classList.remove('class-name')	Entfernt die CSS-Class.	`document.getElementById('header').classList.remove('green');`
=	Das Gleichzeichen (=) ist der Assigment-Operator. Assigment bedeutet "zuweisen, übertragen". Z.B. können damit Variablen bestimmt werden oder EVENT-HANDLER können erstellt werden indem Javascript-Events wie .onclick auf FUNCTIONS übertragen werden.	`document.getElementById('image').onclick = toggle _ mobile _ menu;`

Javascript	Bedeutung	Beispiel
.play()	Startet das Abspielen.	`Audio.play();`
.pause()	Pausiert das Abspielen.	`Audio.pause();`
.paused()	Pausierter Zustand.	`if (Audio.paused) { ...`
.played()	Abspielender Zustand.	`if (Audio.played) { ...`
.volume	Lautsärke als Zahl eingestellt. 0 ist Ton aus. 1 ist normale Lautstärke. 1.5 ist um 50% lauter.	`Audio.volume = 0.3;`
.loop	Dauerschleife true (ja) oder false (nein).	`Audio.loop = true;`
.currentTime	Setzt den Startpunkt für das Abspielen.	`Audio.currentTime = 5;`
.progress	Feuert eine FUNCTION während des Ladens des Mediums.	`.progress = function() {alert("Download");};`
.onended	Feuert die FUNCTION das Ende des Mediums erreicht ist.	`.onended = function() {alert("Das Ende ist gekommen");};`

<!--CSS-->

```css
html {
  background: radial-gradient(#000,#333); min-height: 100%;
  padding: 4% 0; margin: 0; overflow: hidden;}

.box {
  display: block; height: 50%; width: auto; margin: 0 auto;
  transition: opacity 0.2s; opacity: 0.5}

.box:hover {opacity: 1}
.beat-animation {
  animation: pulse 0.5s ease-in-out alternate infinite; opacity: 1}

@keyframes pulse {
  0%    {transform: scale(1);}
  100%  {transform: scale(1.12) translateY(1.12%);}}
```

<!--HTML-->

```html
<img id='beat-button' title='click for beat'
       class='box' src='http://codedata.absks.de/148969.svg'/>
<audio id='audio-player' loop preload='auto'>
   <source src='http://codedata.absks.de/CODETube_API.ogg' type='audio/ogg' />
   <source src='http://codedata.absks.de/CODETube_API.mp3' type='audio/mp3' />
   <!-- ! Loop by Thomas Erszeny
        https://www.facebook.com/teddpittbeats/ -->
</audio>
```

<!--Javascript-->

```javascript
function audio_play_pause() {                          // ! Function
  var Audio = document.getElementById('audio-player');  // ! Variable
    if (Audio.paused) {                                 // ! AV API Methode
        Audio.play();
        Audio.volume = 0.3;                             // ! AV API Property
        Audio.loop = true;
  } else {
      Audio.pause();                                    // ! AV API Methode
  }}

function beat_animation_toggle() {                     // ! Function
  document.getElementById('beat-button').classList.toggle('beat-animation');
  }

function audio() {                                     // ! Set of Functions
  audio_play_pause();
  beat_animation_toggle();
  }

window.onload = function() {                           // ! Anonyme Function
  document.getElementById('beat-button').onclick = audio;
  }
```

Farbgeschichte

Menschen teilen Informationen seit Anbeginn ihrer Geschichte mit Farben und über Farben weiter. Im Laufe der Generationen untersuchten Religiöse, Künstler und Wissenschaftler Farben und schrieben den Farben Bedeutungen zu und erkannten archetypische Harmonien und Gesetze. Als Kinder ihrer Zeit sind diese Untersuchungen durch den Zeitgeist geprägt und diese Gemeinsamkeit haben auch wir mit den Altvorderen gemeinsam. Unsere heutige CSS Technik baut auf dieser Geschichte der Farben auf.

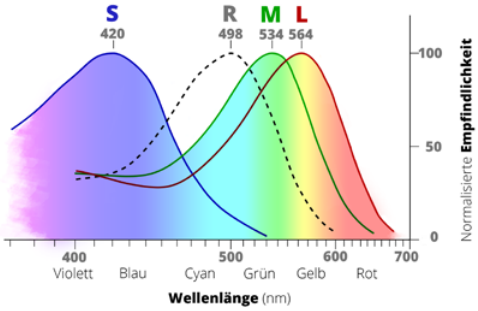
Aristoteles prägte spätere Epochen mit seiner Philosophie und Sicht der Farben. Goethe und Schopenhauer systematisierten Farbkreise[1] über die komplementären Farben. Aga kombinierte in der Blauen Moschee meisterhaft Licht, Architektur und Baustoff. Newton spaltete weißes Licht mit einem Prisma in die Spektralfarben auf und entdeckte, dass diese Farben übereinander projiziert zusammen erneut weißes Licht[2] ergeben. Itten formulierte die Farbgesetze und Farbkontraste[3].

Grundlage für alle Farbdefinitionen, die wir im CSS benutzen werden, ist die Arbeit von David und Guild aus dem Jahr 1931 für die internationale Farbkomission. Beide bestimmten den sogenannten Normalbetrachter, indem Sie mit roten, grünen und blauen Filtern vor Quecksilberdampflampen die sichtbaren Farben mischten und die Mischverhältnisse sowie derenWahrnehmung dokumentierten.

1 *Siehe* **Farbakkorde komponieren und Geschichte der Farbkreise**
2 *Siehe* **Additives Farbmischsystem RGB**
3 *Siehe* **Farbkontraste**

Farbphysiologie, Licht und Sehen

Farbphysiologie beinhaltet die physikalischen Aspekte des Lichtes und des Sehens. Unsere Sonne strahlt ihre Energie in den Weltraum und einen Teil dieses Energiespektrums nehmen wir als sichtbares Licht wahr. Nicht nur unsere Augen sondern der komplette Planet hat sich auf die Sonne als Energielieferanten eingestellt. So sind z.B. die Pflanzen in der Regel Grün, um einen höheren Ertrag aus dem Licht zu ernten, da der grüne Teil des Spektrums im Gegensatz zu den anderen Bereichen im Laufe des Tages etwas größer ist.



Jede Tierart hat sich unterschiedlich angepasst und bei unseren Augen sind drei Arten von Zäpfchen (S, M, L) für das Farbsehen und eine Art Stäbchen (R) für Wahrnehmung bei geringer Helligkeit zusammen mit dem Gehirn für das Sehen wichtig. Die Zäpfchen reagieren entweder für den eher kurzen (S: Short), mittleren (M: Medium) oder langen (L: Long) Bereich des Spektrums. Die Stäbchen (R: Rod Cells) werden bei Tageslicht vom Gehirn mehr oder weniger ignoriert, bei wenig Licht sind sie jedoch noch dann sensibel, wenn die S, M und L-Zäpfchen nicht mehr stimuliert werden. Die Stäbchen im Vergleich zu den Zäpfchen sind für Helligkeit in etwa dreimal so sensibel. Hinzu kommt die Fähigkeit des Gehirns durch zusätzliche Regulation einen recht hohen Dynamik-Umfang an Tonwerten zu erzeugen.

»Farbe ist nicht im Licht,
nicht im Auge, sondern im Hirn.«

Isaac Newton, 17. Jahrhundert.

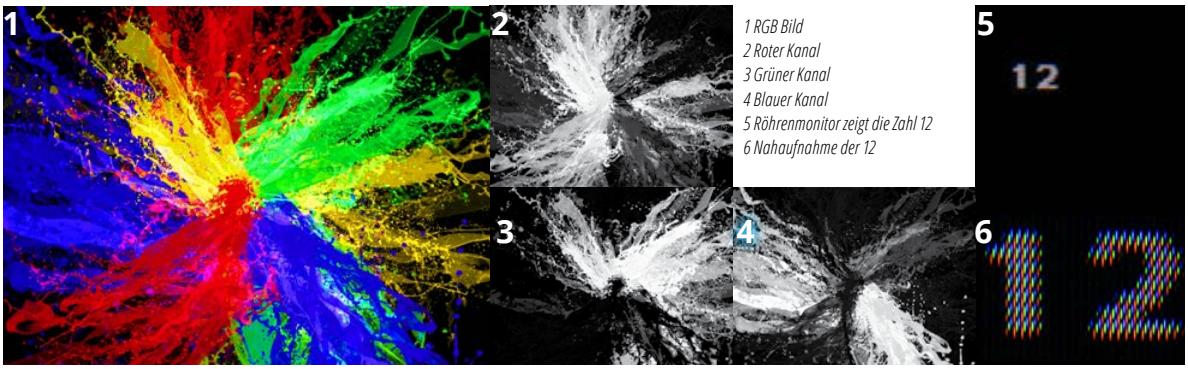
Bereits Newton hat es richtig gedeutet, seitdem ist das Prinzip Farben-Sehen immer weiter untersucht worden. Heute kann nachgewiesen werden: Die Zäpfchen unserer Augen sind entweder auf den kurzwelligen bläulichen Bereich (S), den mittleren cyan-grünlichen (M) oder den längeren gelb-grüm-rötlichen (L) Bereich sensibel. Das Gehirn verarbeitet diese Reize dann zu Farben. Die Wellenbereiche der Stäbchen entsprechen nicht der reinsten Sättigung von Blau, Grün und Rot. 534 ist eher Cyan -Grün und 564 Grün-Gelb-Orange

Farbmetrik und additives Farbmischsystem RGB

Die Farbmetrik ist die Lehre von den Maßbezeichnungen der Farben. Sie stellt mittels mathematischer Formeln das visuelle Ergebnis einer Farbe dar. Im CSS benutzen wir in erster Linie das additive Farbmischsystem RGB. Die drei Grundfarben Rot, Grün und Blau addieren sich zu Weiß. Das RGB Farbmischsystem ist die Grundlage für den großen Teil der Ausgabegeräte von Webseiten wie Beamer, Computermonitore, sonstige Bildschirme und Displays. Auch erfassen die Eingabegeräte für Bilder, Digitalcameras und Scanner, ihre Daten in RGB. Die Technik orientiert sich an den Eigenschaften des menschlichen Auges, allerdings sind die verwendeten Farben gegenüber den sensibelen Bereichen der Stäbchen verschoben und trennen Rot, Grün und Blau eindeutig. RGB entspricht ziemlich genau den Stäbchen von Goldfischen, weniger den Stäbchen von Menschen. Funktioniert aber auch bei uns offensichtlich sehr gut.

RGB Farbumfang und Farbwiedergabe

Heutige Monitore können problemlos den typischen RGB Farbumfang von 16,7 Millionen Farbabstufungen wiedergeben. Dies entspricht einer Abstufung von 256 Farben (bzw. 8 Bit) pro Grundfarbe Rot, Grün oder Blau. Allerdings ist die Farbwiedergabe von RGB Daten geräteabhängig und es kommt zu großen Schwankungen der Qualität zwischen verschiedenen Geräten.



1 RGB Bild
2 Roter Kanal
3 Grüner Kanal
4 Blauer Kanal
5 Röhrenmonitor zeigt die Zahl 12
6 Nahaufnahme der 12

Farbdefinitionen in CSS

Seit dem 7. Juni 2011 gilt die das CSS Color Module Level 3[1] (CSS3) als Empfehlung für Farben von dem W3C. Mittlerweile werden die im CSS3 vorgestellten Möglichkeiten für Farbdefinitionen von den aktuellen Browsern unterstützt. Am gebräuchlichsten ist für die Bestimmung Hex RGB und rgba bei Transparenz.

Farbdefinition	Beispiel	Vorteil	Nachteil
Hex RGB *Hexadezimaler Notation*	**#450043** **#fof**	• *Wird seit 1994 benutzt* • *Sehr praktisch beim Coding*	• *Keine Transparenz* • *RGB nach Hex umrechnen*
RGB mit Alpha *rgba*	**rgba(**255,0,0,**1)** **rgba(**100%,0%,0%,**0.5)**	• *Transparenz* • *RGB Wert*	• *Unpraktischer beim Coding*
RGB *rgb*	**rgb(**255,255,0**)** **rgb(**100%,100%,0%**)**	• *RGB Wert*	• *Keine Transparenz* • *Unpraktisch beim Coding*
Color Key Word *Farbbezeichnung*	**red, black, lime** *etc*	• *Wird seit 1994 benutzt* • *Sehr praktisch beim Coding*	• *Nicht sehr viele Farbnuancen* • *burlywood ist nicht intuitiv*
HSL *Hue, Saturation, Lightness*	**hsl***(118,99%,50%)* **hsl***(118,98%,50%)*	• *Farbvariationen sind intuitiv anpassbar*	• *Unpraktischer beim Coding* • *Keine Transparenzen*
HSLA *HSL mit Alpha*	**hsla***(121,95%,50%,***1***);* **hsla***(121,95%,50%,***0.4***);*	• *Transparenz* • *Intuitiver als RGB*	• *Unpraktischer beim Coding*

Hex RGB

Das Format der RGB Werte in hexadezimaler Notation ist # (Raute), welches direkt von entweder sechs (#rrggbb) oder drei (#rgb) hexdezimalen Werten[2] gebildet wird. Hex RGB ist neben den Color Key Words die älteste Möglichkeit für Farbdefinitionen in CSS. Sie ist sehr praktisch, da aufgrund der kompakten Schreiweise im Editor per Doppelklick der komplette Wert erfasst und kopiert, ersetzt oder ausgeschnitten werden kann. Der wesentliche Nachteil ist die Berechnung dieser Farbwerte. Die Berechnung sollte aus praktischen Gründen mittels Bildbearbeitungssoftware oder anderen Tools erfolgen.

```
#FF0000;
*/ Hex für Rot 255 Grün 0 Blau 0 /*
```

```
background: #5b4d3d;
color: rgba(255,255,255,.6);
box-shadow: 0 1px 2px rgba(0,0,0,.25);
```

Klassiker der Farbdefinition im Webdesign: Hex RGB und rgba.

RGB

Ein RGB Wert kann direkt in das CSS eingebunden werden. Die Farbe wird über RGB gefolgt von einer geschweiften Klammer mit den Werten für RGB Werten, entweder von 0 bis 255 oder von 0 bis 100%, durch Komma getrennt definiert.

```
rgba(255,0,0,);      */ Rot 255, Grün 0, Blau 0 /*
rgba(100%,0,0);      */ Rot 100% (entsprich 255), Grün 0, Blau 0 /*
```

rgba – RGB mit Alpha

Soll z.B. ein * *, eine Schrift oder ein sonstiges Objekt nicht vollständig sichtbar sein, dann kann ein RGB Wert mit einer Zusatzbestimmung für den Alpha angegeben werden. Die Farbe wird über rgba gefolgt von einer Klammer gefolgt von den RGB Werten entweder von 0 bis 255 oder von 0 bis 100% definiert. Der Alpha (Transparenz) erfolgt über den vierten Wert und kann von 0.01 bis 1 in beliebiger Abstufung angelegt werden.

```
rgba(255,0,0,0.4);   */ Rot 255, Grün 0, Blau 0, Alpha 40% Sichtbar /*
rgba(100%,0,0,0.3); */ Rot 100%, Grün 0, Blau 0, Alpha 30% Sichtbar /*
```

1 *Siehe* **http://www.w3.org/TR/css3-color/**
2 *Siehe* **Hexadezimale Zahlen berechnen**

Color Keyword

Ursprünglich gab es die Keywords aqua, black, blue, gray, green, lime, maroon, navy, olive, purple, red, silver, teal, white, and yellow. Groß- und Kleinschreibung ist dabei egal. Die Liste wurde mittlerweile erweitert. Zum Beispiel burlywood, fuchsia oder darkslategrey. Eine vollständige Tabelle der 140 Farben mit Hex RGB und Dezimal RGB ist unten abgebildet.

```
red;
fuchsia;
```

Color Key Word	Hex RGB	Dezimal RGB
aliceblue	#f0f8ff	240,248,255
antiquewhite	#faebd7	250,235,215
aqua	#00ffff	0,255,255
aquamarine	#7fffd4	127,255,212
azure	#f0ffff	240,255,255
beige	#f5f5dc	245,245,220
bisque	#ffe4c4	255,228,196
black	#000000	0,0,0
blanchedalmond	#ffebcd	255,235,205
blue	#0000ff	0,0,255
blueviolet	#8a2be2	138,43,226
brown	#a52a2a	165,42,42
burlywood	#deb887	222,184,135
cadetblue	#5f9ea0	95,158,160
chartreuse	#7fff00	127,255,0
chocolate	#d2691e	210,105,30
coral	#ff7f50	255,127,80
cornflowerblue	#6495ed	100,149,237
cornsilk	#fff8dc	255,248,220
crimson	#dc143c	220,20,60
cyan	#00ffff	0,255,255
darkblue	#00008b	0,0,139
darkcyan	#008b8b	0,139,139
darkgoldenrod	#b8860b	184,134,11
darkgray	#a9a9a9	169,169,169
darkgreen	#006400	0,100,0
darkgrey	#a9a9a9	169,169,169
darkkhaki	#bdb76b	189,183,107
darkmagenta	#8b008b	139,0,139
darkolivegreen	#556b2f	85,107,47
darkorange	#ff8c00	255,140,0
darkorchid	#9932cc	153,50,204
darkred	#8b0000	139,0,0
darksalmon	#e9967a	233,150,122
darkseagreen	#8fbc8f	143,188,143
darkslateblue	#483d8b	72,61,139
darkslategrey	#2f4f4f	47,79,79
darkturquoise	#00ced1	0,206,209
darkviolet	#9400d3	148,0,211
deeppink	#ff1493	255,20,147
deepskyblue	#00bfff	0,191,255
dimgrey	#696969	105,105,105
dodgerblue	#1e90ff	30,144,255
firebrick	#b22222	178,34,34
floralwhite	#fffaf0	255,250,240
forestgreen	#228b22	34,139,34
fuchsia	#ff00ff	255,0,255
gainsboro	#dcdcdc	220,220,220
ghostwhite	#f8f8ff	248,248,255
gold	#ffd700	255,215,0
goldenrod	#daa520	218,165,32
gray	#808080	128,128,128
green	#008000	0,128,0
greenyellow	#adff2f	173,255,47
grey	#808080	128,128,128
honeydew	#f0fff0	240,255,240
hotpink	#ff69b4	255,105,180
indianred	#cd5c5c	205,92,92
indigo	#4b0082	75,0,130
ivory	#fffff0	255,255,240
khaki	#f0e68c	240,230,140
lavender	#e6e6fa	230,230,250
lavenderblush	#fff0f5	255,240,245
lawngreen	#7cfc00	124,252,0
lemonchiffon	#fffacd	255,250,205
lightblue	#add8e6	173,216,230
lightcoral	#f08080	240,128,128
lightcyan	#e0ffff	224,255,255
lightgoldenrodyellow	#fafad2	250,250,210
lightgray	#d3d3d3	211,211,211
lightgreen	#90ee90	144,238,144
lightgrey	#d3d3d3	211,211,211
lightpink	#ffb6c1	255,182,193
lightsalmon	#ffa07a	255,160,122
lightseagreen	#20b2aa	32,178,170
lightskyblue	#87cefa	135,206,250
lightslategrey	#778899	119,136,153
lightsteelblue	#b0c4de	176,196,222
lightyellow	#ffffe0	255,255,224
lime	#00ff00	0,255,0
limegreen	#32cd32	50,205,50
linen	#faf0e6	250,240,230
magenta	#ff00ff	255,0,255
maroon	#800000	128,0,0
mediumaquamarine	#66cdaa	102,205,170
mediumblue	#0000cd	0,0,205
mediumorchid	#ba55d3	186,85,211
mediumpurple	#9370db	147,112,219
mediumseagreen	#3cb371	60,179,113
mediumslateblue	#7b68ee	123,104,238
mediumspringgreen	#00fa9a	0,250,154
mediumturquoise	#48d1cc	72,209,204
mediumvioletred	#c71585	199,21,133
midnightblue	#191970	25,25,112
mintcream	#f5fffa	245,255,250
mistyrose	#ffe4e1	255,228,225
moccasin	#ffe4b5	255,228,181
navajowhite	#ffdead	255,222,173
navy	#000080	0,0,128
oldlace	#fdf5e6	253,245,230
olive	#808000	128,128,0
olivedrab	#6b8e23	107,142,35
orange	#ffa500	255,165,0
orangered	#ff4500	255,69,0
orchid	#da70d6	218,112,214
palegoldenrod	#eee8aa	238,232,170
palegreen	#98fb98	152,251,152
paleturquoise	#afeeee	175,238,238
palevioletred	#db7093	219,112,147
papayawhip	#ffefd5	255,239,213
peachpuff	#ffdab9	255,218,185
peru	#cd853f	205,133,63
pink	#ffc0cb	255,192,203
plum	#dda0dd	221,160,221
powderblue	#b0e0e6	176,224,230
purple	#800080	128,0,128
red	#ff0000	255,0,0
rosybrown	#bc8f8f	188,143,143
royalblue	#4169e1	65,105,225
saddlebrown	#8b4513	139,69,19
salmon	#fa8072	250,128,114
sandybrown	#f4a460	244,164,96
seagreen	#2e8b57	46,139,87
seashell	#fff5ee	255,245,238
sienna	#a0522d	160,82,45
silver	#c0c0c0	192,192,192
skyblue	#87ceeb	135,206,235
slateblue	#6a5acd	106,90,205
slategray	#708090	112,128,144
slategrey	#708090	112,128,144
snow	#fffafa	255,250,250
springgreen	#00ff7f	0,255,127
steelblue	#4682b4	70,130,180
tan	#d2b48c	210,180,140
teal	#008080	0,128,128
thistle	#d8bfd8	216,191,216
tomato	#ff6347	255,99,71
turquoise	#40e0d0	64,224,208
violet	#ee82ee	238,130,238
wheat	#f5deb3	245,222,179
white	#ffffff	255,255,255
whitesmoke	#f5f5f5	245,245,245
yellow	#ffff00	255,255,0
yellowgreen	#9acd32	154,205,50

HSL

Die W3C hat beobachtet, dass das RGB Modell nicht besonders intuitiv ist und erst recht nicht die Umrechnung nach Hex RGB. Für eine Vereinfachung ist das HSL System in CSS integriert worden. Im HSL können Hue (Farbton), Saturation (Sättigung) und Lightness (Helligkeit) getrennt eingestellt werden. Im HSL System wird der Farbton als Grad angegeben und Sättigung sowie Helligkeit in Prozent. Rot ist per Definition sowohl 0° als auch 360°. Grün hat 120°und Blau 240°. Es gibt im Photoshop Farbwähler ein vergleichbares System: HSB. HSB ist Hue, Saturation, Brightness. Doch Vorsicht, denn HSB und HSL sind nicht identisch.

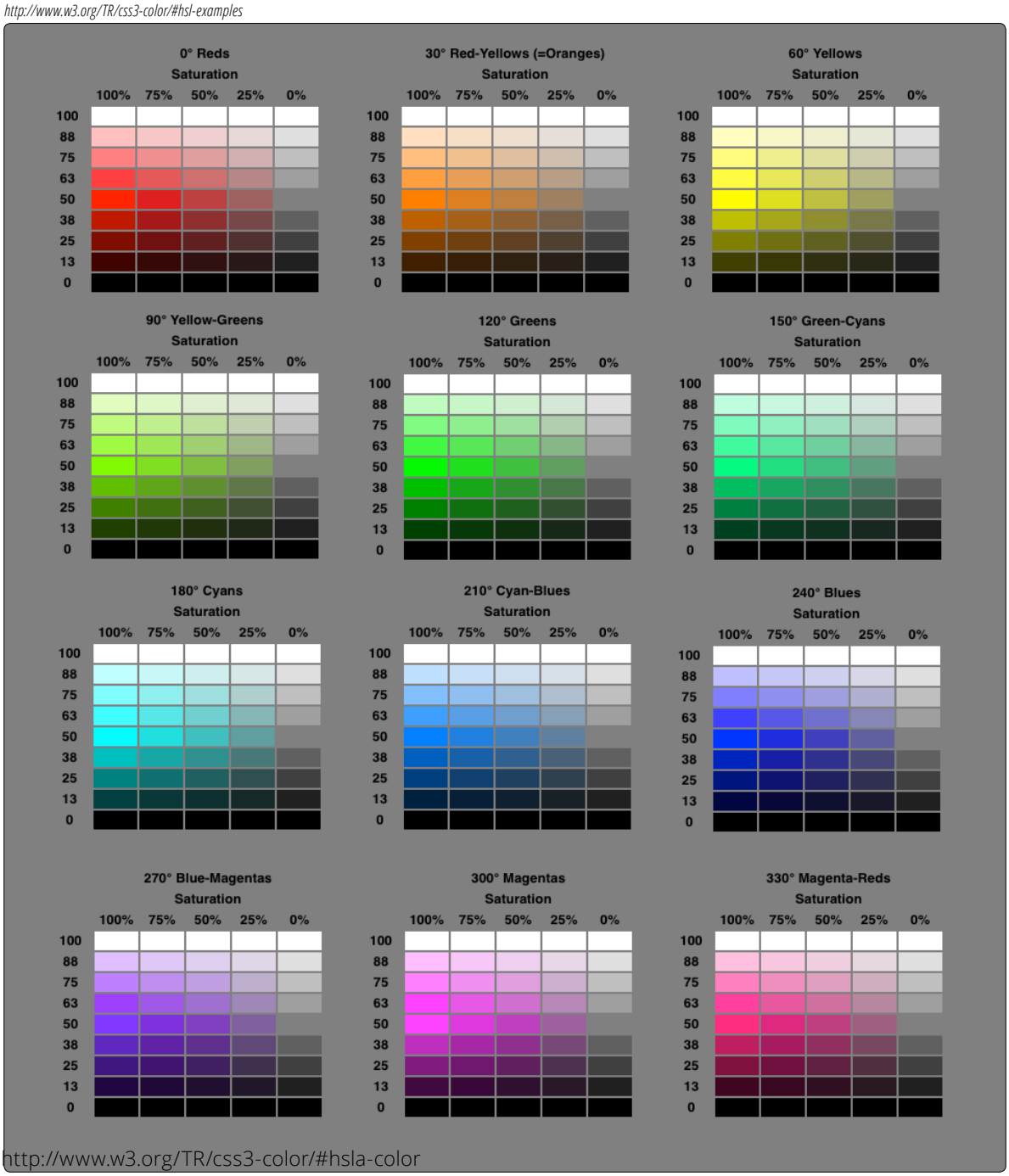
```
hsl(118,99%,50%)
```

HSLA

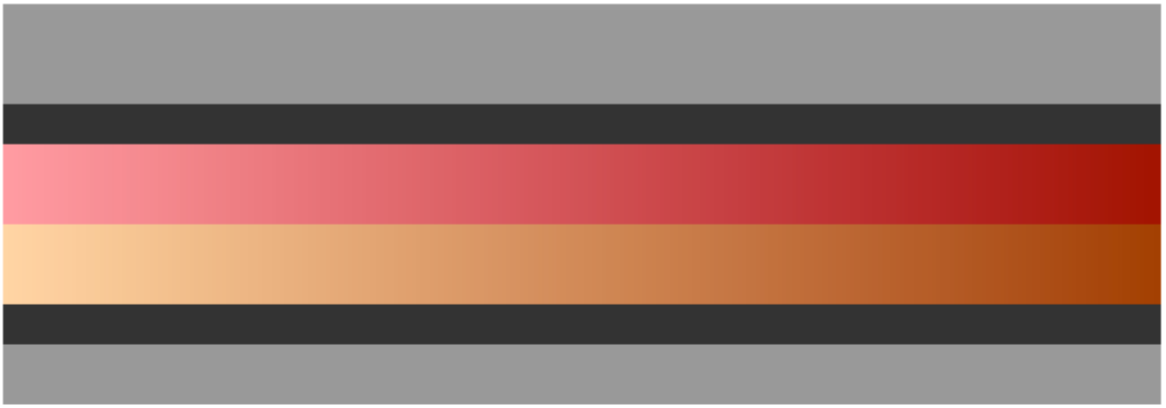
Die HSL Werte können ebenso um Transparenz erweitert werden. Wie im RGBA ist der letzte Wert der Alpha Wert und regelt die Deckkraft von 0.0 (unsichbar) bis 1.0 (sichtbar).

```
hsla(121,95%,50%,0.4)
```

http://www.w3.org/TR/css3-color/#hsl-examples



http://www.w3.org/TR/css3-color/#hsla-color



<!--HTML-->

```
<main>
   <header></header>
   <nav></nav>
   <section>
      <article>
         <div class='red'></div>
      </article>
      <article>
         <div id='orange'></div>
      </article>
   </section>
   <aside></aside>
   <footer></footer>
</main>
```

<!--CSS-->

```
main  {background:#eee;}
header {background:#999; height:50px;}
nav  {background:#333; height:20px;}
footer {background:#999; height:30px;}
aside {background:#333; height:20px;}

section {background:
linear-gradient(to right, #fff,#600);}

article div {height: 40px;
             opacity: 0.4;}

.red     {background: red;}
#orange  {background: #ff9900;}
```

code.arnoldbodeschule.de/design-mit-css/

a) Die Farbe von header und footer wird von #999 auf #777 geändert. Wird der Grauton heller oder dunkler?

b) Section hat einen Verlauf von #fff zu #900. Welche Farben sind das?

RGB, Hex RGB und HSL Farbwerte bestimmen

Aus wirtschaftlichen Gründen ist es empfehlenswert alle möglichen Hilfen und Tools bei der Bestimmung von Farbwerten zu nutzen, statt jede Farbe auszurechnen. Zahlreiche Programme und Webseiten bieten hier Tools an, sicherlich ist der Farbwähler aus Adobe Photoshop eine der am häufigsten genutzen Möglichkeiten. Sehr nützlich ist zum Beipspiel die Webseite Colorzilla.

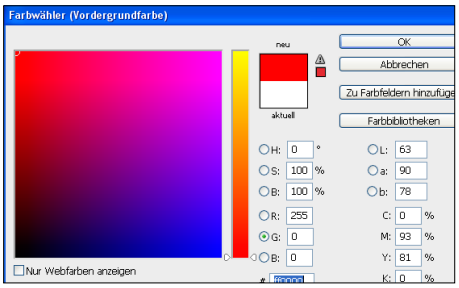
Echte HSL Werte können nicht aus dem Photoshop direkt ausgelesen werden. Das dort benutzte und auch ähnliche Farbmischsystem HSB berechnet die Farben etwas anders. Colorzilla gibt die für CSS benötigten HSL Werte hingegen korrekt aus.

Die Pipette aus Photoshop hilft bei der Bestimmung von Farbwerten aus Bildern.

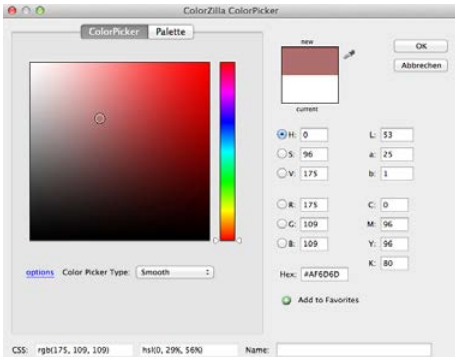
CSS Farbverläufe erstellen

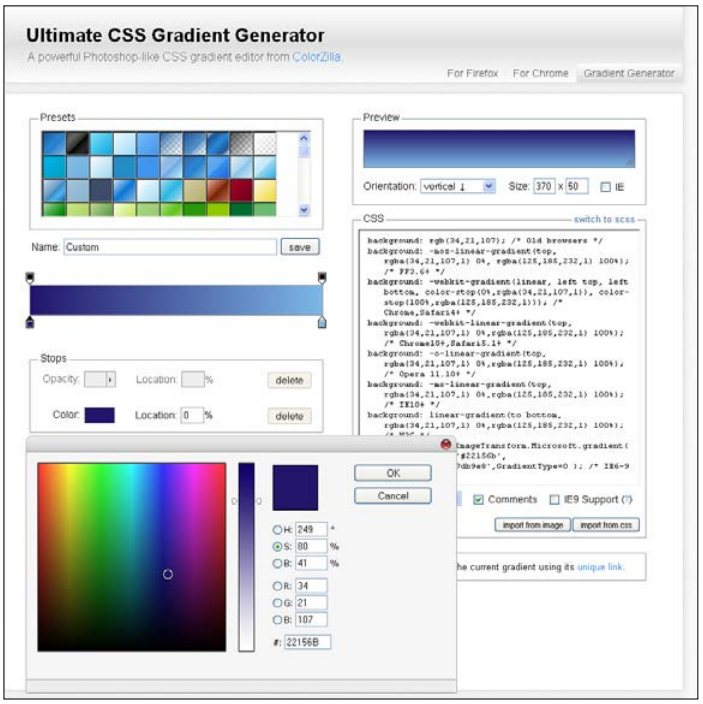
Bei den einfachen Hex RGB Werten sind die Tools und Hilfen bereits eine große Erleichterung. Sollen jedoch Farbverläufe erstellt werden, dann kommt man um die Benutzung dieser Tools nicht mehr drumherum.

Die kostenlose Webseite Colorzilla ist eine sehr gute Lösung, um komplexe Fabrverläufe, auch mit Transparenz zu erstellen. Es wird der vollständige CSS Code erstellt.



***Oben**: Der Farbwähler aus Photoshop ist der Klassiker für die Auswahl von Farben. Allerdings ist der Photoshop HSB nicht der CSS HSL.*
***Unten**: Kostenlos und sehr praktisch: Der Colorzilla Colorpicker ist ein Addon für Firefox und Chrome. Im Gegensatz zu Photoshop rechnet Colorzilla in CSS-HSL um.*





http://www.colorzilla.com/gradient-editor/

https://color.adobe.com/de/create/color-wheel/

Farbakkorde komponieren

Für eine Webseite benötigen wir häufig mehrere Farben die zueinander harmonisch sind, aber auch kontrastreich zusammen wirken. Es ist völlig korrekt, diese Farbakkorde nach Gefühl zu komponieren. Letztlich muss das künstlerische Gespür vor die blanke Berechnung gehen.

Geschichte der Farbkreise

Newton, Goethe, Bezold, Hering, Scriabin … im Laufe der Zeit greifen diese und weitere Forscher und Künstler das Prinzip des Farbkreises auf. Eine Arbeit prägt die Arbeit mit Farben bis heute: Die Forschungen des Professors Johannes Itten. Seine Systematik zum Gebrauch von Farben mittels Farbkreis ist in den unteren Abbildungen illustriert. Auch Itten steht in der Kritik durch Andere, z.B. Küppers.

Für eine systematische Einarbeitung in die Farbkomposition sind Farbkreise nach wie vor empfehlenswert und mit einer modernen Abwandlung, Adobe Kooler, werden wir arbeiten.

***Oben Links**: Newton sortiert die Farben seines Farbkreises nach seinen religiösen Vorstellungen. Der violette, blaue und indigofarbene Bereich nimmt z.B. relativ viel Raum ein. Eine physikalische Systematik liegt nicht zu Grunde.*

***Oben Rechts**: Goethe geht in bewusste Konfrontation zu der Arbeit von Newton. Er beschwert sich über die Versuche, bei denen Newton das Licht duch enge Tunnel quält und in seine Bestandteile aufspaltet. Sein Farbkreis illustriert in Goethes Farbenlehre das Kapitel: „Allegorischer, symbolischer, mystischer Gebrauch der Farbe". Goethe ordnet im inneren Ring den Farben Eigensschaften zu. Rot: „schön". Orange: Edel. Gelb: Gut. Grün: nützlich. Blau: gemein. Violett: unnöthig. Im äußeren Kreis ordnet er den Mischfarben Substantive zu: Rot-orange: Vernunft. Gelb-Grün: Verstand. Grün-Blau: Sinnlichkeit. Violet-Rot: Phantasie. Letzlich scheitert Goethe mit seinem Versuch, die Theorie Newtons zu zerstören und kann sie nie wiederlegen.*

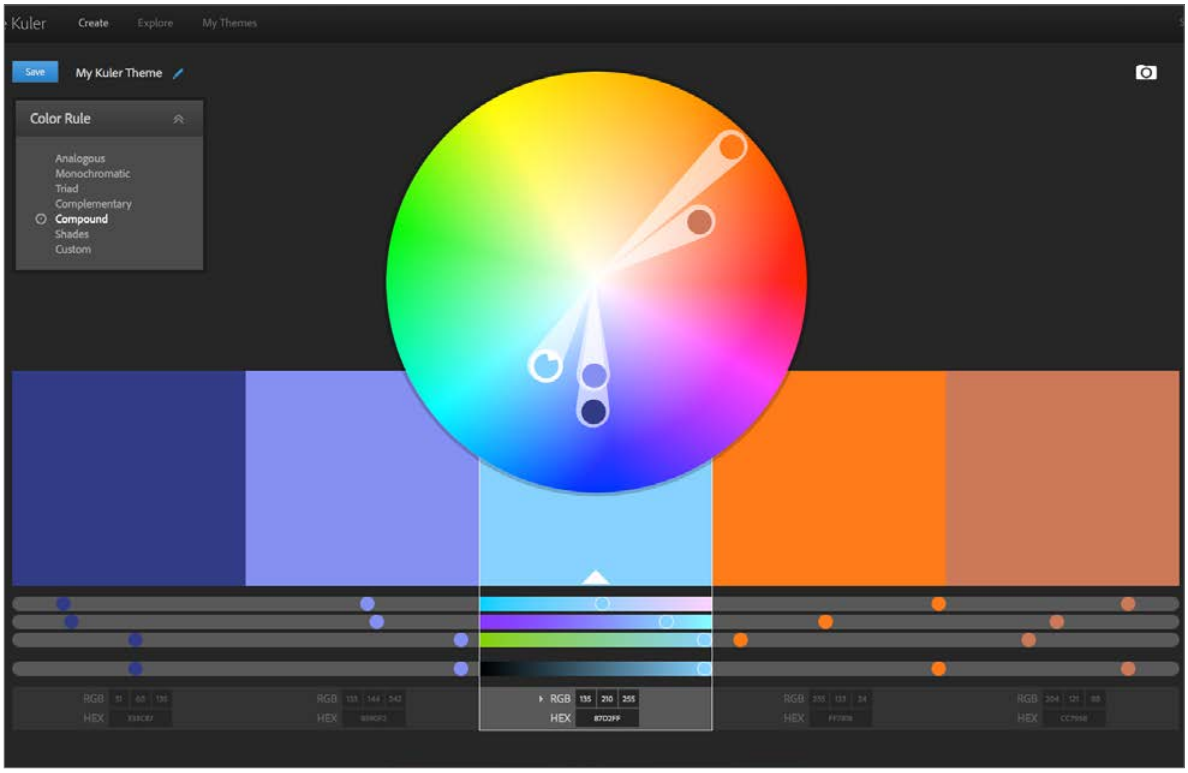
Obwohl Goethe Schopenhauer inspiriert, wird er mit diesem auch nie Grün. »Trüge gern noch länger des Lehrers Bürden, wenn Schüler nur nicht gleich Lehrer würden…« Ist ein Zitat von Goethe über Schopenhauer in dessen Werk „Über das Sehn und die Farbe".

Gelb steht in Ittens Farbkreis oben und die weiteren Grundfarben Rot und Blau sind jeweils um 120° versetzt. Bei 60°, 180° und 300° liegen die Sekundärfarben. Diese Anordnung ist entspricht dem Spektrum und schließt sich bei Rot zu Violett

Komplementäre Farben liegen sich im Farbkreis von Itten gegenüber.

Mit einem spitzen Dreieck können zwei harmonisch zueinanderpassende Farben und eine kontrastreiche Farbe ermittelt werden.

Mit einem Rechteck können zwei harmonische Paare die zueinander kontrastreich sind ermittelt werden. Denkbar sind auch andere geometrische Formen.

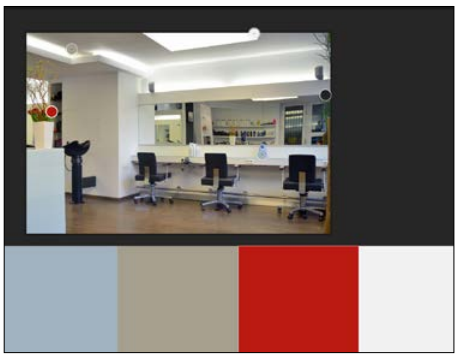


Farbakkorde mit Adobe Kuler

Eine interessante App im Internet ist Adobe Kuler[1]. Die Funktion ist auch in der Creative Suite integriert. Auf Basis des Farbkreises und Fotos können Farbakkorde ausgewählt und abgestimmt werden.

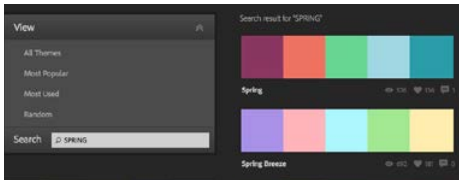
https://color.adobe.com/de/create/color-wheel/

Variationen der Farbakkorde können einfach erstellt werden und basieren auf unterschiedlichen Kontrast-Prinzipien, wie z.B. Komplementär- oder Hell-Dunkel Kontrast.



Kuler sammelt die erstellten Farbakkorde in einer Datenbank und diese können verwendet und abgeändert werden. Auch kann man sich vorhandene Akkorde über Suchwörter, z.B. zum Thema Herbst, anzeigen lassen.

Sehr inspirierend ist der eingebaute Bilder-Upload. Auf Basis des Bildes können Farbprinzipien entwickelt werden. Populäre Farbakkorde (Themes) können in der App durchstöbert werden.



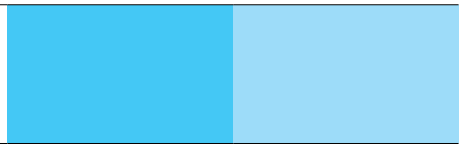
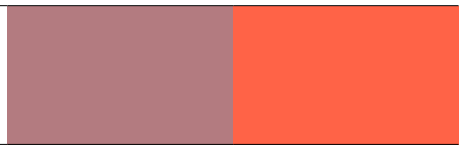
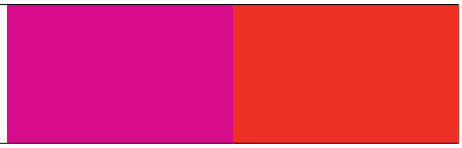
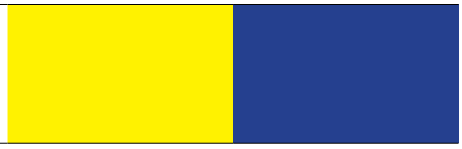
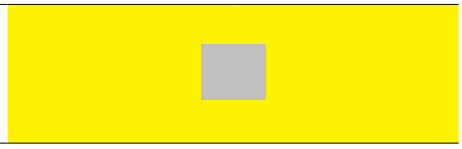
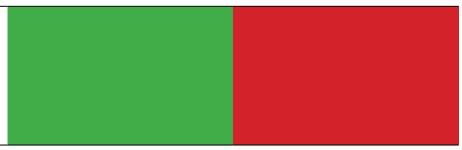
1 *Kuler ist ein mauretanisches Wort für Farbe*

Farben fein abstimmen

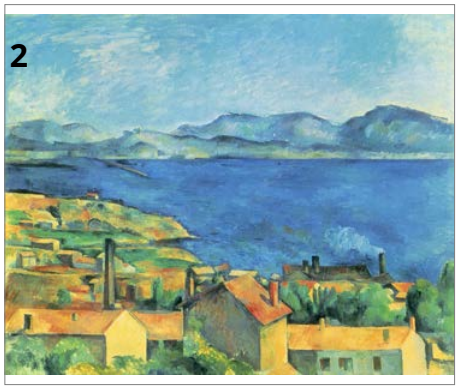
Farbkontraste nach Itten

Kontrast ist ein wesentliches Element in der Gestaltung. Ein starker Kontrast erzeugt Signalwirkung und lenkt den Blick, ein geringer Kontrast wiederum ist an anderer Stelle das stimmige Gestaltungselement.

Die Farbkontraste hat Johannes Itten untersucht. Diese zu kennen hilft bei der Gestaltung im Allgemeinen und bei der Konzeption von Webseiten. Itten formuliert sieben Farbkontraste und diese wirken, ob bewusst oder unbewusst gewählt, als Gestaltungselement.

Kontrast	Kurzbeschreibung	Beispiel
Farbe-an-sich-Kontrast	*Kontrast der Farben aufgrund des Farbtons. Die Fähigkeit die Farbtöne zu unterscheiden ist von der Kultur abhängig und prägt sich durch das Leben.*	
Hell-Dunkel-Kontrast	*Kontrast der Farben aufgrund der Helligkeit: Schwarz-Weiß und Blau-Gelb sind besonders intensive Hell-Dunkel Kontraste. Grün und Rot haben geringen Hell-Dunkel-Kontrast.*	
Qualitäts-Kontrast	*Kontrast der Farbe aufgrund der Sättigung: Eine sehr reine, stark gesättigte Farbe leuchtet strahlend und konrastreich aus ungesättigtem Umfeld hervor.*	
Kalt-Warm-Kontrast	*Kontrast der Farben aufgrund unseres damit verbundenen, subjektiven Wärme-Empfindens.*	
Komplementär-Kontrast	*Kontrast der Farben aufgrund ihrer komplemtären Wirkung. Der Komplementär-Kontrast erzeugt eine starke Signalwirkung.*	
Simultan-Kontrast	*Kontrast und Wirkung der Farbe aufgrund des Sehvorgang. Das Gehirn verändert die gesehene Farbe in Beziehung zu seiner Umgebungsfläche.*	
Quantitäts-Kontrast	*Kontrast der Farbe aufgrund ihrer subjetiv empfundenen, unterschiedlichen Gewichtung: Rot und Grün sind harmonisch, wenig Gelb hält viel Balance zu quantitativ deutlich mehr Violett.*	

a) Welcher Farbkontrast prägt jeweils diese Bilder?



1 Caspar David Friedrich: Nebel, 1807.

2 Paul Cézanne: Die Bucht von Marseille, von L'Estaque aus gesehen, um 1885

3 Franz Marc: Blauschwarzer Fuchs, 1911

4 Van-Gogh: Sternennacht über der Rhône, 1888

5 Gauguin: In den Wellen (Undine), 1889

6 Francisco de Goya: Porträt des Tiburcio Pérez y Cuervo, 1820

LS 03 Farben mit dem Kunden am Telefon abstimmen

Zusammen vor dem Bildschirm sitzen und die Farben abstimmen, ist gegenüber einem Telefonat deutlich einfacher. Die Situation am Telefon ist jedoch häufiger. Für solche Gesprächssituationen ist die systematische Verwendung der Farbkontraste eine große Hilfe, um den Korrekturaufwand zu verkleinern. Dies simulieren wir mit der folgenden Lernsituation.

Der Designer hat einen Vorschlag gemacht, der Kunde hat etwas andere Vorstellungen und über das Gespräch muss sich der Designer herantasten.

Ablauf

- Zwei Personen tun sich zusammen, ihr wechselt euch mit der Rolle des Kunden und des Designers ab
- Ihr stellt die Ausgangsfarbe ein
- Einigt euch im Vorfeld auf einen Schwierigkeitsgrad
- Ihr dreht die Monitore so, dass euer Gegenüber das Display nicht sehen kann
- Der Kunde legt zu erst seinen Wunsch fest
- Der Designer versucht nun diesen Wunsch zu ermitteln, indem er Fragen formuliert und den Entwurf ändert
- Abschluss, Auswertung und Eintrag im Logbuch

Schwierigkeitsgrad

- **Einfach:** Der Kunde ist gut gelaunt, hat Zeit, und hilft aktiv bei der Findung des Farbtons mit und bringt ggf. eigene Vorkenntnisse mit ein.
- **Mittel:** Der Kunde ist interessiert, hat aber nur ein enges Zeitfenster von vielleicht zwei bis fünf Fragen, versteht nicht jede Frage – spricht dies jedoch an.
- **Schwierig:** Der Kunde ist genervt, weil die Farbe immer noch nicht stimmt, reagiert verwirrt auf unklare Fragen, schweift ab und erzählt irgendwas oder ändert seine Meinung etc.

Auswertung

Nachdem das Gespräch geendet hat und der Designer seinen neuen Entwurf abgestimmt hat, vergleicht ihr am Monitor Kunden-Wunsch und Designer-Entwurf. Schreibt euch dann die RGB Werte auf. Wie im Folgenden Beispiel ermittelt ihr zum Abschluss die Abweichung von Kundenwunsch und Entwurf.

	Ausgang	Wunsch des Kundens	Entwurf des Designers	Abweichung
Rot	*128*	*135*	*128*	*7*
Grün	*59*	*64*	*89*	*27*
Blau	*28*	*28*	*34*	*6*
				40 in der Summe

Schriftliches Feedback im Logbuch

Team	Kunde	Designer
Summe der Abweichung	*Kann ich mit dem Entwurf leben?*	*Was hat mir in dem Gespräch besonders geholfen?*
Schwierigkeitsgrad 1 bis 3	*Hat mein Designer auf mich professionell gewirkt?*	*Was hat mich in dem Gespräch aus dem Konzept gebracht?*
	Was war richtig gut?	
	Was war gar nicht gut?	

Subjektivität beachten

Der Farbeindruck mit all seinen Farbkontrasten ensteht im Geist des jeweiligen Beobachters und wird dort im Kontext seiner bisherigen Erfahrungen erlebt. Individuelle Vorlieben und kulturelle Gemeinsamkeiten prägen die Wahrnehmung von Farben. Somit ist die Gesprächsführung mit Kunden und Zielgruppen bei der Abstimmung von Farben der wesentliche Faktor in der kommerziellen Farbabstimmung.

„(...) Anzahl und Geltung der zur Verfügung stehenden Farbwörter sind verschieden (wenn auch in den europäischen Sprachen ein weitgehender Ausgleich eingetreten sein mag)" Die Grammatik, Duden, S. 446

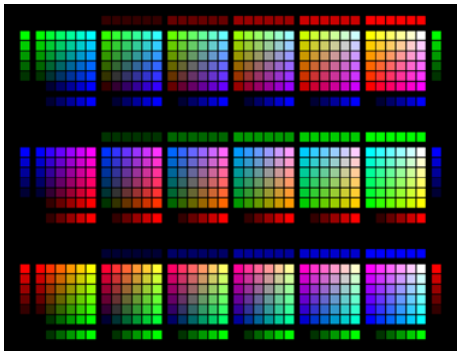
Ein Beispiel für kulturelle Einflüsse ist z.B. das Phänomen, das viele Kulturen nicht oder anders zwischen Grün und Blau unterscheiden, es gibt viele andere Unterschiede.

Kultur	Ausdruck	Beschreibung
Chinesisch	*qīng*	*Containerbegriff für grün, blau, (dunkel-)violett und schwarz*
Arabisch	*al-khadrā'* (الخضراء)	*In der klassichen arabischen Poesie wird mitunter der Himmel als Grün bezeichnet.*
Japanisch	*aoi*	*Beschreibt Grün und Blau gleichzeitig. Man kann auch „aoi" aussehen und es dient als Beschreibung der Ampelfarbe.*
Russisch	*голубой, синий*	*Das Wort Blau gibt es nicht, dafür eindeutige Wörter für Hell- und Dunkelblau*
Griechisch	*chloros*	*Beschreibt die Farbe von Honig und Gras in altgriechischen Texten*
Vietnamesisch	*xanh*	*Beschreibt sowohl Grün und Blau*
Guaraní	*hovy*	*Beschreibt sowohl Grün und Blau*

Hinweis auf eine Ampel in Okazaki. Der Begriff aoi ist flexibel genug, um auch mit reinem Blau als Signalfarbe zu funktionieren.

Einflussfaktor Ausgabegerät

Ich habe es selber leidlich und häufig erfahren: Bei der Gestaltung setze ich hochwertige High-End Geräte ein. Die Webseite wird gestaltet und feinste Nuancen der Farbe abgestimmt. Ein Test erfolgt auf Smartphones, Tablets und anderen Geräten und alle haben eine hohe Qualität. Mag sein, dass ein Farbdrift fest gestellt werden kann, letzlich sieht alles trotzdem gut aus. Dann kommt der Kulturschock: Auf schlechteren Ausgabegeräten werden diese Feinheiten noch nicht einmal im Ansatz dargestellt. Viele Besucher werden jedoch auf eher schlechten Geräten die Webseite besuchen.



Damals als Plastik noch aus Holz war konnten Computer eine Zeitlang nur 256 Farben darstellen. Als wirklich sicher galten diese 216 Farben.

CC3, R. A. Nonenmacher

Ein Lösungsansatz für diese Situation ist die Verwendung von eher deutlichen Kontrasten. Früher war das noch schlimmer: Die Zeiten der 216 websicheren Farben ist sicherlich vorbei und ein paar Jahre wird es bis zu einer flächendeckenden Verbreitung von hochauflösenden Schirmen noch dauern. Daher ist es sehr empfehlenswert die eigenen Farbakkorde auf möglichst vielen Ausgabegeräten zu testen, um ein Gespür dafür zu entwickeln.

Das Gelb ist zum Beispiel auf Monitoren eine eher problematische Farbe. Die meisten Monitore haben einen Blaustich und so kippt die Farbe leicht in einen grünen Farbton, falls der rechnerisch korrekte Werg #ffff00 für das maximale Gelb gewählt ist. Soll eine Fläche sicher gelb wirken, so wird deswegen häufig ein Farbton, der in Richtung Orange driftet, gewählt. Z.B. macht das die Deutsche Post auf ihrer Webseite.

Hex RGB berechnen

Hex RGB ist aufgrund seiner kompakten schreibweise ideal für Coding geeignet. Die Basis für Hex RGB ist das hexadezimale Zahlensystem. Es unterscheidet sich vom bekannten dezimalen Zahlensystem dadurch, dass mit 16 statt mit 10 Zahlen als Grundlage gerechnet wird. Die Zeichen für 10 bis 15 werden durch die Buchstaben A bis F ersetzt.

Dezimal	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Hexal	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F

Druck	RGB	Hex RGB	Grauwert
100 %	0, 0, 0	#000000	Schwarz
75 %	64, 64, 64	#404040	Dunkelgrau
50 %	128, 128, 128	#808080	Mittelgrau
25 %	192, 192, 192	#C0C0C0	Hellgrau
0 %	255, 255, 255	#FFFFFF	Weiß

Je nach Bedarf kann eine Dezimalzahl nach Hex umgerechntet werden oder von Hex nach Dezimal. Dafür gibt es, so ein Taschenrechner nicht greifbar ist, jeweils ein simples Schema.

Um Dezimal nach Hex umzurechnen wird die Zahl durch 16 geteilt. Der Quotient wird anschließend erneut durch 16 geteilt. Der Rest jeder Division wird bei 0-9 als Zahl und bei 10-15 als Buchstabe notiert. Aus dem berechneten Rest ergibt sich der wert in hexadezimaler Schreibweise.

Soll ein Hex RGB in einen dezimalen Wert umgerechnet werden, so wird der linke Wert mit einer 16 als Zehner-Potenz und der rechte Wert mit einer 16 als Nuller-Potenz multipliziert. Das Ergebnis wird dann addiert.

Dezimal nach Hex umrechnen

Die Dezimalzahl **168** wird in das Hexadezimalsystem umgewandelt.

168 : 16 = 10 Rest: 8 = Hex: 8
10 : 16 = 0 Rest: 10 = Hex: A

A8

Hex nach Dezimal umrechnen

Die Hexadezimalzahl **DE** wird in das Dezimalsystem umgewandelt.

E= $14*16^0$ = 14*1 = 14
D= $13*16^1$ = 13*16 = 208 +

222

Rechne A8 zurück in das Dezimalsystem und 222 zurück in das Hexadezimalsystem.

Fragen zur Umrechnung von RGB und Hex RGB

a) Welcher RGB Wert ist #d1c233?

b) Welcher Hex RGB Wert ist rgb {128,128,128}?

c) Welcher Hex RGB Wert ist rgb {120,0,255}

Musterlösung für die Aufgaben mit RGB und Hex RGB

a) Rechnen Sie #d1c233 nach RGB

Die Hexadezimalzahl D1 wird ins Dezimalsystem umgewandelt

1: 1 · 1 = 1
D: 13 · 16 = 208
209

Die Hexadezimalzahl C2 wird ins Dezimalsystem umgewandelt

2: 2 · 1 = 2
C: 12 · 16 = 192
194

Die Hexadezimalzahl 33 wird ins Dezimalsystem umgewandelt

3: 3 · 1 = 3
3: 3 · 16 = 48
51

```
Anwort: rgb (209, 194, 51) ist der dezimale Wert für #d1c233
```

b) Welcher Hex RGB Wert ist rgb {128,128,128}?

Die Dezimalzahl 128 wird ins Hexadezimalsystem umgewandelt

128 : 16 = 8 Rest: 0 » Ziffer: 0
8 : 16 = 0 Rest: 8 » Ziffer: 8
Resultat: **80**

```
Anwort: #808080 ist der hexadezimale Wert für rgb (128, 128, 128)
```

c) Welcher Hex RGB Wert ist rgb {120,0,255}?

Die Dezimalzahl 120 wird ins Hexadezimalsystem umgewandelt

120 : 16 = 7 Rest: 8 » Ziffer: 8
7 : 16 = 0 Rest: 7 » Ziffer: 7
Resultat: **78**

Die Dezimalzahl 0 wird ins Hexadezimalsystem umgewandelt

0 : 16 = 0 Rest: 0 --> Ziffer: 0
Resultat: **0**

Die Dezimalzahl 255 wird ins Hexadezimalsystem umgewandelt

255 : 16 = 15 Rest: 15 » Ziffer: F
15 : 16 = 0 Rest: 15 » Ziffer: F
Resultat: **FF**

```
Anwort: #7800FF ist der hexadezimale Wert für rgb (120, 0, 255)
```

Informationen verlinken

Der Begriff „hyperlink" ist 1964 oder 1965 durch Ted Nelson beim Start des Projektes Xanadu geprägt worden.

Projekt Xanadu war ein Konzept für eine Art globales Wiki zu einer Zeit in der es das Internet in der heutigen Form nicht gab.

Im Rahmen des Projekt Xanadu hat Douglas Engelbart mit seinem Tean den ersten Hyperlink implementiert.

Seit Beginn des Internets ist die Möglichkeit Informationen zu verlinken (bzw. zu referenzieren) ein wesentliches Element von HTML (Hypertext Markup Language). Der Link wurde ursprünglich auch Hyperlink genannt.

Das *Tag* <*a*> und das *Tag* <*link*> werden für diese Verlinkung genutzt. Auch andere Tags, z.B. <*img*>, <*script*> oder <*video*>, nutzen die Möglichkeit der Verlinkung mittels des Zusatzes *href* (Hypertext Referenz). Referenzen können über zwei verschiedene Konzepte verknüpft werden: Relative Pfade und absolute Pfade. Beide Konzepte haben Vor- und Nachteile, ergänzen sich jedoch ideal.

Element mit Attributen	Aktion	Art des Verknüpfungs-Pfades
`<link type="text/css" href="mobile.css">`	Lädt automatisch die CSS-Datei mobile.css	**Relativ**
`<a href="HTML/Seite2.html">weiter</a>`	Öffnet im Ordner „HTML" die Datei Seite 2.html, wenn der Besucher auf den Text weiter klickt.	**Relativ**
`<a href="http://www.absks.de">Arnold</a>`	Öffnet die Domain http://www.absks.de, wenn der Besucher auf den Text Arnold klickt.	**Absolut**
`<script src="http://code.jquery.com/jquery-latest.min.js" charset="utf-8"></script>`	Lädt automatisch die aktuellste Version der Javascript-LibraryjQuerry.	**Absolut**
`<img src= "kitten.jpg"/>`	Zeigt die Datei kitten.jpg als Bild an. Das Bild wird aus dem gleichen Ort der HTML Datei geladen.	**Relativ**
`<img src= "http://www.cats.de/kitten.jpg"/>`	Zeigt die Datei kitten.jpg als Bild an. Das Bild wird von der Adresse http://www.cats.de geladen.	**Absolut**
`<a href="#arnold">Arnold</a>`	Klickt man auf den Link Arnold springt der Browser zu dem Element mit der ID Arnold.	**relativ**
`<a href="arnold.html#arnold">Arnold</a>`	Klickt man auf den Link Arnold wird Arnold.html geladen und dort springt der Browser zu dem Element mit der ID Arnold.	**relativ**
`<a href="https://de.wikipedia.org/wiki/Arnold_Bode#documenta_1">Docu1</a>`	Klickt man auf den Link Doku wird die Seite bei Wikipedia geladen und dort springt der Browser zu dem Element mit der ID documenta1..	**Absolut**
`<img src="/Volumes/Fred/Hallo.jpg"/>`	Ein Mac wird nach einem Laufwerk Fred suchen und von dort das Bild Hallo.jpg laden. Unter Windows wird es nicht gefunden.	**Absolut**

<link href> für automatisches Laden im <head>

Das Element *<link href>* führt eine automatische Verlinkung aus. Der Browser öffnet selbstständig, die über *<link href="...>* definierte Datei. In erster Linie brauchen wir den Tag *<link>* für die Verknüpfung externer CSS Dokumente. Die *<link>* Tags werden im *<head>* der Webseite eingefügt.

Im Beispiel öffnet der Browser selbstständig die über das href-Attribut verlinkte Datei mobile.css. Noch bevor die Datei geladen wird, ist dem Browser bekannt, um welche Art es sich handelt. Der MIME-Type[1] ist in diesem Fall ein Textdokument, welches die Spezifikationen des CSS[2] erfüllt.

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

<a href="...> Links zum Klicken <body>

Durch das Anker-Element <a> ist eine aktive Verlinkung möglich. In dem Fall klickt der Benutzer selber auf den Link. Der Tag <a> wird erst durch die Angabe href="„. zum Link. Die Referenz kann entweder ...

- auf eine bestimmte Stelle (Anker-ID) auf der aktuellen Seite
- oder zu einer externen Seite auf
- oder auch außerhalb des Servers verweisen.

```
<a href="#ID">Verweist auf die Stelle #ID im Dokument</a>

<a href="HTML/Seite2.html">Öffnet im Ordner das HTML Dokument Seite 2</a>

<a href="http://www.herrseeliger.de">Öffnet die Domain herrseeliger.de</a>

<a href="HTML/Seite2.html#HuhuLol">Öffnet im Ordner
die Datei Seite2.html und geht dort zu der ID #HuhuLol</a>
```

Beispiel für ein Inhaltsverzeichnis auf Basis von <a href="„ und ID. Gezeigt wird die Wikipedia Seite zu Arnold-Bode.

Mit <a> können alle sichtbaren Elemente als Link genutzt werden. Z. B. Bilder. Klickt man auf das Bild, öffnet das Linkziel. Im gezeigten Beispiel wird das Linkziel (Seite auf Wikipedia) in einem neuen Browser geöffnet (target="_ blank„). Der Link hat als Title „Link zu Wikipedia".

```
<a href='https://de.wikipedia.org/wiki/Arnold_Bode'
   title='Link zu Wikipedia' target='_blank'>
   <img src='bilder/ArnoldStamp.png'></a>
```

1 *Siehe auch Punkt „Definition des Datei Typs mit MIME"*
2 *Siehe auch Kapitel Gestalten mit CSS*

<a href="… > mit Pseudoklassen gestalten

Sobald das <a> Element das Attribut href bekommen hat ist eine die Gestaltung des <a> mit Hilfe der Pseudoklassen :link, :visited, :hover und :active möglich. Diese müssen aufgrund der Spezifizität der Kaskade in genau dieser Reihenfolge im CSS bestimmt sein. Das Linkziel kann außerdem mit :target gestaltet werden.

```
<!DOCTYPE html>
<html>
    <head>
        <title>Hallo Welt</title>
        <style>
            a         {font-family: arial;}
            a:link    {}
            a:visited {}
            a:hover   {}
            a:active  {}

            #ID:target {background-color: red}

        </style>
    </head>
    <body>
        <a href="#ID">Link</a>

        <h1 id="ID">Hallo Welt</h1>
    </body>
</html>
```

:link

Unbesuchte Links, kann nur auf <a> angewendet werden. Wird erst aktiv, wenn bei <a&g das Attribut HREF gesetzt ist. Es gilt folgende Reihenfolge für die Gestaltung des Anker-Elements: <a> {...;} :LINK {...;} :VISITED {...;} :HOVER {...;} :ACTIVE {...;}.

:visited

Wählt besuchte Links aus, kann nur auf <a> angewendet werden. :VISITED wird erst aktiv, wenn auf <a> das Attribut HREF angewendet ist und der Link geklickt wurde. Es gilt folgende Reihenfolge für die Gestaltung des Anker-Elements: <a> {...;} :LINK {...;} :VISITED {...;} :HOVER {...;} :ACTIVE {...;}.

:active

Aktiver Link, kann nur auf <a> angewendet werden. Wird erst aktiv, wenn bei <a&g das Attribut HREF gesetzt ist UND der Link geklickt wird, also währen die Maustaste gedrückt ist. Es gilt folgende Reihenfolge für die Gestaltung des Anker-Elements: <a> {...;} :LINK {...;} :VISITED {...;} :HOVER {...;} :ACTIVE {...;}

Definiert man keine Pseudoklassen, dann gilt der Standard. Per CSS Standard hat die Pseudo-klasse :link die Farbe Blau und ist unterstrichen (text-decoration: underline). :visited bekommt die Farbe Purpur. :hover hat keinen Status, es ändert sich nur der Mauszeiger (cursor: pointer; cursor: hand;). Während des Klicks bekommt der Link die Farbe Rot. Das ist in der Pseudo-class :active bestimmt. Im gezeigten Beispiel bekommt das Link-Ziel einen roten Hintergrund.

:hover

Veränderung bei Kontakt mit dem Mauszeiger. Es gilt folgende Reihenfolge für die Gestaltung des Anker-Elements: <a> {...;} :LINK {...;} :VISITED {...;} :HOVER {...;} :ACTIVE {...;}. :HOVER kann im Gegensatz zu :LINK :VISITED und :ACTIVE auf jedes sichtbare Element angewendet werden. :HOVER spielt eine wichtige Rolle in der Benutzerfreundlichkeit. Mit :HOVER wird dem Nutzer eine Möglichkeit für Interaktion signalisiert.

:target

Wählt alle diese Elemente, wenn Sie mittels eines Links auf ihre ID verlinkt und angeklickt sind. Klicke ich auf einen Link der auf eine ID verweist, dann erscheint diese ID als Anhang mit # in der Inter-netadresse oben im Browserfenster. Dieses so gewählte Element kann mit :TARGET gestaltet werden.

Transitition

Die Übergänge von einem Zustand in eine andere Pseudoklass kann mit der Eigenschaft Transition animiert werden.

Eigenschaft	Werte	Bedeutung	Übersetzung	Beispiel
transition:	`all 1s width 2s ease, color 0.5s ease-on-out;`	Kurzschreibweise für delay duration property und timing function	Übergang	`a {transition: 0.5s color 3s; color: #000} a:hover {color: #fff}`

(Siehe Punk Transition unter „Gestalten mit CSS“ im Skript bzw. auf der CODE Plattform für weitere Informationen)

```
<style>

nav a {
  margin-right: 2rem;
  color: #f11;
  font-weight: bold;
  text-decoration: none;
  transition: color 0.5s;}

nav a:link {
  color: #f11;}

nav a:visited {
  color: #f11;}

nav a:hover {
  color: #fff;}

nav a:active {
  color: #900;}

  </style>
</head>
<body>

  <nav>
      <a href='#'>Link</a>
      <a href='#'>Link</a>
      <a href='#'>Link</a>
  </nav>
```

Beispiel für eine Animiation bei einem Link mit Transition. Die Farbe wird innerhalb von 0.5 Sekunden animiert.

Pseudo-Elemente

Mit den Pseudo-Elementen ::after und ::before können Inhalte vor oder nach dem Link platziert werden. Z.B ist es möglich im Ausdruck die URL anzugeben, während am Bildschirm die URL nicht angezeigt wird.

Download Bild
http://codedata.absks.de/149815.svg

```
@media PRINT { /*! Gilt für PDF und Ausdruck */
html          {font-size: 12pt;}
body          {background: #fff;}
img.big       {width: 8cm; margin-bottom: 1cm}
.float-right  {float: none; display: block;}
h4            {color: #555;}
a, a::after   {color: #000; text-decoration: none;}
a::after      {content: attr(href);
               display: block; font-size: 1rem; font-weight: bold}}
```

```
<a class='float-right'
   href='http://codedata.absks.de/149815.svg'>Download Bild</a>
```

Relative Pfade

Relativ zu der aktuell geöffneten Datei wird der Link gesetzt. Dabei gibt es die Möglichkeit in der Ordnerstruktur die Ebenen zu wechseln.

Relative Pfad-Angabe	Position des Links
`farben.htm`	Gleiche Ebene wie geöffnete Datei.
`./farben.htm`	Gleiche Ebene wie geöffnete Datei. Der Zusatz ./ wird selten genutzt.
`bilder/lolkitten.gif`	Eine Ebene weiter, im Ordner bilder wird die Datei lolkitten.gif geöffnet.
`./bilder/LOLdog.gif`	Eine Ebene weiter, Im Ordner bilder wird die Datei lolkitten. gif geöffnet. Der Zusatz ./ wird selten genutzt.
`../bilder/lolcat.jpg`	Datei lolcat.jpg liegt relativ gesehen im übergeordneten Ordner und dort im Ordner bilder. Man geht eine Ebene zurück und dort in den Ordner bilder.
`../../../weird.html`	Datei weird.html findest Du, indem Du drei Ebenen zurück gehst.

CSS Praxisbeispiel mit relativen Pfaden

Die Webseite Friseur Geoffrey benutzt mehrere CSS Stylesheets um aktuelle Geräte und Auflösungen optimal zu nutzen. Diese Stylesheets liegen im Ordner css und sind im <head> über <link> eingebunden. Angenommen der Bildschirm hat eine Auflösung von mindestens 1900 Pixel Breite 1202 Pixel Höhe, dann wird die Datei 90.css aus dem Ordner css benutzt und ansonsten eine der anderen CSS.

Soll in dieser CSS nun z.B. ein Hintergrundbild für ein DIV bestimmt werden, dann könnte der relative Pfad zu diesem Bild so wie im folgenden Bild aussehen. Relativ zu der Datei 90.css im aktuellen Ordner css wird eine Ebene zurück und dort im Ordner img-slide auf die Datei geoffreyweb-logo2.png verwiesen.

```
<link rel="stylesheet" media="screen and (min-height: 1202px) and (max-height: 1399px) and (min-width: 1900px)" href="css/90.css" />
```

```
    background-repeat: no-repeat;
background-image:url(../img-slide/geoffreyweb-logo2.png)
```

Absolute Pfad

Gerade wenn es um die Verlinkung auf externe Quellen geht, reichen die Möglichkeiten der relativen Pfade jedoch nicht aus.

Externe Quellen verlinken

Korrekterweise werden diese absoluten Pfade auch als vollständige Pfade bezeichnet. Genau bei dieser Funktion ergänzen die absoluten (bzw. vollständigen) Pfade die relativen.

`http://code.arnoldbodeschule.de`

Vorsicht bei absoluten Pfaden

Während die absolute Verlinkung bei Servern aufgrund des HTTP sehr gut, weil sie plattformunabhängig ist, funken uns die Betriebssysteme jedoch dazwischen.

Öffnet ein Mac einen USB Stick mit Namen Fred, dann sieht der absolute Pfad unter Mac OS so aus:

`/Volumes/Fred/Hallo.jpg`

Unter Windows jedoch ... und hier kann sich der Laufwerk-Buchstabe ändern.

`e://Hallo.jpg`

Zusammefassung absolute Pfade und relative Pfade

Art des Links	Beschreibung
Relativ	Grundsätzlich werden alle Verlinkungen der eigenen Homepage mit relativen Pfaden umgesetzt.
Absolut / vollständig	Links auf externe Quellen sind jedoch immer ein absoluter Link.

Tipps für barrierefreie Links mit <a>

- Pack die <a> Elemente für die Navigation in das <nav> Element. Screenreader erkennen diesen Bereich als Navigation.
- Verzichte auf das Wort „Link“ im Text. Screenreader teilen ihren Benutzern mit, wenn diese auf einen Link gehen.
- Keine GROSSBUCHSTABEN. Die werden oft Zeichen für Zeichen und nicht als Wort vom Screenreader vorgelesen. Sie sind außerdem schwerer zu lesen und für Menschen mit Leseschwäche somit schwerer zu verstehen.
- Keine ASCII oder Unicode Symbole. Diese werden nicht übersetzt.
- Vermeide URL als Linktext. Der Screenreader lesen URL mitunter Buchstabe für Buchstabe vor.
- Begrenze den Text zu dem Link deutlich. 100 Zeichen sind absolute Obergrenze.
- Begrenze die Anzahl von Links
- auf einer Seite.
- Beschreibe Links auf Dateien zum Download entsprechend.
- Setz in den Link-Text nicht einfach nur eine Zahl. Beschreibe vielmehr: „Gehe zu Seite 2“
- Behalte die Benutzung von Keyboards im Auge. Links sollten „Tab“ sein.
- Links werden mit dem <a> Element und dem Attribut href=“„ erstellt. Besser keine Javascript „onclick“ Eventhandler bzw. Eventlistener auf andere Element anwenden. Onclick Events können mit einem <a> gewrappt werden.

Festlegen des Link-Ziels mit target=

Wir können festlegen, wo unser Link nach dem Klick geöffnet wird. Heute interessant[3] sind diese beiden Ergänzungen, wobei. target=“_self“ der Standard ist und normalerweise nicht extra angegeben sein muss.

```
<a href="1.html" target="_self">SELFHTML aktuell</a>
<a href="2.html" target="_blank">Öffnet im neues Browserfenster</a>
```

Defintition des Download-Datei Typs mit MIME

Idealerweise[4] wird im Link zu der Datei der MIME über den Zusatz type= näher beschreiben, MIME ist die „Multipurpose Internet Mail Extensions“. So kann das ideale Programm und Plugin zum Öffnen vom Browser besser gefunden werden. Im folgenden Beispiel ist der MIME zum Beispiel die Software Microsoft Word. Allerdings ist dies nur bei Dateitypen notwendig, welche nicht Webstandard (z.B. HTML, JPEG etc.) sind. Die MIME Typen sind seit 1996 in [RFC2045] und [RFC2046] definiert.

```
<a href="11GMTA-klassenliste.doc" type="application/msword">Klassenliste 11GMTA</a>
```

MIME-Typ	Beschreibung	Beispiele
text	Textdateien	`*.css; *.rtf; *.html; .txt;  *.js; *.xml`
image	Grafikdateien	`*.jpg; *.gif; *.png;*.tiff`
video	Videodateien	`*.mov; *.mp4`
audio	Audiodateien	`*.mp3; *.wav`
application	Ein bestimmtes Programm ist erforderlich	`*.doc; *.excel; *.pdf`
multipart	Mehrteilige Dateien	`z.B. Sprachnachrichten`
message	Nachrichten	`http, news`
model	Strukturen mit meheren Dimensionen	`*.vrml`

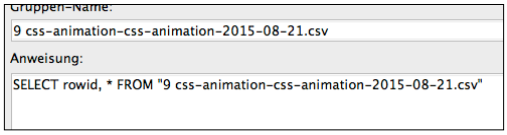
3 *Es gibt darüber hinaus noch historische Definitionen des Targets, diese basieren jedoch auf Frames und spielen im Webdesign wie es in diesem Script vorgestellt keine Rolle mehr. Diese beiden Relikte sind TARGET = „_parent“ und TARGET = „_top“. Mit Frames sind jedoch keine iFrames gemeint, diese sind durchaus interessant und auch Teil des Scripts.*

4 *Es gibt neben Stylesheet und MIME in der HTML Spezifikation wesentlich mehr Ergänzungen des Link Typs. Diese spielen häufig keine wesentliche Rolle in der beruflichen Praxis:* **http://www.w3.org/TR/html4/types.html#type-links**

Layout im Web

Gemeinsamkeiten bei Webdesign und Printdesign

Im Kern ist der Unterschied nicht groß. Sicherlich ist es ein größerer Unterschied mit InDesign zu arbeiten und eine Webseite per Hand zu coden. Jedoch läuft hinter den Kulissen ein vergleichbarer Code, auch wenn dies bei der gewöhnlichen Arbeit nicht sonderlich auffällt. PDF und sein Vorgänger Postscript beschreiben Inhalte vergleichbar zu HTML und CSS.



Beim Satz dieses Skriptes in InDesign kommt das Plugin EasyCatalog zum Einsatz. Es bietet unter anderem die Möglichkeit mit SQL Syntax Datenquellen wie MySQL oder CSV auszulese. Ein unglaublicher Produktionsvorteil gegenüber dem händischen Import solcher Tabellen über den Befehl „Platzieren".

Stilvorlagen, Absatzformate, Zeichenformate, Farbfelder und ähnliche Funktionen aus der Druckvorstufe sind dem Konzept CSS sehr ähnlich. Große Gemeinsamkeiten bei der Konzeption gibt es bei der datenbankgestützten Produktion von Printmedien. Hier muss das Layout und die Positionierung für größere Datenmengen geplant werden. Seiten, Rahmen und Tabellen sollen sich im Katalog automatisch aufbauen, mit Inhalt füllen oder Bilder automatisiert platziert werden. Spätestens wenn crossmediale Kampagnen produziert werden und neben dem Printmedium auch eBooks publiziert werden oder XML für Datenaustausch eingesetzt wird. Eine der wesentlichen Gemeinsamkeiten sind die Benutzung von *<Tags>*. In so einem modernen Arbeitsumfeld sind die Übergänge von Printdesign zu Webdesign sehr fließend

```
\section{Hallo Welt!}
```

Dieser Code ist aus der immer noch sehr populären Sprache LaTeX. In LateX können Bücher und andere Schriftstücke gesetzt werden. Das Beispiel oben hat die gleiche Funktion wie unser Tag <h1>. Es ist noch gar nicht so lange her, da wurden alle Schriftsachen gecodet. Das war die Revolution nach 400 Jahren Bleisatz.

Die Auflösung moderner Tabletts oder Smartphones ist gegenüber Monitoren zwei- bis dreimal so detailliert.

Unterschiede bei Webdesign und Printdesign

Im Gegensatz zu der Gestaltung mit Papier oder anderen festen Medien, kennen wir im Webdesign die von unseren Besuchern benutzten Anzeigegeräte nicht. Der Besucher kann die Webseite mit einem Smartphone, auf einem großen Monitor betrachten oder ein Tablett verwenden. Ein Pixel hat auch keine bekannte Größe, so gibt es meistens quadratische aber auch rechteckige Pixel bei Ausgabegeräten und sehr große Unterschiede bei der Bildschirmauflösung. Gestalte ich hingegen ein Plakat im Format DIN A2, dann sind die Breite und Höhe sowie das Ausgabegerät klar bestimmbar. Dieser Unterschied bedingt und ermöglicht im Vergleich zum Printdesign andere Formen der Positionierung.

»Be water, my friend.
Don't get set into one form,
adapt it and build your own,
and let it grow,
be like water.
Empty your mind,
be formless, shapeless —
like water.
Now you put water in a cup,
it becomes the cup;
You put water into a bottle
it becomes the bottle;
You put it in a teapot
it becomes the teapot.
Now water can flow
or it can crash.
Be water, my friend.«
Bruce Lee

Be water my friend

Sobald die Wirkung der CSS Eigenschaften auf den Fluss der HTML Elemente klar und damit konzeptionel greifbar ist, nutzen wir das volle Potential im Webdesign.

Ursprünglich waren nur relativ zueinander fließende Elemente in HTML und CSS möglich. Dieser Fluss wurde durch die Möglichkeit absoluter Positionierungen erweitert, jetzt kann auch pixelgenau und unabhängig vom Fluss gestalten werden.

Wir bleiben in diesem Kapitel mit den Größen und Werten bei Pixel und Prozent, da diese beiden Begriffe geläufig und für den Einstieg sehr griffig sind.

Wir können zwar Elemente Absolut und Fix positionieren, dies sollte die Ausnahme sein. Ein Layout im Fluss der Elemente sollte immer die Grundlage sein. Für den Fluss haben wir mittlerweile zwei Ansätze: FLOAT und FLEXBOX.

Die auf der folgenden Seite gezeigten Beispiele sind natürlich auf CODE verfügbar:

`http://code.arnoldbodeschule.de/float-und-clear/`

`http://code.arnoldbodeschule.de/flex-im-layout/`

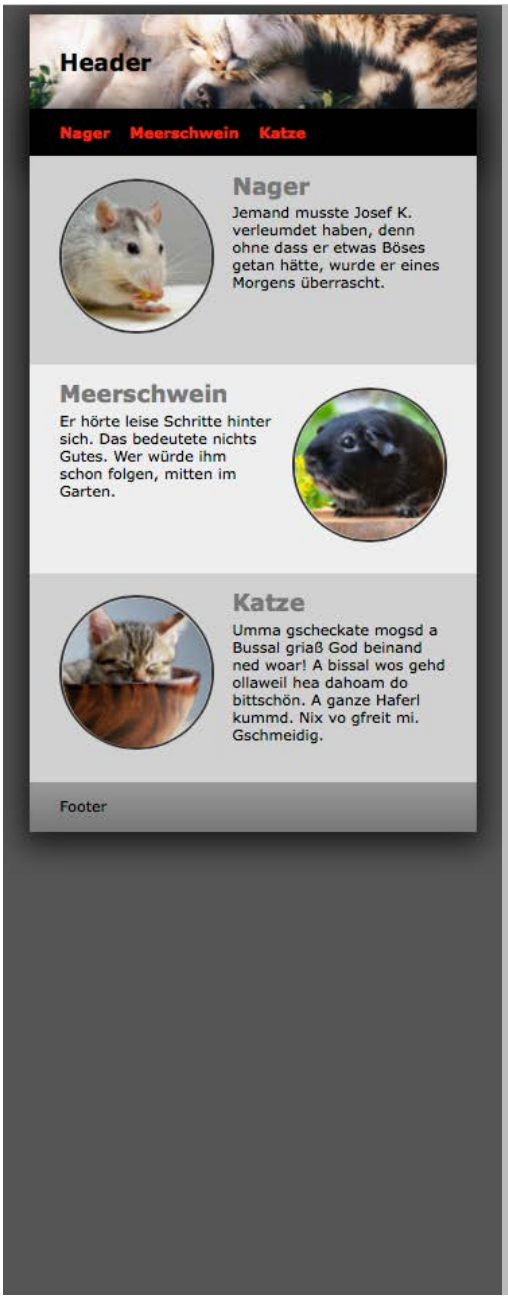
Layout mit Float und Clear

Mit FLOAT und CLEAR können ansprechende Layouts umgesetzt werden. FLOAT funktioniert besser in Layouts mit festen Vorgaben für die Höhe. Dies beinhaltet auch Vorgaben für maximale Längen von Text. Der CLEAR erfolgt mit der CLEARFIX Methode bei jedem ARTICLE. In diesem Beispiel wird mit Hilfe der CSS-Pseudoclass „NTH-OF-TYPE(n)" die Richtung der FLOATS und Hintergrundfarbe automatisch abgewechselt. Ursprünglich ist FLOAT aber nur für den Umfluss von Text um Bilder gemacht worden. Flexbox gehört die Zukunft. Float spielt aber noch eine Rolle.

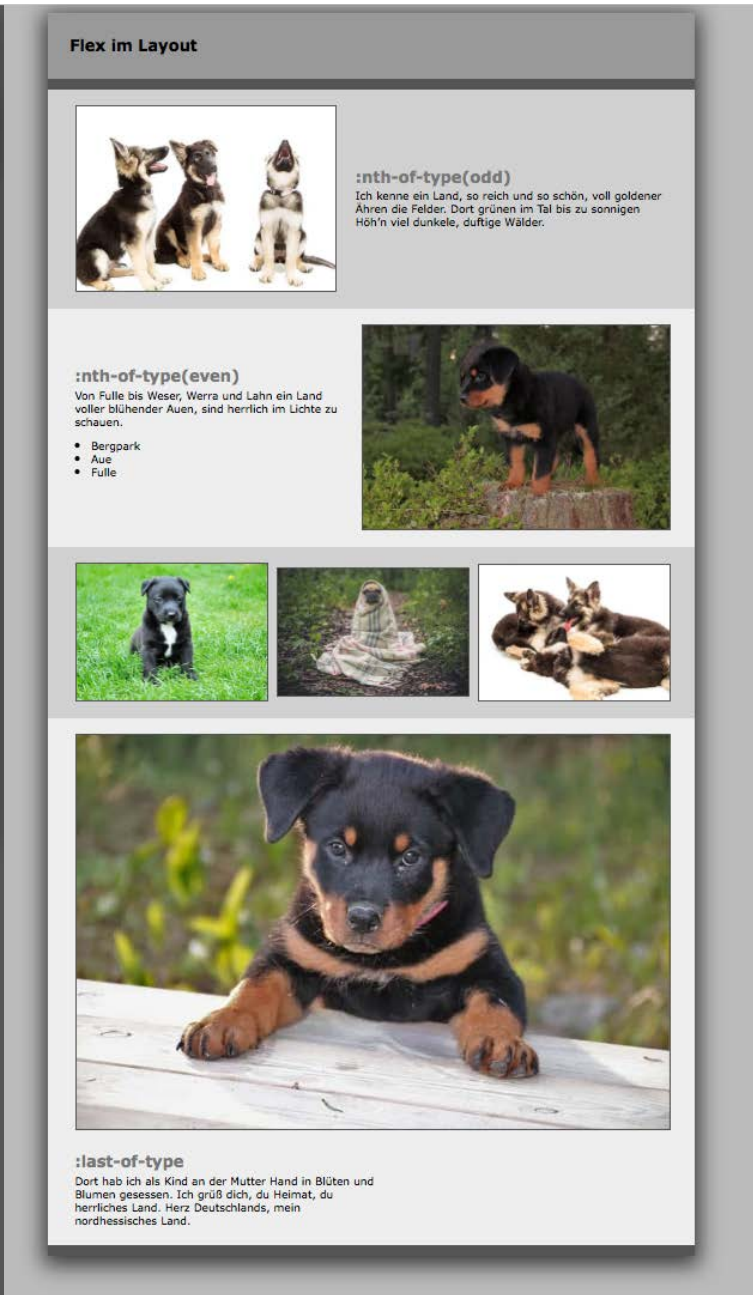
Layout mit Flexbox

FLEX ist für das Layout von Komponenten, z.B. NAV oder ARTICLE gemacht. Es ersetzt nicht die W3C GRID Candidate Recommendation für das Zusammenspiel der Bereiche im Sinne eines Gestaltungsrasters. Layout einer Seite mit FLEX bedeutet dass FLEX bei bestimmten Komponenten zum Einsatz kommt. FLEX gestaltet im Beispiel die einzelnen ARTICLE. Das restliche Layout erfolgt mit den konventionellen CSS Eigenschaften. Im Beispiel werden Variationen im Layout mit :PSEUDO-CLASS erzielt. Der letzte ARTICLE hat z. B. automatisch ein großes Bild. Bemerkenswert ist auch, dass FLEX bei Bildern „irgendwie" besser funktioniert, wenn diese in einen Container gepackt werden. Bei FLEX werden Breiten angepasst. Das ist ein Feature, kein Nachteil. Je nach Bild und Überschrift variiert das Layout ewas. Der große Vorteil von FLEX gegenüber FLOAT sind außerdem die dynamischen Höhen. Je nachdem wie Hoch der Inhalt ist, es funktioniert. Das ist bei FLOAT nicht der Fall. Bei FLOAT sind die Höhen am Besten fix eingestellt. FLEX wirkt nur auf der ersten Kinder-Ebene. Ein FLEX-Kind kann selber zu einem FLEX-CONTAINER werden.

Layout auf Basis von Float und Clear.
http://code.arnoldbodeschule.de/float-und-clear/



Layout auf Basis von Flexbox.
http://code.arnoldbodeschule.de/flex-im-layout/



Impressum

Es kann der Frömmste nicht in Frieden leben, wenn es dem Nachbarn nicht gefällt.

„Dienste-Anbieter haben für geschäfts-mäßige, in der Regel gegen Entgelt angebotene Telemedien folgende Infor-mationen leicht erkennbar, unmittelbar erreichbar und ständig verfügbar zu halten.“.

Telemediengesetz §5

Beispiel für ein Impressum

Angaben gemäß § 5 TMG

Max Muster
Musterweg
12345 Musterstadt
Vertreten durch:
Max Muster
Kontakt:
Telefon: 01234-789456
Fax: 1234-56789
E-Mail: max@muster.de

Registereintrag

Eintragung im Registergericht: Musterstadt
Registernummer: 12345

Umsatzsteuer-ID
Umsatzsteuer-Identifikationsnummer
gemäß §27a Umsatzsteuergesetz: Musterustid.

Wirtschafts-ID
Musterwirtschaftsid
Aufsichtsbehörde:
Musteraufsicht Musterstadt

Verantwortlich für den Inhalt nach § 55 Abs. 2 RStV:
Max Muster
Musterweg
12345 Musterstadt

Haftungsausschluss

Website Impressum erstellt durch impressum-generator.de

Haftung für Inhalte

Die Inhalte unserer Seiten wurden mit größter Sorgfalt erstellt. Für die Richtigkeit, Vollständigkeit und Aktualität der Inhalte können wir jedoch keine Gewähr übernehmen. Als Diensteanbieter sind wir gemäß § 7 Abs.1 TMG für eigene Inhalte auf diesen Seiten nach den allgemeinen Gesetzen verantwortlich. Nach §§ 8 bis 10 TMG sind wir als Diensteanbieter jedoch nicht verpflichtet, übermittelte oder gespeicherte fremde Informationen zu überwachen oder nach Umständen zu forschen, die auf eine rechtswidrige Tätigkeit hinweisen. Verpflichtungen zur Entfernung oder Sperrung der Nutzung von Informationen nach den allgemeinen Gesetzen bleiben hiervon unberührt. Eine diesbezügliche Haftung ist jedoch erst ab dem Zeitpunkt der Kenntnis einer konkreten Rechtsverletzung möglich. Bei Bekanntwerden von entsprechenden Rechtsverletzungen werden wir diese Inhalte umgehend entfernen.

Haftung für Links

Unser Angebot enthält Links zu externen Webseiten Dritter, auf deren Inhalte wir keinen Einfluss haben. Deshalb können wir für diese fremden Inhalte auch keine Gewähr übernehmen. Für die Inhalte der verlinkten Seiten ist stets der jeweilige Anbieter oder Betreiber der Seiten verantwortlich. Die verlinkten Seiten wurden zum Zeitpunkt der Verlinkung auf mögliche Rechtsverstöße überprüft. Rechtswidrige Inhalte waren zum Zeitpunkt der Verlinkung nicht erkennbar. Eine permanente inhaltliche Kontrolle der verlinkten Seiten ist jedoch ohne konkrete Anhaltspunkte einer Rechtsverletzung nicht zumutbar. Bei Bekanntwerden von Rechtsverletzungen werden wir derartige Links umgehend entfernen.

Urheberrecht

Die durch die Seitenbetreiber erstellten Inhalte und Werke auf diesen Seiten unterliegen dem deutschen Urheberrecht. Die Vervielfältigung, Bearbeitung, Verbreitung und jede Art der Verwertung außerhalb der Grenzen des Urheberrechtes bedürfen der schriftlichen Zustimmung des jeweiligen Autors bzw. Erstellers. Downloads und Kopien dieser Seite sind nur für den privaten, nicht kommerziellen Gebrauch gestattet. Soweit die Inhalte auf dieser Seite nicht vom Betreiber erstellt wurden, werden die Urheberrechte Dritter beachtet. Insbesondere werden Inhalte Dritter als solche gekennzeichnet. Sollten Sie trotzdem auf eine Urheberrechtsverletzung aufmerksam werden, bitten wir um einen entsprechenden Hinweis. Bei Bekanntwerden von Rechtsverletzungen werden wir derartige Inhalte umgehend entfernen.

Datenschutz

Die Nutzung unserer Webseite ist in der Regel ohne Angabe personenbezogener Daten möglich. Soweit auf unseren Seiten personenbezogene Daten (beispielsweise Name, Anschrift oder eMail-Adressen) erhoben werden, erfolgt dies, soweit möglich, stets auf freiwilliger Basis. Diese Daten werden ohne Ihre ausdrückliche Zustimmung nicht an Dritte weitergegeben.
Wir weisen darauf hin, dass die Datenübertragung im Internet (z.B. bei der Kommunikation per E-Mail) Sicherheitslücken aufweisen kann. Ein lückenloser Schutz der Daten vor dem Zugriff durch Dritte ist nicht möglich.
Der Nutzung von im Rahmen der Impressumspflicht veröffentlichten Kontaktdaten durch Dritte zur Übersendung von nicht ausdrücklich angeforderter Werbung und Informationsmaterialien wird hiermit ausdrücklich widersprochen. Die Betreiber der Seiten behalten sich ausdrücklich rechtliche Schritte im Falle der unverlangten Zusendung von Werbeinformationen, etwa durch Spam-Mails, vor.

Google Analytics

Diese Website benutzt Google Analytics, einen Webanalysedienst der Google Inc. (‚Google"). Google Analytics verwendet sog. ‚Cookies", Textdateien, die auf Ihrem Computer gespeichert werden und die eine Analyse der Benutzung der Website durch Sie ermöglicht. Die durch den Cookie erzeugten Informationen über Ihre Benutzung diese Website (einschließlich Ihrer IP-Adresse) wird an einen Server von Google in den USA übertragen und dort gespeichert. Google wird diese Informationen benutzen, um Ihre Nutzung der Website auszuwerten, um Reports über die Websiteaktivitäten für die Websitebetreiber zusammenzustellen und um weitere mit der Websitenutzung und der Internetnutzung verbundene Dienstleistungen zu erbringen. Auch wird Google diese Informationen gegebenenfalls an Dritte übertragen, sofern dies gesetzlich vorgeschrieben oder soweit Dritte diese Daten im Auftrag von Google verarbeiten. Google wird in keinem Fall Ihre IP-Adresse mit anderen Daten der Google in Verbindung bringen. Sie können die Installation der Cookies durch eine entsprechende Einstellung Ihrer Browser Software verhindern; wir weisen Sie jedoch darauf hin, dass Sie in diesem Fall gegebenenfalls nicht sämtliche Funktionen dieser Website voll umfänglich nutzen können. Durch die Nutzung dieser Website erklären Sie sich mit der Bearbeitung der über Sie erhobenen Daten durch Google in der zuvor beschriebenen Art und Weise und zu dem zuvor benannten Zweck einverstanden.

Google AdSense

Diese Website benutzt Google Adsense, einen Webanzeigendienst der Google Inc., USA (‚Google"). Google Adsense verwendet sog. ‚Cookies" (Textdateien), die auf Ihrem Computer gespeichert werden und die eine Analyse der Benutzung der Website durch Sie ermöglicht. Google Adsense verwendet auch sog. ‚Web Beacons" (kleine unsichtbare Grafiken) zur Sammlung von Informationen. Durch die Verwendung des Web Beacons können einfache Aktionen wie der Besucherverkehr auf der Webseite aufgezeichnet und gesammelt werden. Die durch den Cookie und/oder Web Beacon erzeugten Informationen über Ihre Benutzung diese Website (einschließlich Ihrer IP-Adresse) werden an einen Server von Google in den USA übertragen und dort gespeichert. Google wird diese Informationen benutzen, um Ihre Nutzung der Website im Hinblick auf die Anzeigen auszuwerten, um Reports über die Websiteaktivitäten und Anzeigen für die Websitebetreiber zusammenzustellen und um weitere mit der Websitenutzung und der Internetnutzung verbundene Dienstleistungen zu erbringen. Auch wird Google diese Informationen gegebenenfalls an Dritte übertragen, sofern dies gesetzlich vorgeschrieben oder soweit Dritte diese Daten im Auftrag von Google verarbeiten. Google wird in keinem Fall Ihre IP-Adresse mit anderen Daten der Google in Verbindung bringen. Das Speichern von Cookies auf Ihrer Festplatte und die Anzeige von Web Beacons können Sie verhindern, indem Sie in Ihren Browser-Einstellungen ‚keine Cookies akzeptieren" wählen (Im MS Internet-Explorer unter ‚Extras > Internetoptionen > Datenschutz > Einstellung"; im Firefox unter ‚Extras > Einstellungen > Datenschutz > Cookies"); wir weisen Sie jedoch darauf hin, dass Sie in diesem Fall gegebenenfalls nicht sämtliche Funktionen dieser Website voll umfänglich nutzen können. Durch die Nutzung dieser Website erklären Sie sich mit der Bearbeitung der über Sie erhobenen Daten durch Google in der zuvor beschriebenen Art und Weise und zu dem zuvor benannten Zweck einverstanden.

Beispiel Kostenvoranschlag

Manfred Mustermann
Gepr. Medienfachwirt
Musterstraße. 10
34130 Kassel
info@server.net

(0561) 12 34 34 34

Firma XXX
Musterweg 4
34131 Kassel
z.H. Frau XXX

Steuernummer 26 XXX XXXX 6 - G09
Identifikationsnummer 80 XXX XXX XX 761
IBAN DE10 XXXX XXXX XXXX XXX XXXX
BIC HELA XXXX XXXX XXXX
Kasseler Sparkasse

Spätere Rechnungsnummer: AW_000001 . . . . . Kassel, 17.XX.20XX

Sehr geehrte Frau XXX,
wie vereinbart präsentiere ich Ihnen den **Kostenvoranschlag**.

Homepage Entwurf . . . . . Zeitraum XX.XX bis XX.XX 20XX
Erstbesprechung . . . . . Service
Scribble . . . . . Service
Besprechung . . . . . Service
Verfeinerung des Entwurf 4
Struktur der Inhalte und Menuplanung 2
Besprechung Struktur und Menu . . . . . Service
Reinzeichnung der Homepage 2
Besprechung der Reinzeichnung . . . . . Service

Logo . . . . . Zeitraum XX.XX bis XX.XX 20XX
Entwürfe 2
Besprechung . . . . . Service
Verfeinerung des Entwurf 2
Reinzeichnung und Bereitstellung für verschiedene Medien 2

Kostenvoranschlag Entwurf und Logo (zahlbar Anfang XX 20XX) . . . . . 1120,00 Euro

Bitte beachten Sie die zweite Seite.

Homepage Produktion . . . . . Zeitraum XX.XX bis XX.XX 20XX
Datenbank einrichten. 1
Wordpress installieren. 1
CSS, HTML und PHP anpassen 10
Inhaltsseiten anlegen, Bilder und Texte einpflegen, Anmeldeformular etc.. 4
Google eintragen, Google Places etc. 1
Wordpress Schulung . . . . . Service
Funktionstest der Seite . . . . . Service
Kostenvoranschlag Homepage Produktion (zahblbar wenn Online). . . . . 1350,00 Euro

Printmedien. . . . . . Zeitraum XX.XX bis XX.XX 20XX
Visitenkarte Entwurf 1
Flyer Entwurf 2
Bildbearbeitung für hohe Druckqualität 1
Stempelvorlage, Klingelschild, etc. . . . . Service
Besprechung Drucksachen . . . . . Service
Reinzeichnung Flyer 2
Reinzeichnung Visitenkarte 1
Korrekturdurchgänge. 1
Erzeugung druckfähiger PDF Dokumente, Versand an Dienstleister etc.. 1
Kostenvoranschlag Printmedien (zahblbar nach Anlieferung). . . . . 720,00 Euro

Bitte beachten Sie, dass für den Druck und den Server die Kosten durch die externen Dienstleister bestimmt werden.
Ich empfehle folgende Dienstleister aufgrund sehr guter Qualität und günstiger Preise.

Übersicht weiterer Kosten . . . . . Zeitraum XX.XX bis XX.XX 20XX
Server, Domain für Homepage pro Monat. . . . . 5,00 Euro
zzgl. einmalige Einrichtungsgebühr . . . . . 15,00 Euro

Flyer DIN Lang Wickelfalz, CMYK, 160g, PT1, Druckerei Hayn, 2500 Stück . . . . . 110,00 Euro
Flyer DIN Lang Wickelfalz, CMYK, 160g, PT1, Druckerei Hayn, 1000 Stück . . . . . 65,00 Euro
Stempel. . . . . 30,00 Euro

Die spätere Rechnung wird aufgrund des § 19 UStG ohne Ausweisung der Mehrwertsteuer ausgestellt.
Ich hoffe, Sie sind mit der bisherigen Arbeit sehr zufrieden und ich freue mich auf die weitere Zusammenarbeit.

Mit freundlichen Grüßen
Manfred Mustermann

CODE Styleguide für HTML und CSS

Generelle Tipps für besseren Quellcode

Tipp	Beschreibung
Divida et impera	Teile und herrsche. Brich die Komponenten auf ein für Dich verständliches Niveau herunter. Das gilt besonders wenn Du ein Projekt übernimmst. Druck es Dir aus. Gliedere die Bereiche. Markiere die Funktionen und recherchiere alles was Dir unbekannt ist. Bring den Code dann in eine Struktur die für Dich bzw. Euch üblich ist und füge deine Erklärungen als Kommentare an den entsprechenden Stellen ein.
Einzug	Gestalte deinen Texteinzug mit 2 Leerzeichen. Verwende keine Tabs. Mixe auch keine Tabs und Leerzeichen für die Einzüge.
Englisch	CODE ist eine Plattform für Einsteiger. Daher wird hier viel Wert auf verständliche Sprache gelegt. Die eigentliche Sprache für Web ist Englisch. Da kommt man nicht vorbei und sollte seine Projekte auch entsprechend anlegen und dies konsequent bei Namensgebung und Kommentaren durchziehen.
Kein Modor- oder Highlander	Bau deine Klassen im CSS oder Funktionen im JS modular auf. Vermeide gigantische Klassen in denen alle Eigenschaften definiert sind. Diese Modor-Klassen (eine um alles zu knechten) bzw. Highlander-Klassen (es kann nur eine geben) machen die spätere Pflege und Korrektur meist extrem aufwendig. Außerdem kann man solche Modor/Highlander Klassen bzw. Funktionen schlecht von einem in das nächste Projekt übertragen.
Kleinschreibung	Benutze für Quellcode immer nur die Kleinschreibung.
Kommentare	Kommentare dienen zur Erklärung von Quellcode. Sie sind direkt dort zu setzen, wo Erklärungs-, bzw. Hinweisbedarf besteht. Sinnvoll sind Kommentare für Angaben wie: Wofür benötige ich den Codeabschnitt? Warum habe ich mich für diesen Lösungsweg entschieden ? Oder einfach nur als Markierung. Kommentare sind selbstverständlich optional und nicht verpflichtend. 15 Minuten saubere Kommentierung können Monate später Stunden der Einarbeitung vermeiden.
Modulare Konzeption	Bau deine Projekte so auf, dass Du die einzelnen Module (Komponenten) leicht in anderen Projekten erneut nutzen kannst. Bevorzuge generell eine zeitlose Namensgebung und verfasse aussagekräftige Kommentare um auch nach 1-2 Jahren schnell wieder hineinzufinden.
Unnötige Leerzeichen oder Tabs	Entferne alle unnötigen Leerzeichen oder Tabs. Insbesonders nachstehender Leerzeichen sind unnötig und können automatische Datei-Vergleiche umständlicher machen.

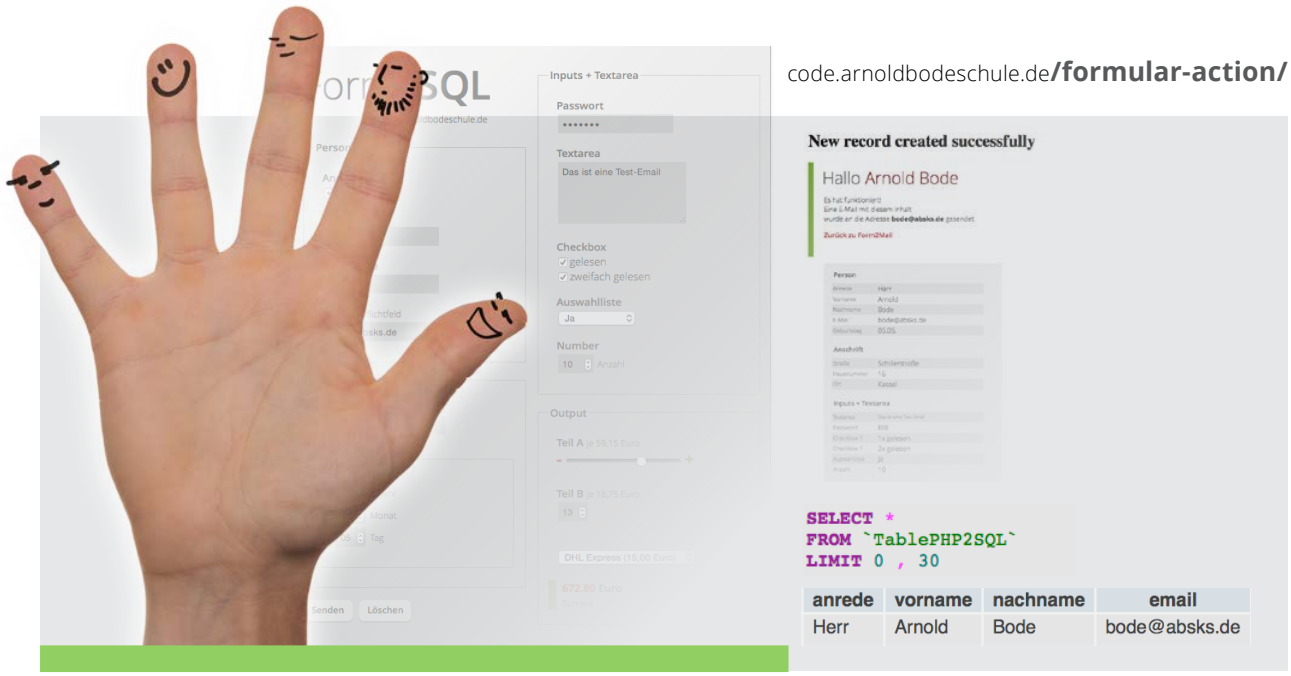
Tipps für HTML

Tipp	Beschreibung
** -Suppe of hell**	Bau dein Projekt nach semantischen Gesichtspunkten auf. Benutzt Du für alle Elemente einfach nur ein DIV, dann landest Du in der -Suppe of hell.
Anführungszeichen	Benutze doppelte Anführungszeichen (" ") statt einfache Anführungszeichen (' ').
HTML Elemente	Jedes Elements sollte in einer neuen Zeile stehen. Kinder-Elemente werden eingerückt.
HTML Validität	Benutze gültiges HTML 5.
Multimedia Fallback	Halte alternative Inhalte für verschiedene Multimedia-geräte bereit. Für Bilder (omg) z.B. das alt-Attribut.
Semantik	Benutze HTML-Elemente wofür sie bestimmt sind. Benutze zum Beispiel h-Elemente nur für Überschriften und p-Elemente nur für Absätze. Bau auch keine Javascript Konstrukte wenn es bereits HTML Elemente für diese Funktion gibt.
Trennung von Funktionsbereichen	Trenne die Bereiche Styling (CSS) und Struktur (HTML) strikt von einander. Versuche die Überschneidungen zwischen den Bereichen, z.B. durch eine Verwendung des style Attribut bei HTML Elementen so gering wie möglich zu halten. Das sollten immer nur Ausnahme-Fälle sein.
Type-Attribute	Du brauchst kein "Typ"-Attribute bei Stylesheets und Skripten, es sei denn du benutzt eine andere Styling-Sprache als CSS oder eine andere Skripting-Sprache als Javascript.
UTF-8	Stelle sicher das dein Editor UTF-8 als Zeichenkodierung benutzt.
Zeichenreferenzen	Benutze keine Zeichenreferenzen. Wenn der Zeichensatz auf UTF-8 gesetzt ist, ist es nicht notwendig Zeichenreferenzen wie —, ” oder ☺ zu benutzen. Die einzige Ausnahme sind Zeichen die die HTML-Syntax stören können wie Beispielsweise "<" oder ">".

Tipps für CSS

Tipp	Beschreibung
0 wird ausgeschrieben	Bei Werten wird die 0 angegeben. Es macht deutlicher dass die Zahl kleiner als 1 ist.
Abgrenzung von Abschnitten	Bereiche im CSS können mit auffälligen Kommentaren abgegrenzt werden. Bei größeren CSS Angaben hilft auch ein Inhaltsverzeichnis am Anfang. Zwischen Abschnitten werden 3 Leerzeilen gesetzt.
Abgrenzung zwischen Eigenschaft und Wert.	Es ist immer ein Leerzeichen zwischen der Eigenschaft und dem dazugehörigen Wert zu setzen.
Abgrenzung zwischen Regeln	Zwischen zwei verschiedenen Regeln sollte immer 1 Leerzeile gesetzt werden. Dies sollte im Dokument systematisch umgesetzt werden.
Abgrenzung zwischen Selektor und {	Es ist immer ein Leerzeichen zwischen dem Selektor und der öffnenden geschweiften Klammer { zu setzen. Die öffnende Klammer sollte zudem immer auf derselben Linie wie der Selektor sein.
Abgrenzung zwischen von Eigenschaften.	Jede Eigenschaft muss mit einem Semikolon beendet werden. Die nächste Eigenschaft beginnt in einer neuen Zeile.
Animationen	Animationen werden mit CSS umgesetzt. Benutze dafür kein Javascript. Animiere für bessere Geschwindigkeit am Besten nur: Transform; Opacity; Color; Border-Color; Background-Color.
Browser-Unterstützung	Kläre die Browser und Generationen für die Du dein Projekt erstellst und passe die Vendor-Prefixe für diesen Einsatzbereich an. Entwickle in einem modernen Browser wie Google Chrome und nutze dann für die benötigten Vendor-Prefixe Tools wie "Autoprefixer".
CLASS für Styling	Benutz Klassen für die Gestaltung und gehe dabei Modulorientiert vor.
CSS Selektion	Vermeide Kombinationen die Kombination Typ-Seelektoren und ID/Klasse. Die Kombination aus Klasse/ID und Typ-Selektor hingegen kann Sinn machen.
CSS Validität	Benutze gültiges CSS 3.
Hierarchie	Alle CSS Regeln sollten der Hierarchie entsprechend eingerückt werden.
ID nur für Identifikation	Style keine ID im CSS. Benutz ID nur um ein Element eindeutig zu identifizieren. Styling erfolgt mit Klassen.
ID und Klassen-Benennung	Benutze sinnvolle ID- und Klassen-Namen. Anstatt präsentative oder kryptische Namen zu verwenden, sollten IDs und Klassen immer den Zweck des fraglichen Elements wiedergeben oder in anderer Form sinnstiftend sein.
ID und Klassen-Benennungs-Stil	Benutze ID und Klassen-Namen die so kurz wie möglich sind aber so lang wie nötig. Versuche zu vermitteln worauf sich eine ID oder Klasse bezieht, während du dich so knapp wie möglich ausdrückst. Im Zweifelsfall liegt die Priorität immer auf der Verständlichkeit gegenüber der Knappheit.
Kombination von Selektoren	Vermeide extreme Kombination von Selektoren mit dem Komma. Arbeite stattdessen mit modulorientierten Klassen, der Vererbung im CSS und ggf. über die Namensgebung der Module bei der Attribute Selektion.
Kurzschreibweisen von Eigenschaften	Benutze Shorthand Properties wann immer möglich. CSS offeriert eine Vielfalt an Shorthand Properties wie z.B. background, font, border oder animation.
margin-collapse	Treffen sich ein margin-bottom und ein margin-top, dann wirkt nur der Größere. Gewöhn Dir am Besten an für Texte Margin nur in eine Richtung zu vergeben.
Muster mit Pseudo-Klassen	Erfolgt das Styling über ein konkretes Muster kann es mit Pseudo-Klassen umgesetzt werden.
Reihenfolge der CSS-Eigenschaften	Überlege Dir ein Konzept für die Reihenfolge der CSS-Eigenschaften.
Tiefe der CSS Selektion	Versuche mit 1 bis 2 Ebenen der CSS-Kaskade auszukommen.

Form2SQL



code.arnoldbodeschule.de/**formular-action/**

Fünf gute Freunde haben wir in der Webentwicklung: HTML, CSS, Javascript, PHP und SQL. In dem kleinen Projekt Form2SQL haben alle ihren PLatz.

Daten von einem Formular prüfen, in eine Datenbank eintragen und per E-Mail senden

Bisher haben wir die Möglichkeiten mit HTML und CSS recht gut kennen gelernt. Es gibt natürlich viele Varianten um eine Webseite zu gestalten. Einige Funktionen sind jedoch häufig wichtig, die eben NICHT mit HTML und CSS umgesetzt werden können. Dazu gehören so alltägliche Dinge wie das sichere Versenden der Daten aus einem HTML Formular.

An der Stelle zeigt Form2SQL das Zusammenspiel von HTML, CSS, Javascript, PHP und MySQL. Dafür bedarf es unbedingt weitere Software wie einem guten Editor (Sublime Text 2) und einem Apache Server (z.B. Xampp). Die sind kostenlos und auf CODE verlinkt.

- Einrichten der Datenbank und des Benutzers
- Übersicht HTML, CSS, Javascript, PHP und MySQL
- Struktur des HTML Formulars
- Gestaltung des Formulars mit CSS
- Prüfung mit HTML
- Prüfung der E-Mail mit Javascript.
- Übergabe der Daten an PHP
- Versenden der Daten per E-Mail mit PHP
- Verbinden mit der Datenbank
- Eintrag der Daten in die Datenbank
- Erfolgsmeldung

Die verschiedenen Varianten des Formular und natürlich das Paket Form2SQL mit Links zu empfohlener Software:

code.arnoldbodeschule.de/formular-action/

Übersicht HTML, CSS, Javascript, PHP und MySQL

Sprache	Beschreibung	Umgebung
HTML	Seitenbeschreibungsprache für digitale Medien. HTML definiert den Inhalt. HTML bedeutet „Hypertext Markup Language". Wurde 1992 erstmals öffentlich onnline eingesetzt.	Browser (Client-Seitig). Safari, Firefox, Chrome, etc.
CSS	Die Sprache CSS beschreibt die Ausgabe der HTML Elemente. Dabei bauen die CSS Elemente aufeinander auf. Standardmässig hat jedes HTML Element bereits zugewiesene CSS Eigenschaften. Diese gelten bis man eigene definiert. CSS bedeutet „Cascading Style Sheet". Wurde 1994 von Håkon Wium Lie bei der W3C vorgestellt.	Browser (Client-Seitig) Safari, Firefox, Chrome, etc.
Javascript (bzw. JQuery)	Skriptsprache für dynamisches Verhalten in Browsern. Z.B. Plausibilitätsprüfung (Datenvalidierung) von Formulareingaben, Anzeige von Dialogfenstern, aber auch Grafik und Animation. JQuery ist eine darauf aufbauende Bibliothek bei der es wichtige Funktionen bereits fertig gibt. Wurde 1995 von Netscape entwickelt.	Browser (Client-Seitig) <script>, jQuery, Safari, Firefox, Chrome, etc.
PHP	Populärste serverseitiger Preprocessor. Wird für besondere, meist dynamische Funktionen verwendet. Z.B. um die Verbindung mit einer Datenbank herzustellen oder andere Funktionen wie z.B. das Versenden von E-Mails. PHP bedeutet „Hypertext Preprocessor". Wurde 1995 von Rasmus Lerdorf entwickelt.	z.B. Apache Server (Server-Seitig) PHP 5.6
MySQL	Populärste relationale Datenbank-Management-Software als Open-Source. Wird auch bei sehr großen Unternehmen wie Facebook, Google oder Adobe eingesetzt und ist nicht nur kostenlos sondern sehr gut dokumentiert. Wurde 1994 von MySQL AB entwickelt. SQL bedeutet „Structured Query Language".	z.B. Apache Server (Server-Seitig) MySQL

Einrichten der Datenbank und des Benutzers

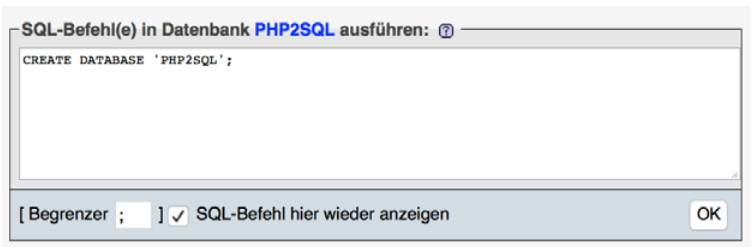
Auf dem Server wird die Datenbank und dann die Tabelle angelegt. Für den Einstieg brauchen wir in der Tabelle nur den Primärschlüssel, sowie anrede, vorname, nachname und emai. Das einrichten der Datenbank geschieht über SQL. Man kann bei Xampp dies auch mit einer graphischen Benutzeroberfläche machen.

Als Benutzer liefert Xampp einen Standardnutzer mit dem Namen „root" und ohne Passwort. Für lokale Versuche ist das praktisch. Aber bei einer Veröffentlichung im Internet unbedingt ein sicheren Namen und Passwort anlegen.

Wenn eine Email versendet werden soll, dann muss Xampp auch entsprechend eingestellt werden. Bei einem Online-Server ist dies meist schon aktiv.

```
CREATE DATABASE `PHP2SQL`;
```

```
CREATE TABLE  `PHP2SQL`.`TablePHP2SQL` (
`PrimaryKey` INT( 3 ) NOT NULL AUTO_INCREMENT PRIMARY KEY ,
`anrede` VARCHAR( 50 ) CHARACTER SET utf8 COLLATE utf8_bin NOT NULL ,
`vorname` VARCHAR( 50 ) CHARACTER SET utf8 COLLATE utf8_bin NOT NULL ,
`nachname` VARCHAR( 50 ) CHARACTER SET utf8 COLLATE utf8_bin NOT NULL ,
`email` VARCHAR( 05 ) CHARACTER SET utf8 COLLATE utf8_bin NOT NULL
) ENGINE = MYISAM CHARACTER SET utf8 COLLATE utf8_bin;
```



Struktur des HTML Formulars

Die Datei index.html enthält die HTML Daten. Sie heisst index.html da der Browser diese dann später automatisch öffnet. Besonders wichtig ist die definierte action. Dem Formular ist dort die Daten send.php zugeordnet. Außerdem muss jedem Feld von dem wir die Daten verarbeiten möchten ein Name zugeordnet sein. Der Jacascript „check.js“ wird geladen.

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset='UTF-8'>
        <title>Form2SQL</title>
        <link href='http://fonts.googleapis.com/css?family=Open+Sans:300,600' rel='stylesheet' type='text/css'> <!--Lädt die Schrift Open Sans in Light und Bold-->
        <link rel="stylesheet" href="style.css">
        <script language="javascript" type="text/javascript" src="check.js"></script>                          <!--Lässt eine Fehlermeldung erscheinen wenn E-Mail nicht ausgefüllt ist--
    </head>
    <body>
        <form action="send.php" method="POST" name="Form2Mail" target="_self" accept-charset="UTF-8"
              onsubmit="return validateForm()"
              autocomplete='off' oninput='summe.value = (parseFloat(TeilA.value) * 59.15 + parseFloat(TeilB.value) * 18.75 + parseFloat(versand.value)).toFixed(2)'>
            <h4 class="title">Form<i>2</i><strong>SQL</strong></h4>
            <a class="home" href="http://www.code.arnoldbodeschule.de" target="_blank"><strong>code</strong>.arnoldbodeschule.de</a>
            <fieldset id="Person">
                <legend>Person</legend>
                <p>
                    <label for='radio'>Anrede</label><br />
                    <input type='radio' id='radio1' value='Herr' name='anrede' >Herr
                    <input type='radio' id='radio2' value='Frau' name='anrede' >Frau
                </p>
                <p>
                    <label for='textare'> Vorname</label><br />
                    <input type='text' name='vorname' maxlength='30'>
                </p>
                 <p>
                    <label for='textare'> Nachname</label><br />
                    <input type='text' name='nachname' maxlength='30'>
                </p>
                <p>
                    <label for='emailID'>E-mail</label><small>* Pflichtfeld</small><br />
                    <input type='email' name='email' id='emailID' maxlength='40' required>
                </p>
            </fieldset>
            <fieldset id='Anschrift'>
                    <legend>Anschrift</legend>
                    <input type='text' id='str' name='str' size='17' placeholder='Straße'>
                    <input type='text' id='hnr' name='hnr' size='3' pattern='\d{1,3}' placeholder='Nr.'><br />
                    <input type='text' id='plz' name='plz' size='5' pattern='\d{5}' placeholder='PLZ' >
                    <input type='text' id='ort' name='ort' size='15' placeholder='Ort'>
            <br/><br/>
            <fieldset>
                <legend>Geburtstag</legend><p>
                <input type='number' min='1900' max='2020' step='1' id='jahr' name='GJahr' placeholder='1990'> <small>Jahr</small><br/>
                <input type='number' min='1' max='12' step='1' id='monat' name='GMonat' placeholder='1'> <small>Monat</small><br/>
                <input type='number' min='1' max='32' step='1' id='tag' name='GTag' placeholder='1'> <small>Tag</small><br/></p>
            </fieldset>
            </fieldset>
             <div id="Control">
                <button type='send'>Senden</button> <button type='reset'>Löschen</button>
            </div>
             <fieldset id="input">
                    <legend>Inputs + Textarea</legend>
            <p>
                <label for='password'>Passwort</label><br />
                <input type='password' name='pass' maxlength='20'>
            </p>
            <p class="float">
                <label for='nachricht'>Textarea</label><br />
                <textarea rows='5' id='nachricht' name='nachricht' maxlength='70' wrap='soft'></textarea>
            </p>
            <p>
                <label for='checkbox'>Checkbox</label><br />
                <input type='checkbox' id='checkbox1' value='1x gelesen' name='AGB1'>gelesen<br/>
                <input type='checkbox' id='checkbox2' value='2x gelesen' name='AGB2'>zweifach gelesen
            </p>
            <p>
            <label for='Auswahl'>Auswahlliste</label><br />
            <select name='auswahl'>
                    <optgroup label='klar'>
                        <option value='Ja'>Ja</option>
                        <option value='nein'>nein</option>
                    </optgroup>
                    <optgroup label='unklar'>
                        <option value='vielleicht'>vielleicht</option>
                        <option value='eher nein'>eher nein</option>
                        <option value='eher ja'>eher ja</option>
                    </optgroup>
            </select>
            <p>
                <label for='number'>Number</label><br />
                <input type='number' min='0' max='100' step='5' id='number' name='number'> <small>Anzahl</small><br/>
            </p>
            </fieldset>
            <fieldset id='Output'>
                <legend>Output</legend>
                <p>
                    <label for='range'>Teil A</label> <small>je 59,15 Euro</small><br />
                     <strong class='minus'>-</strong> <input type='range' id='TeilA' name='TeilA' min='0' max='10' step='1' value='0'> <strong class='plus'>+</strong>
                     </p>
                     <p>
                    <label for='Output'>Teil B</label> <small>je 18,75 Euro</small><br />
                    <input type='number' min='0' max='20' step='1' id='TeilB' name='TeilB' value='0'><br/><br/>
                    </p>
                    <p>
                    <select id='versand' name='versand'>
                        <option value='15.00'>DHL Express (15,00 Euro)</option>
                        <option value='4.25'>DHL normal (4,25 Euro)</option>
                        <option value='0.00'>Selbstabholer (0,00 Euro) </option>
                    </select>
                    </p>
                    <h4><output for='TeilA TeilB versand' name='summe'>0.00</output> Euro<br/>
                    <small>Summe</small></h4>
                </p>
            </fieldset>
        </form>
```

Gestaltung des Formulars mit CSS

```css
body          { background: #fff;}
*             { transition: all 0.4s ease-out; }                /* Erzeugt animierten Übergang bei geänderten Eigenschaften bei Pseudoklassen */


form   {
          border: 1px solid #555;
          margin: 30px 0 0 5px ;
          padding: 5px 10px 15px 10px;
          width: 560px;
          box-shadow: 0px 2px 3px 0px rgba(0,0,0,0.3);
          background: linear-gradient(135deg, #ededed 20%,#999 100%); /* Verlauf */
          -webkit-columns: 2 215px; -webkit-column-gap: 10px;      /* 2 Spalten und Abstand für Webkit Browser (Safari, Chrome) */
          -moz-columns: 2 215px; -moz-column-gap: 10px;            /* 2 Spalten und Abstand fpr Mozilla Browser (Firefox) */
       }


p      {
          font: 300 13px/18px Open Sans;
          margin-left: 10px;
       }


button  {
          font: 600 11px/18px Open Sans;
          background-color: rgba(250,250,245,0.6);
          border-radius:6px;                                  /* Abgerundete Kanten */
          border: 0px solid #bbb;
          outline: 0px;
          padding: 3px 10px 5px 10px;
          box-shadow: 0px 0px 0px 0px rgba(255,255,255,0.0),
                      0px 1px  1px  0px rgba(0,0,0,0.4);

          color: #000; margin-bottom: 40px;
       }


button:hover    {
                    background-color: #fff;
                    border-radius: 4px;
                    border: 0px solid #fff;
                    box-shadow:0px 0px 20px 2px rgba(255,255,255,0.85),
                               0px 1px 2px 0px rgba(0,0,0,0.5);   /* Der 1. Wert ist der Fancy Glow und der 2. Wert der eigentliche Schlagschatten */
                }


input, textarea     {
                        border: 0px solid;
                        padding: 5px;
                        background: rgba(20,20,10,0.15);
                        margin: 2px;
                    }


input:focus,textarea:focus  {
                                border-left: 9px solid #7a3;
                                outline:0 none;
                                background: #fff;
                                box-shadow:0px 0px 20px 2px rgba(255,255,255,0.75);}


input:invalid   {
                                border-left: 10px solid #900;

::input-placeholder, small         {font: 300 11px/14px Open Sans; color: #555;}
::-webkit-input-placeholder, small {font: 300 11px/14px Open Sans; color: #555;}


legend, label, h4   {font: 600 13px/18px Open Sans; color: #333;}


fieldset            {margin-bottom: 13px;
                     border: 1px solid rgba(50,50,40,0.3); /* Die Rahmen haben einen RGBA Wert damit sie immer Dunkler als der Hintergrund-Verlauf sind. */
                     background: transparent;}


form fieldset  {
                    -webkit-column-break-inside: avoid; /* Verhindert Umbruch im Element in Webkit Browsern (Safari, Chrome) */
                     page-break-inside: avoid;          /* Verhindert Umbruch im Element in Webkit Mozilla (Firefox) */
                     break-inside: avoid;               /* Verhindert Umbruch im Element in Internet Explorer IE 10+ */}


optgroup option {color:#333;}


output          {color: #900;}


/* Link auf der Ausgabe Seite nach dem Senden, :hover und :visited gelten auch für die Subline; */

a:link              {font: 600 13px/18px Open Sans; color: #b00; text-decoration: none;}
a:hover             {font: 600 13px/18px Open Sans; color: #7a3; text-decoration: underline;}
a:visited           {font: 600 13px/18px Open Sans; color: #900; text-decoration: none;}


.title              {font: 300 48px/25px Open Sans; color: #555; margin: 30px 0 25px 0;}
.title i            {font: 300 48px/25px Open Sans; color: #700;}
.title strong       {font-weight: 600;}
.home:link          {font: 300 11px/10px Open Sans; text-align: right; margin-right: 49px; color: #333; margin-bottom: 20px; display: block;} /*Subline-Link


/* Output-Bereich im HTML-Formular */

#Output             {height: 250px;}
#Output strong      {color: #000; position: relative; top: -8px; font-size: 20px;}    /* Korrektur der Höhe für das + und das - bei dem Range-Slider */
#Output h4          {padding-left: 7px; border-left: 10px solid #896;}
#Output .minus      {color: #900;}
#Output .plus       {color: #370;}
#Control            {height: 90px;}


/* Gestaltet die Ausgabe auf der Webseite nach dem Senden der Inhalte mit PHP */

#Ausgabe-Anrede      {width: 650px; padding: 10px 10px 20px 10px; border-left: 10px solid #7a3; margin: 2px 0 13px 0;}
#Ausgabe-Anrede h1   {font: 300 32px/40px Open Sans; color: #555; margin: 0px 0 8px 7px;}
#Ausgabe-Anrede h1 i {font: 300 32px/40px Open Sans; color: #700;}


#Ausgabe-Inhalt                         {width: 320px; padding: 10px 10px 20px 10px; background-color: #efefef; margin-left: 30px;}
#Ausgabe-Inhalt h4                      {margin: 2px 0 4px 8px;}
#Ausgabe-Inhalt table                   {border-spacing: 0px; padding: 4px; margin-bottom: 13px;}
#Ausgabe-Inhalt table td:nth-child(odd)  {font: 300 11px/14px Open Sans; color: #555; width: 80px; padding: 3px;}
#Ausgabe-Inhalt table td:nth-child(even) {font: 300 13px/14px Open Sans; color: #000; width: 200px; padding: 3px;}
#Ausgabe-Inhalt table tr:nth-child(odd)  {background: #ddd;}
```

Die Datei style.css enthält auch die Informationen für den später benutzten Erfolgsmeldungs-bildschirmt.

»All input is evil«

Alte Entwickler Weisheit

Prüfung mit HTML

Bei einigen Eingeabefeldern wird mit dem Attribut pattern die Eingeabe definiert. Bei dem E-Mail Feld wird über den Type="email" dies von dem Browser auch erledigt. Dabei ist anzumerken das die Daten so auch falsch gesendet werden würden. Es wird erst einmal nur die pseudoclass invalid auf das Feld aufmerksam.

```
<input type='text' id='hnr' name='hnr' size='3' pattern='\d{1,3}' placeholder='Nr.'><br />
<input type='text' id='plz' name='plz' size='5' pattern='\d{5}' placeholder='PLZ' >
```

```
<p>
    <label for='emailID'>E-mail</label><small>* Pflichtfeld</small><br />
    <input type='email' name='email' id='emailID' maxlength='40' required>
</p>
```

Prüfung mit Javascript

Der Javascript „check.js" prüft ob Daten in der E-Mail angegben sind. Ist dies nicht der Fall, dann Poppt ein Warnhinweis auf. Einige Browser, z.B. Chrome, ignorieren dies und ziehen ihr eigenes Wanrporgramm durch.

```
function validateForm()
{
    var x = document.forms["Form2Mail"]["email"].value;
            if (x == null || x == "") {
                alert("Bitte E-Mail Adresse angeben");
                return false;
    }
}
```

Übergabe der Daten an PHP

Die Datei send.php ist etwas länger. Sie beginnt mit einem HTML Teil.
Dann öffnet mit <?php der eigentliche PHP Abschnitt.

Zu Beginn werden den Daten die wir von dem Formular per POST bekommen haben Variablen zugeordnet. Diese Variablen werden im späteren Verlauf benutzt.

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset='UTF-8'>
        <link rel="stylesheet" href="style.css">
    </head>
    <body>

<!-- In .php Dokumenten ist HTML grundsätzlich möglich.
     Außerhalb des tatsächlichen PHP gelten dann die Regeln des HTML.
     Das sieht man schön an den unterschiedlichen Kommentaren. -->



    <?php
/*
```

```
    VARIABLEN

    II Definition der $Variablen (Platzhalter)

    Von dem Formular sind per POST Daten gesendet.
    Sie sind im Formular mit dem Attribut name="..." gekennzeichnet.
    Zu Beginn werden nun diese Namen mit Variablen in Verbindung gesetzt.
    Die Variablen werden später in der Ausgabe genutzt.
*/


/* Anrede */
$anrede = $_POST['anrede'];
$vorname = $_POST['vorname'];
$nachname = $_POST['nachname'];
$email = $_POST['email'];

/* Anschrift */
$str = $_POST['str'];
$hnr = $_POST['hnr'];
$plz = $_POST['plz'];
$ort = $_POST['ort'];

/* Geburtstag */
$GJahr = $_POST['GJahr'];
$GMonat = $_POST['GMonat'];
$GTag = $_POST['GTag'];

/* Input + Textarea */
$pass = $_POST['pass'];
$nachricht = $_POST['nachricht'];
$AGB1 = $_POST['AGB1'];
$AGB2 = $_POST['AGB2'];
$auswahl = $_POST['auswahl'];
$number = $_POST['number'];

/* Output */
$TeilA = $_POST['TeilA'];
$TeilB = $_POST['TeilB'];
$versand = $_POST['versand'];
$summe = $_POST['summe'];
```

Versenden der Daten per E-Mail mit PHP

Die Variablen werden in Verbindung mit HTML so angeordnet wie die spätere E-Mail aussehen soll. Im $header wird definiert wohin die Daten gehen sollen.

Der PHP Befehl mail ist dann verantworltich dass die E-Mail gesendet wird.

```
113
114
115
116
117
118
119
120
121
122
123     III.1 Aufbau der E-Mail
124
125
126
127     Im folgenden Abschnitt sind der Absender, der Empfänger
128     und die Daten der E-Mail definiert.
129     */
130
131 /* Empfänger */
132 $to  = "$email" . ", " ; // beachtet bitte das Komma
133
134 /* Betreff */
135 $subject = "Mail2PHP code.arnoldbodeschule.de";
136
137 /* Nachricht */
138 $message = '
139 <b> '.$anrede.' '.$vorname.' '.$nachname.'</b> <br />
140 '.$email.'<br />
141 <br /><br />
142 <b>Anschrift</b><br />
143 '.$str.' '.$hnr.'<br />
144 '.$plz.' '.$ort.'<br />
145 <br /><br />
146 <b>Geburtstag</b><br />
147 '.$GTag.'.'.$GMonat.'.'.$GJahr.'<br />
148 <br /><br />
149 <b>Geburtstag</b><br />
150 '.nl2br($nachricht).'<br />
151 '.$pass.' (Passwort)<br />
152 '.$AGB1.' <br />
153 '.$AGB2.' <br />
154 '.$auswahl.' (Auswahlliste) <br />
155 '.$number.' (Anzahl)<br />
156 <br /><br />
157 <b>Output</b><br />
158 '.$TeilA.' (je 59,15 Euro)<br />
159 '.$TeilB.' (je 18,75 Euro)<br />
160 '.$versand.' (Versandkosten) <br />
161 ';
162
163 /*
164
165
166 III.2 Aufbau des Headers der E-Mail
167
168
169
170 Die Variable $Header setzt sich wie folgt zusammen:
171
172 */
173 $headers  = 'MIME-Version: 1.0' . "\r\n";                              /* MIME bedeutet Format of In
174 $headers .= 'Content-type: text/html; utf-8' . "\r\n";         /* Der Type text/html erlaubt HTML Code i
175                                                                        /* charset ist für die Umlaut
176
177 $headers .= 'To: $anrede $vorname <$email>' . "\r\n";                 /* Angabe von dem Empfänger *
178 $headers .= 'From: CODE.arnoldbodeschule Testformular <$email>' . "\r\n";   /* Angabe von dem Absender */
179
```

```
211
212 $gesendet = mail($to, $subject, $message, $headers); /* Verschicken der Mail */
213
```

Verbinden mit der Datenbank

Für dasVerbinden zu der Datenbank bedarf es des Benutzernamen und Passwort. Dies sind in der Regel sehr sensibele Daten. Für die Sicherheit sind diese Informationen in der externen php2sql.php abgelegt. Dies geschieht über den Befehl include.

```
240
241    include 'include/php2sql.php';
242
```

Dies hat den konkreten Vorteil das der Ordner include samt Inhalt gesondert gesichert werden kann. Z.B. kann dort eine .htaccess abgelegt werden die jeglichen Zugriff, außer er ist Serverseitig, verhindert.

Ein vergleichbarer Prozes findet übrigens auch in CMS wie Wordpress statt.

```php
<?php
$servername = "localhost";
$username = "root";
$password = "";
$dbname = "PHP2SQL";

// Create connection
$conn = new mysqli($servername, $username, $password, $dbname);
// Check connection
if ($conn->connect_error) {
    die("Connection failed: " . $conn->connect_error);
}

$sql = "INSERT INTO TablePHP2SQL (anrede, vorname, nachname, email) VALUES ('$anrede', '$vorname', '$nachname', '$email')";

if ($conn->query($sql) === TRUE) {
    echo "<h1>New record created successfully</h1>";
} else {
    echo "Error: " . $sql . "<br>" . $conn->error;
}

$conn->close();
?>
```

Wenn (if) die Datenbank erfolgreich angebunden ist, wird in die Spalten anrede, vorname, nachname, email die Werte (PHP Veriablen) eingefügt und bei Erfolg in der <h1> der Text „New record created sucessfully“ ausgegeben. Abschließend wird die Verbindung beendet.

```
    ECHO

    VI Anzeige der übergebenen Daten

    Es soll sich auch was auf der Webseite tun.
    Der Nutzer braucht direkt auf der Webseite ein Feedback.

    Verzichten wir auf diesen wichtigen Part,
    dann sieht der Nutzer nur irgendwann in seinem Mailfach die Mail.
    Die meisten Nutzer würden denken, da ist was kaputt.

    */

if ($gesendet == true) /* Wenn das Senden geklappt hat, dann ... */
{
    echo "
        <div id='Ausgabe-Anrede'>
            <h1>Hallo <i>$vorname $nachname</i></h1>
            <p>Es hat funktioniert! <br />
                Eine E-Mail mit diesem Inhalt <br />wurde an die Adresse <b>$email</b> gesendet.</p>
                <p><a href='index.html'>Zurück zu Form2Mail</a></p>
        </div>
            <div id='Ausgabe-Inhalt'>
            <h4>Person</h4>
            <table>
                <tr>
                    <td>Anrede</td>
                    <td>$anrede</td>
                </tr>
                <tr>
                    <td>Vorname</td>
                    <td>$vorname</td>
                </tr>
                <tr>
                    <td>Nachname</td>
                    <td>$nachname</td>
                </tr>
                <tr>
                    <td>E-Mail</td>
                    <td>$email</td>
                </tr>
                <tr>
                    <td>Geburtstag</td>
                    <td>$GTag.$GMonat.$GJahr</td>
                </tr>
            </table>
            <h4>Anschrift</h4>
            <table>
                <tr>
                    <td>Straße</td>
                    <td>$str</td>
                </tr>
                <tr>
                    <td>Hausnummer</td>
                    <td>$nnr</td>
                </tr>
                <tr>
                    <td>Ort</td>
                    <td>$ort</td>
                </tr>
            </table>
            <h4>Inputs + Textarea</h4>
            <table>
                <tr>
                    <td>Textarea</td>
                    <td style='font-size: 9px;'>$nachricht</td>
                </tr>
                <tr>
                    <td>Password</td>
                    <td>$pass</td>
                </tr>
                <tr>
                    <td>Checkbox 1</td>
                    <td>$AGB1</td>
                </tr>
                <tr>
                    <td>Checkbox 1</td>
                    <td>$AGB2</td>
                </tr>
                <tr>
                    <td>Auswahlliste</td>
                    <td>$auswahl</td>
                </tr>
                <tr>
                    <td>Anzahl</td>
                    <td>$number</td>
                </tr>
            </table>
            <h4>Output</h4>
            <table>
                <tr>
                    <td>Teil A<br/> je 59,15 €</td>
                    <td>$TeilA</td>
                </tr>
                <tr>
                    <td>Teil B<br/> je 18,75 €</td>
                    <td>$TeilB</td>
                </tr>
                <tr>
                    <td>Versandkosten</td>
                    <td>$versand €</td>
                </tr>
                <!-- <tr>
                    <td>Summe wird nicht übertragen</td>
                    <td>$summe </td>
                    </tr> -->
            </table>
            </div>
          ";
}
else /* Wenn das Senden nicht geklappt hat, dann ... */
{
    echo "<p>Leider konnte Ihre eMail nicht an uns übersendet werden.<br />
            Bitte versuchen Sie es zu einem späteren Zeitpunkt nocheinmal.</p><br /><br />
        ";
}


/* Ende des PHP */
?>

</body>
```

Erfolgsmeldung

Nachdem die Mail gesendet wurde und der Eintrag in die Datenbank abgeschlossen, wird diese Erfolgsmeldung ausgegeben. Dabei wird in einer HTML Tabelle die gesendeten Daten angezeigt.

Zum Abschluss schließt noch mit ?> das PHP und dann mit <body> das HTML.

New record created successfully

Hallo Arnold Bode

Es hat funktioniert!
Eine E-Mail mit diesem Inhalt
wurde an die Adresse **bode@absks.de** gesendet.

Zurück zu Form2Mail

Person

Anrede	Herr
Vorname	Arnold
Nachname	Bode
E-Mail	bode@absks.de
Geburtstag	05.05.

Anschrift

Straße	Schillerstraße
Hausnummer	16
Ort	Kassel

Inputs + Textarea

Textarea	Das ist eine Test-Email
Password	lllllll
Checkbox 1	1x gelesen
Checkbox 1	2x gelesen
Auswahlliste	Ja
Anzahl	10

Überfachliche Kompetenzentwicklung

In der Zusammenarbeit sind nicht nur die fachlichen Kompetenzen wichtig ...

Wir haben mittlerweile mehr als 450 Arbeitgeber, Abteilungsleiter und Ausbilder in Betrieben der Medienbranche und auch anderer Wirtschaftszweige nach einer Einschätzung der relevanten überfachlichen Kompetenzen gebeten.

Auf den folgeden beiden Seiten ist die Papierversion des Fragebogens abgebildet. Es gab natürlich auch eine Version mit HTML Formular und Datenbank. Danach kommt die Auswertung.

Wie wichtig sind also diese Sozial- und Selbstkompetenzen sowie Sprach- und Lernkompetenzen?

Sozial- und Selbst-kompetenzen
Pünktlichkeit
Genauigkeit
Zuverlässigkeit
Verantwortungs-bewusstsein
Flexibilität
Mobilität
Aussehen
Höflichkeit
Freundlichkeit
Teamgeist
Motivation

Sprach- und Lernkompetenzen
Sprechen
Schreiben
Lesen
Fremdsprachen
Informations-beschaffung
Allgemeinbildung
Lernbereitschaft

Wie wichtig sind diese überfachlichen Kompetenzen für die Stelle in Ihrem Unternehmen?

Die erhobenen Daten werden nur anonymisiert ausgewertet und ausschießlich für den Unterricht in der Schule verwendet. Eine Weitergabe der originalen Daten ist ausgeschlossen.
Die Beschreibung der überfachlichen Kompetenzen finden Sie auf der zweiten Seite.

Überfachliche Kompetenzen	unwichtig	teils wichtig	wichtig	sehr wichtig	entscheidend
Sozial- und Selbstkompetenzen					
Pünktlichkeit	☐	☐	☐	☐	☐
Genauigkeit	☐	☐	☐	☐	☐
Zuverlässigkeit	☐	☐	☐	☐	☐
Verantwortungsbewusstsein	☐	☐	☐	☐	☐
Flexibilität	☐	☐	☐	☐	☐
Mobilität	☐	☐	☐	☐	☐
Aussehen	☐	☐	☐	☐	☐
Höflichkeit	☐	☐	☐	☐	☐
Freundlichkeit	☐	☐	☐	☐	☐
Teamgeist	☐	☐	☐	☐	☐
Motivation	☐	☐	☐	☐	☐
Sprach- und Lernkompetenzen					
Sprechen	☐	☐	☐	☐	☐
Schreiben	☐	☐	☐	☐	☐
Lesen	☐	☐	☐	☐	☐
Fremdsprachen	☐	☐	☐	☐	☐
Informationsbeschaffung	☐	☐	☐	☐	☐
Allgemeinbildung	☐	☐	☐	☐	☐
Lernbereitschaft	☐	☐	☐	☐	☐

Stelle

Haupttätigkeit / Funktion

Branche

Unternehmen

Anzahl der Mitarbeiter

Ansprechpartner *

E-Mail *

* freiwillige Angabe

Vielen Dank für die Bewertung einer oder mehrerer Stellen in ihrem Unternehmen.
Mit ihren Angaben tragen Sie zu einem praxisnahen und unternehmensorientierten Unterricht bei.

Freundlicher Gruß
Norman Seeliger

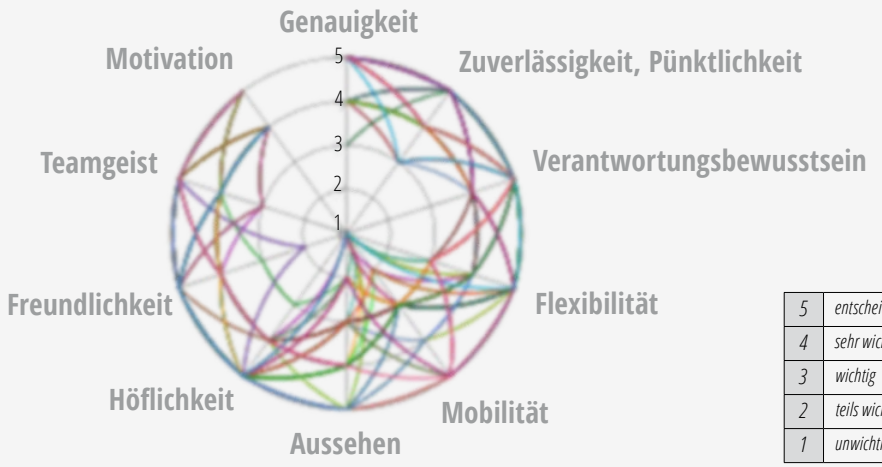
Sozial- und Selbstkompetenzen

Pünktlichkeit	Die Mitarbeiterinnen und Mitarbeiter sind in der Lage ihre Zeit so zu planen, dass sie die verabredeten Termine und Projekte präzise und verlässlich einhalten.
Genauigkeit	Die Arbeitsergebnisse der Mitarbeiterinnen und Mitarbeiter sind durch Sorgfalt, Gründlichkeit, Richtigkeit, Zielgerichtetheit und Übereinstimmung mit den Anforderungen und Absprachen geprägt.
Zuverlässigkeit	Die Mitarbeiterinnen und Mitarbeiter erfüllen regelmäßig und mit hoher Sicherheit die Absprachen.
Verantwortungsbewusstsein	Die Mitarbeiterinnen und Mitarbeiter übernehmen und tragen bewusst die Verantwortung für ihre Aufgabe.
Flexibilität	Die Mitarbeiterinnen und Mitarbeiter sind gegenüber Veränderungen schnell aufgeschlossen, umstellungsfähig und anpassungsbereit.
Mobilität	Die Mitarbeiterinnen und Mitarbeiter sind in der Lage sich an einen anderen Ort, auch im Ausland, zu begeben und dort gegebenfalls länger zu arbeiten. Dafür nutzen Sie verschiedene Verkehrsmittel, so wie ihr eigenes Auto, den Zug oder das Flugzeug.
Aussehen	Die Mitarbeiterinnen und Mitarbeiter passen ihre Kleidung, Frisur und Auftreten den Anforderungen des Unternehmens, ihrer Funktion und ihren Aufgaben an.
Höflichkeit	Das Verhalten der Mitarbeiterinnen und Mitarbeiter ist gegenüber Kunden, Kollegen und Vorgesetzten angemessen, rücksichtsvoll und respektvoll.
Freundlichkeit	Das Verhalten der Mitarbeiterinnen und Mitarbeiter gegenüber Kunden, Kollegen und Vorgesetzten ist durch liebenswürdiges Verhalten und wohlwollenden Interesse am Gegenüber geprägt.
Teamgeist	Die Mitarbeiterinnen und Mitarbeiter arbeiten partnerschaftlich und mit Interesse innerhalb eines Arbeitsteams zusammen.
Motivation	Die Mitarbeiterinnen und Mitarbeiter haben eigenen starken Antrieb und zeigen großes Interesse für ihre Aufgabe.

Sprach- und Lernkompetenzen

Sprechen	Die Mitarbeiterinnen und Mitarbeiter führen Gespräche im adäquaten Stil. Sie können sich spontan und fließend ausdrücken, z.B. ohne öfter deutlich erkennbar nach Wörtern suchen zu müssen.
Schreiben	Die Mitarbeiterinnen und Mitarbeiter schreiben korrekte Texte hinsichtlich Inhalte, Stil, Grammatik und Rechtschreibung, z.B. Korrespondenz oder Arbeitsablaufpläne.
Lesen	Die Mitarbeiterinnen und Mitarbeiter verstehen geschriebene Texte unterschiedlicher Art, z.B. technische Anleitungen oder Arbeitspläne.
Fremdsprachen	Die Mitarbeiterinnen und Mitarbeiter beherrschen eine oder mehrere Fremdsprachen. Sie verstehen die Information in Sprache und Text und können sich spontan und fließend ausdrücken.
Informationsbeschaffung	Die Mitarbeiterinnen und Mitarbeiter können aus relevanten Quellen die benötigten Informationen recherchieren, zusammenfassen und dabei Begründungen und Erklärungen in einer zusammenhängenden Darstellung wiedergeben.
Allgemeinbildung	Die Mitarbeiterinnen und Mitarbeiter haben eine umfassende allseitige Bildung. Sie interessieren sich für Weltnachrichten.
Lernbereitschaft	Die Mitarbeiterinnen und Mitarbeiter sind bereit, Neues für ihren Beruf und Arbeitsaufgabe zu lernen, auch in ihrer Freizeit.

Sozial- und Selbstkompetenzen



Bemerkenswert ist die sehr hohe Einschätzung bei Motivation und Verantwortungsbewusstsein. Teamgeist, Zuverlässigkeit, Pünktlichkeit und Genauigkeit sind ebenso grundsätzlich wichtig und nirgendwo unwichtig.

5	entscheidend
4	sehr wichtig
3	wichtig
2	teils wichtig
1	unwichtig

Die Bewertung der anderen Kompetenzen ist sehr von der Stelle im Betrieb abhängig.

a) Welche der Sozial- und Selbstkompetenzen sind deiner Meinung nach für die Schule wichtig?

b) Welche der Sozial- und Selbstkompetenzen sind deiner Meinung nach für deinen Traumjob wichtig?

c) Wann bist Du motiviert und was kannst Du dafür tun?

d) Nenne ein Beispiel für eine Arbeitsstelle bei der Freundlichkeit und Höflichkeit entscheident ist.

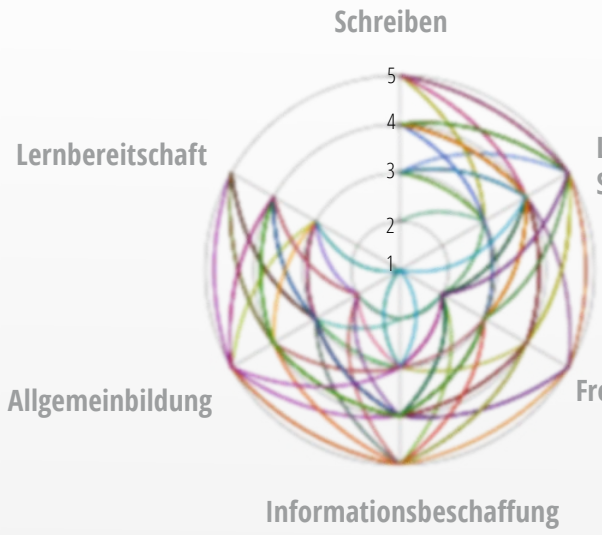
e) Nenne ein Beispiel für eine Arbeitstelle bei der Teamgeist unwichtig ist.

f) Nenne ein Beispiel für eine Arbeitstelle bei der Mobilität unwichtig ist.

g) Nenne ein Beispiel für eine Arbeitstelle bei der Flexibilität entscheidend ist.

h) Nenne ein Beispiel für eine Arbeitstelle bei der Aussehen und Kleidung sehr wichtig ist.

Sprach- und Lernkompetenzen



5	entscheidend
4	sehr wichtig
3	wichtig
2	teils wichtig
1	unwichtig

Im Gegensatz zu den Sozial- und Selbstkompetenzen ist bei den Sprach- und Lernkompetenzen kein eindeutiger Trend ablesbar. Die Bedeutung ist von der jeweiligen Stelle im Betrieb abhängig.

Relativ gesehen spielen bei den Sprach- und Lernkompetenzen Schreiben, Lesen, Sprechen, Lernbereitschaft, die Bereitschaft und Fähigkeit zur Informationsbeschaffung gegenüber Fremdsprachen und Allgemeinbildung eine größere Rolle.

Welche der Sprach- und Lernkompetenzen sind deiner Meinung nach für die Schule wichtig?

Welche der Sprach- und Lernkompetenzen sind deiner Meinung nach für deinen Traumjob wichtig?

Was kannst Du für die Entwicklung deiner Fähigkeit zur Informationsbeschaffung konkret tun?

Anregungen für dein Lernen

Ich wünsche Dir viel Freude und interessante Erfahrungen bei deinem Weg. Du kannst frei gestalten, wie Du dabei vorgehst. Deinen Ordner zu pflegen, macht sicherlich großen Sinn, vielleicht brauchst Du dies auch nicht. Viele Lernenden machen gute Erfahrungen mit dem zusammenfassenden Beschreiben von Einlegern, Markern und Trennkarten. Auch der Abgleich mit Anderen und das Zusammentragen einer gemeinsamen Musterlösung ist eine gute Ergänzung. Wenn Du Interessantes aus anderen Quellen findest, dann hefte diese Ergänzungen in deinem Ordner an der entsprechenden Stelle ein. Letzendlich findest kannst Du viel experimentieren, um eine für dich stimmige Lösung zum Lernen zu finden.

Was hat Dir in der Vergangenheit häufig beim Lernen geholfen?

Was hat in der Vergangenheit häufig dein Lernen eher verhindert?

Hast du praktische Erfahrung im Design?

Hast Du bereits einen Bezug zu Webdesign?

Intention

Intention des Unterrichts für Webdesign ist nicht, dass Du den Inhalt dieses Ordners auswendig lernst. Vielmehr soll der Ordner Dir dabei helfen, Jobs wie den für den Fotografen zuverlässig umsetzen zu können.

Wirklich ohne externe Quelle arbeiten zu können, dass gilt nur für bestimmte Elemente kurz und während deiner Prüfung. Bis dahin ist der Ordner, auch in Klassenarbeiten, dein griffbereites und persönliches Nachschlagewerk. Die Fähigkeit, unbekannte Inhalte zu recherchieren, ist wesentlich relevanter für beruflichen Erfolg, als wenige Elemente auswendig zu lernen. Bringe bitte deine Unterlagen zu jedem Unterricht mit.

Wann ist deine Prüfung?

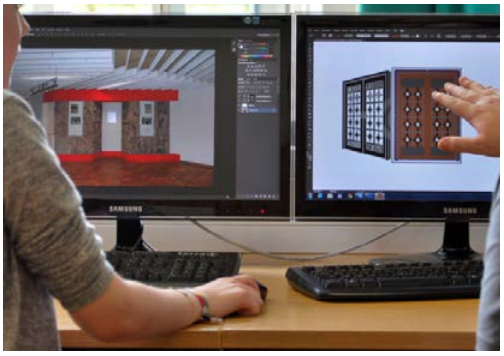
Wann ist deiner Meinung nach eine Klassenarbeit fair?

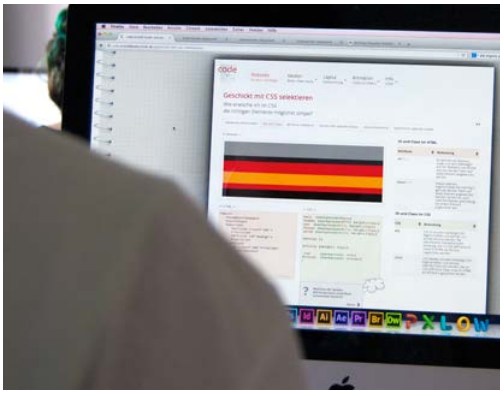
Angenommen die Lösung für ein Problem steht nicht in diesem Ordner, welche Möglichkeiten siehst Du, um unbekannte Inhalte zu erschließen?

Gestaltung und Medien an der Arnold-Bode-Schule

Neubau der Arnold-Bode-Schule.
Zugegeben ... wir haben auch einen Altbau. :)

Die Arnold-Bode-Schule bietet im Bereich der Medien und Gestaltung viele Bildungsangebote. Eine Übersicht mit Ansprechpartner ist auf der nächsten Seite. Besuche, auch in den regulären Unterricht, sind an unserer Schule grundsätzlich möglich. Wir freuen uns auf deine Fragen und bitten bei Besuchen um ein wenig Vorlaufzeit.





Verschiedene Arbeiten unserer Schülerinnen und Schüler: Skulpturen in der Fachoberschule für Kunst und Gestaltung, Fotografie oder Mediengestaltung für Digital und Druck.

Kontakt

- Fragen rund um CODE
  „concept for open digital education“
  können an **seeliger@absks.de** gesendet werden.

code.arnoldbodeschule.de
Aktuelle Informationen zu CODE.

www.arnoldbodeschule.de
Aktuelle Informationen zu den Bildungsangeboten und dem Leben an unserer Schule finden sich auf unserer Webseite.

Bildungsangebot	Inhalte und Dauer	Abschluss	Vorraussetzungen[1]
Berufliches Gymnasium für Gestaltungs- und Medientechnik Silvia Zicklam-Werner **zicklam-werner@absks.de**	Kommunikations-Design, Audio, Video, Produkt-Design, Interface-Design, Design-Prozesse, Medienprodukten, DTP-Grundlagen, Bildbearbeitung, Print- und Non-Print-Medien, Marketing- und PR-Maßnahmen **36 Monate in Vollzeit**	Allgemeine Hochschulreife[2]	Versetzung in die gymnasiale Oberstufe. Bei mittleren Bildungsabschluss: Besser als Note 3 in Deutsch, Mathematik, Englisch und einer Naturwissenschaft.
Gestaltungs- und Medientechnischer Assistent (GMTA) Ute Kampmann **kampmann@absks.de**	Audiovisuelle Medien und Animationen, Medienprodukte, Statische Webseite, Mathematische Denk- und Arbeitsweisen, Computerarbeitsplätze, Englisch, Printprodukte, Betriebspraktikum, Kalkulation, Dynamische Webseiten, Rechnernetzwerke, Marktforschung und Marketing, Projektmanagment **24 Monate in Vollzeit**	Staatlich geprüfter Gestaltungs- und Medientechnischer Assistent Fachhochschulreife[3]	Abschluss Sekundarstufe I Mindestens zwei Mal die Note 3 oder besser in Deutsch, Mathematik und Englisch. Höchstens eine Note 4. Auswahlverfahren aufgrund von Bewerbungsgespräch und Gestaltungs-Mappe
Fachoberschule für Gestaltung (FOS Gestaltung) Silvia Zicklam-Werner **zicklam-werner@absks.de**	Zweidimensionale Gestaltung, Form, Farbe und Material, Objekt und Raum, Plastik, Skulptur, Relief, Gestaltung von Lebensräumen, physiologisch Wirkung von Farben und Werkstoffen, Stilepochen, Designgeschichte, Designprozess und Designprodukte, Medienrecht, Konstruktives Zeichnen, 800 Stunden spezifisches Praktikum **24 Monate in Vollzeit**	Fachhochschulreife	Abschluss Sekundarstufe I Mindestens zwei Mal die Note 3 oder besser in Deutsch, Mathematik und Englisch. Höchstens eine Note 4. Fachspezifischer Eignungstest
Mediengestalter Digital und Print Ute Kampmann **kampmann@absks.de**	Medienbetrieb, Typografie, Ausgabedateien, Druckverfahren, Computerarbeitsplatz, Netzwerke, Statische Webseite, Bildbearbeitung, Ausgabeprozesse, Logo, Konzeption, Kalkulation, Medienprojekte, Color-Management, Dynamische Webseiten, Digitalmedien **36 Monate in Teilzeit**	Gepr. Mediengestalter/in IHK	Ausbildungsplatz in einem Betrieb für die Duale Ausbildung
Fotografen Anja Birkenfeld **birkenfeld@absks.de**	Medienbetrieb, Aufnahmesysteme, Licht, Bildbearbeitung, Aufnahmentwürfe, Kamerasysteme, Motive, Aufträge, Konzeptionen, Bilder bewerten, Bildsprache, Bilddaten, Medienprodukte, Projekte **36 Monate in Teilzeit**	Gepr. Fotograf/in Handwerk	Ausbildungsplatz in einem Betrieb für die Duale Ausbildung
Medientechnologen Christian Heitland **heitland@absks.de**	Druckbetrieb, Arbeitsabläufe, Daten, Werkstoffe, Druckprodukte, Produktionsmaterialien, Druckmaschinen, Mess- und Prufverfahren, Prozessstandards, Personalisierte Digitaldrucke, Bogendruckerzeugnisse, Rollendrucksysteme, Digitale Drucksysteme, Planen **36 Monate in Teilzeit**	Gepr. Medientechnologe/in IHK	Ausbildungsplatz in einem Betrieb für die Duale Ausbildung
Geomatiker Steffen Kreker **kreker@absks.de**	Geoinformationstechnologie, Geodaten, Datenbanken, Geodaten gestalten, Geobasisdaten, Fernerkundungsdaten, Multimedia-Geodaten, Printbezogene-Geodaten, Mehrdimensionale Geoprodukte, Geoprodukte **36 Monate in Teilzeit**	Gepr. Geomatiker/in IHK	Ausbildungsplatz in einem Betrieb für die Duale Ausbildung
Abteilungsleiterin	Petra Jany **jany@absks.de**		

Anmerkungen
1 Bei Fragen zu den Vorraussetzungen, Aufnahmegesprächen, Aufnahmeprüfung und Anmeldung helfen wir gerne weiter.
2 Berufliches Gymnasium: Kooperation zwischen der Max-Eyth-Schule Kassel und der Arnold Bode Schule Kassel.
3 GMTA: Fachhochschulreife kann durch Zusatzunterricht und -prüfung in Deutsch, Englisch, Mathmatik sowie einem Zusatz-Praktikum erworben werden.

Bildnachweis