Einführung

in

HTML

Version 2019.6

Autor: Dieter Lindenberg

Inhalt

Struktur

Die **HyperText Markup Language** (**HTML**, dt. *Hypertext-Auszeichnungs-sprache*), dient zur Strukturierung von Texten, Bildern und sog. Hyperlinks in Dokumenten. HTML-Dokumente sind die Grundlage des Internets und werden von einem Internetbrowser dargestellt. Leider interpretieren die zur Zeit (2018) gebräuchlichen Internet-Browser die HTML-Befehle nicht immer einheitlich.

Die normale HTML-Sprache ist ziemlich tolerant gegenüber Groß- oder Kleinschreibung von Anweisungen. In den meisten Computersprachen wird jedoch Wert darauf gelegt. Auch in Weiterentwicklungen von HTML, z.B. in XHTML (e**X**tended **H**yper **T**ext **M**arkup **L**anguage) wird darauf geachtet. Wir sollten deshalb alle Anweisungen klein schreiben.

```
<html>
   <head>
      <title>Meine erste Homepage</title>
   </head>
   <body>
      Dies ist meine erste<br />
      HTML-Seite !
   </body>
</html>
```

Jedes Dokument im HTML-Format hat am Anfang die Startanweisung **<html>** und am Schluss die Endanweisung **</html>** . Der Schrägstrich wird auch als *Slash* bezeichnet. Alle HTML-Befehle werden in spitzen Klammern geschrieben und als *Tag* (englisch: Etikett, Aufhänger) bezeichnet.

Dazwischen besteht das Dokument aus zwei Teilen: dem Kopfteil (*Header*), eingeleitet durch **<head>** und beendet durch **</head>** und dem Datenteil (*BODY*), eingeleitet durch **<body>** und beendet durch **</body>** .

Im Header steht üblicherweise u.a. eine Titelangabe für diese HTML-Seite. Man sollte hier möglichst immer einen aussagekräftigen Namen für die Seite verwenden, da viele Suchmaschinen diesen als Suchkriterium benutzen.

Außerdem könnten im Header noch Informationen über den Inhalt der HTML-Seite stehen, sodass Internet-Suchmaschinen schneller diese Seite für ihre Kataloge einordnen können. Auch Anweisungen für Darstellungen von Schriften und Farben (Stylesheets) findet man hier öfter.

Aufgaben

1. Schreibe obige HTML-Beispielseite mit einem normalen Texteditor (z.B. Windows-Editor, Wordpad oder Notepad++) und speichere sie unter dem Namen *Testseite.htm* oder *Testseite.html* .
Öffne anschließend die Seite im Inhaltsverzeichnis deines Rechners mit einem Doppelklick. Dadurch wird sie automatisch mit dem Standard-Browser geöffnet.

2. Die gängigen Textverarbeitungsprogramme können ihre Seiten auch im HTML-Format speichern. Zum Beispiel bietet *Microsoft Office Word* die Möglichkeiten, eine Seite als *Webseite in einer Datei*, als *Webseite* oder als *gefilterte Webseite* zu speichern.
Schreibe nun mit deinem Textverarbeitungsprogramm nur folgenden Zweizeiler: Dies ist meine erste
HTML-Seite !

Speichere anschließend mit deinem Textverarbeitungsprogramm diese Probeseite als HTML-Seite! Öffne sie danach mit einem normalen Texteditor! Erkennst du, warum es sich lohnt, HTML zu erlernen?

Formatierungen

Attribute

Beispiel: ** **

Vom Prinzip her kann jeder HTML-Befehl mit Attributen erweitert werden! Manche Attribute wie z.B. „id“ und „class“ können bei jedem HTML-Tag genutzt werden. Andere Attribute dagegen nur bei passenden HTML-Tags. Eine Attributangabe besteht immer aus 2 Teilen. Zuerst kommt der Attributname, danach der Attributwert. Nach dem Attributnamen kommt ein Gleichheitszeichen und der Attributwert wird in Anführungszeichen angegeben. Es kann mehrere Attributangaben für einen HTML-Tag geben. Diese Angaben stehen einfach mit einem Leerzeichen getrennt hintereinander. Die Reihenfolge der Attributangaben ist unwichtig.

Bereiche

Normale Zeilenumbrüche werden durch **
** bewirkt. Dieser *Tag* ist ein sog. *standalone-Tag*. Deshalb ist der Schrägstrich auch am Ende des *Tags*. Beachte außerdem die Leertaste zwischen dem Buchstaben r und dem Schrägstrich!

Einen Absatz (engl. *Paragraph*) markiert man nur dann durch ** ** und ** **, wenn man ihm zusätzlich besondere Eigenschaften (Schriftart, Farbe, Einrücken, Zeichengröße usw.) zuordnen will. Ansonsten, wenn er sich nur optisch durch Leerzeilen vom Rest des Dokumentes abheben soll, so genügen eben ein oder mehrere **
**. Dieser Absatz-*Tag* fügt außerdem noch etwas leeren Raum zwischen der letzten Zeile und der neuen ein.
Normalerweise werden Textabsätze linksbündig ausgerichtet. Aber es gibt auch andere Möglichkeiten:

```
<p align="left">
<p align="right">
<p align="center">
<p align="justify">                <!--Blocksatz -->
```

Die zu den *Tags* gehörenden **Werte** sollte man in Anführungsstrichen schreiben.

p-Blöcke (also Absätze, die zwischen und stehen) dürfen keine weiteren p-Blöcke enthalten.

Mit den Anweisungen ** ** und ** ** kann man einen Bereich *(englisch: division)* definieren. Alles zwischen diesen beiden Anweisungen Liegende (Text, Grafiken, Tabellen usw.) wird als eine Einheit betrachtet. Dieser Bereich bewirkt zunächst nichts weiter, als dass er in einer neuen Zeile des Fließtextes beginnt. Zur Erinnerung: Beim p-Element wird ein sichtbarer Abstand erzeugt, beim div-Element nicht.

Ein div-Block hat zunächst keine besonderen Eigenschaften (=Attribute). Er ist dazu gedacht, um mit Hilfe von weiteren Anweisungen Eigenschaften (z.B. Breite, Farbe, Positionierung) des Bereiches festzulegen:

```
<div style="font-size: 100">
<!-- Schriftgröße wird bestimmt  -->
.....
</div>
```

Oft wird dieser Befehl auch genutzt, um zum Beispiel die Ausrichtung von Text, Bildern oder Tabellen festzulegen:

```
<div align="right">
.....
</div>
```

div-Bereiche dürfen weitere div-Bereiche beinhalten; div-Blöcke können also geschachtelt sein:

```
<body>
    Im Folgenden steht rechts die Adresse. <br />
    <div align="right">
        Goethe-Gymnasium <br />
        Stettiner Str. 12
        <div style="font-size: 70">
            Elite-Schule<br />
            der Stadt Dortmund
        </div>
        Telefon: 0231/ 286 736 30
    </div>
    Hier folgt wieder der normale Text.
</body>
```

Mit den Anweisungen **** und **** legt man einen sog. *Inline-Bereich* fest. Ein Inline-Bereich erzeugt keinen neuen Absatz. Die *span*-Anweisungen werden auch nur zwecks Formatierung genutzt:

```
<h1 style="font-size:500%">
<span style="color:blue">A</span>
<span style="color:red">B</span>
<span style="color:green">C</span>
</h1>
```

***span*-Bereiche können ineinander geschachtelt werden:**

```
<body>
    Nun folgen Informationen ueber unsere Schule:
    <span style="color:red">Unsere Schule <span style="font-size:200%">
    ist schoen</span> und die Lehrer sind nett.</span> Ende der
    Informationen.
</body>
```

Am obigen Beispiel erkennt man, dass der Browser Schwierigkeiten hat, deutsche Umlaute richtig darzustellen. Um dieses Problem zu beseitigen, sollte man **im Head-Teil** des HTML-Textes folgende Anweisung unterbringen:
<meta charset="utf-8" />
Mithilfe dieser Anweisung wird der europäische Zeichensatz definiert. Durch die Angabe kommen die Browser und Suchmaschinen auch mit deutschen Umlauten zurecht.

Schrift

Die gewünschte Schriftart wird folgendermaßen eingestellt:
** .....**
Vorausgesetzt wird natürlich, dass die hier geforderte Schriftart überhaupt vom Betriebssystem des Rechners zur Verfügung gestellt werden kann.

Gleichzeitig lässt sich auch die Schriftgröße einstellen:
** .....**
Es gibt 7 verschiedene Schriftgrößen, die mit den Zahlen 1 bis 7 gekennzeichnet werden. 1 ist die kleinste und 7 die größte Schrift, also anders als bei den Überschriften.
Wichtig: mehrere Attribute werden einfach hintereinander geschrieben, Die Reihenfolge ist dabei egal. Als Trennzeichen fungiert ein Leerzeichen.

Man kann die Schriftgröße auch relativ zur Größe der Ausgangsschrift ändern, indem man die Differenz angibt:
** .....**
** .....**
Diese relativen Angaben beziehen sich immer auf die Größe der Ausgangsschrift.

Man kann 6 verschieden große Überschriftsebenen auswählen durch **<hn>** *dies ist eine Überschrift* **</hn>** wobei n eine Zahl zwischen 1 (groß) und 6 (klein) sein kann. Der Buchstabe *h* steht dabei für das englische Wort *Heading*. Jede Überschrift ist ein eigener Absatz. Ein zusätzliches p-*Tag* vorher und nachher ist also überflüssig.

Es gibt die Möglichkeit, die Schriftgröße um zwei Punkte zu vergrößern oder zu verkleinern. Dazu benutzt man die Tags **<big>** und **</big>** bzw. **<small>** und **</small>**.

Möchte man einen im Editor geschriebenen Text mit allen Leerzeichen und genauen Abständen übernehmen, so muss man diesen Text zwischen **<pre>** und **</pre>** schreiben. *pre* ist die Abkürzung von *preformatted*. Allerdings wird der eingeschlossene Text in der Schriftart *Courier* ausgegeben. In dieser Schriftart haben alle Zeichen -auch das Leerzeichen- dieselbe Breite.

Indizes: Beispiel: H**_{**2**}**O liefert H_2O
Potenzen: Beispiel: 10**^{**3**}** liefert 10^3
Unterstrichen wird der Text durch **<u>** und **</u>** *(Underlined)*
Durchgestrichenes steht zwischen **<strike>** und **</strike>** oder, was anscheinend dasselbe bewirkt, zwischen **~~** und **~~**
Fettschrift steht zwischen **** und **** . *(engl. Fett = bold)*
oder zwischen **** und **** . *(ebenfalls fett gedruckt, die Darstellung dieser Fettschrift lässt sich später mit CSS noch genauer festlegen)*
Kursivschrift wird durch **** und **** ermöglicht *(Italics)*
Der Buchstabe i steht für „italic“ = italienisch. Ein italienischer Buchdrucker (Aldus Manutius, 1449 – 1515) entwickelte als erster schräge Buchstaben, um angeblich mehr Text auf eine Seite zu bekommen.
Auch der Tag **** bzw. **** produziert eine schräge Schrift. *Dabei ist em die Abkürzung für das englische Wort „emphasis“ = „Betonung, Gewichtung, Nachdruck“.*

Eine waagrechte Trennlinie wird durch den *stand-alone-Befehl* **<hr />** erzeugt *(horizontal row)*. Hier können aber auch noch einige Attribute festgelegt werden. Beispiel:

```
<hr size="12" width="60%" align="left"
noshade="noshade" color="red">
```

Obige Linie ist 12 Pixel breit, linksbündig und nimmt 60% des Bildschirms ein. Das erste Wort *noshade* in obigem *Tag* bedeutet eine Eigenschaft (Attribut), das zweite *noshade* ist der Wert dieser Eigenschaft.

```
<hr width="80">
```

Obige Linie ist 80 Pixel breit und (voreingestellt) zentriert.

Aufgabe: Schreibe folgenden Text im HTML-Format:

> Mithilfe der sog. ***pq-Formel*** folgt
> aus der Gleichung $x^2 + 2x - 3 = 0$
> relativ schnell, dass
> $\mathbf{x_1 = 1}$ und $\mathbf{x_2 = -3}$ ist.

Kommentare

Man kann im HTML-Quelltext sog. Kommentare einfügen. Diese sind dazu gedacht, dem Leser des Quelltextes das Verständnis zu erleichtern. Kommentare werden von Web-Browsern einfach ignoriert. Ein Kommentar wird eingeleitet mit dem Präfix **<!--** Dahinter folgt beliebig langer Kommentartext. Beendet wird ein **(einzeiliger !)** Kommentar erst mit dem Suffix **-->**.

Beispiel:
```
.......
<!-- im Folgenden zeichne ich einen Strich -->
<hr   align="left" width="60%">
..........
```

Mehrzeilige Kommentare müssen mit dem Suffix **//-->** beendet werden.
Beispiel:
```
.........
<!--    blah, blah
blah, blah, blah
//-->
.........
```

Sonderzeichen

Einige Zeichen haben in HTML eine besondere Bedeutung oder sie sind auf einer normalen nicht vorhanden. Deshalb können sie als Zeichen nicht direkt, sondern nur durch Hilfskonstruktionen sichtbar gemacht werden. Beachte jeweils das Semikolon!

Zeichen	<	>	"	©	‰	≈
Kodierung	<	>	"	©	‰	≈

Zeichen	☎	☏	±	⇒	⇔	π
Kodierung	☎	☏	±	⇒	⇔	π

Zeichen	α	β	γ	☺	∞	♥
Kodierung	α	β	γ	☺	∞	♥

Außerdem haben spezielle deutsche Zeichen eine eigene Kodierung:

Zeichen	ä	Ä	ö	Ö	ü	Ü	ß
Kodierung	ä	Ä	ö	Ö	ü	Ü	ß

uml bedeutet Umlaut.

Mehrere Leerzeichen hintereinander werden vom Browser ignoriert und zu einem einzigen zusammengefasst. Um mehrere Leerzeichen zu erzwingen, verwende anstelle der normalen Leerzeicheneingabe die Zeichenfolge   (das entspricht einem erzwungenen Leerzeichen), und zwar so oft hintereinander wie gewünscht.

Geschützte Leerzeichen (*non breaking space*): **` `** Damit verhindert man, dass z.B. der Text 1. Mai bei einem Zeilenumbruch auseinandergerissen wird.

Farben

Möchte man einen Text farbig schreiben, so gebe man folgenden Befehl ein:
**** *Jetzt kommt der farbige Text* ****
Dabei stehen die Variablen R, G und B für zweistellige Hexadezimalzahlen. Man muss also für R eine Zahl zwischen 00 und FF einsetzen. R entspricht der Intensität von rot, G grün und B blau.
Beispiel: Dies erscheint **** grün ****.

Um die Farben einzustellen, kann man auch statt der Hexadezimalwerte insgesamt 140 gebräuchliche, englische Namen wählen, z.B. color="green" . Die 16 Grundfarben sind:
black, silver, gray, white, maroon, red, purple, fuchsia, green, lime, olive, yellow, navy, blue, teal, aqua

Die Hintergrundfarbe einer HTML-Seite und die Textfarbe bestimmt man noch innerhalb des *body-Tags*: **<body bgcolor="yellow" text="blue">**

Aufgabe: Zeichne auf einer HTML-Seite in großer Schrift, fett, zentriert und farbig das sog. Pascalsche Dreieck:

```
          1
        1   1
      1   2   1
    1   3   3   1
  1   4   6   4   1
```

Zeichne zusätzlich eine weitere Zeile des Pascalschen Dreiecks in einer Farbe deiner Wahl!

Inline-Style

Sog. *Inline-Styles* werden gerade dann gern verwendet, wenn es sich um „einmalige“ Gestaltungsaktionen z.B. für bestimmte Bereiche (p, div, span) handelt. Das *style*-Attribut wird innerhalb eines öffnenden *Tags* geschrieben. Alle Formatierungen mit *style* gelten für den kompletten *Tag*-Bereich. Style-Formatierungen werden immer nach dem Muster **style=“Attribut: Wert“** eingegeben. Mehrere Attribute werden durch ein Semikolon getrennt. Hinter dem letzten Attribut braucht kein Semikolon zu stehen.

```
<p style="font-family: Arial"> Schriftart…… </p>
<p style="font-size: 20pt"> Schriftgröße…… </p>
<p style="font-weight: bold"> …… </p>
<p style="font-style: italic"> …… </p>

<p style="color: blue"> Schriftfarbe…… </p>
<p style="background-color: green"> Hintergrundfarbe </p>
<p style="background: green"> auch Hintergrundfarbe… </p>

<p style="border-style:solid; border-width:10px">
Rahmen </p>
<p style="border-style:double; text-align: center">
…… </p>
<p style="border-style:double; border-color:red"> </p>
<p style="border-style:dotted">punktierte Umrahmung</p>
<p style="border-style:dashed">mit Strichen umrahmt</p>
<p style="margin-left:50pt; color:blue"> linker Rand </p>
<p style="margin-top:150px"> Rand bis zum oberen Objekt </p>
<p style="margin-bottom:2.5cm">  Rand bis zum unteren Objekt
……</p>
<p style="margin:250">gesamter Rand rundherum…….</p>
```

Bemerkung: **Wenn bei den *margin*-Angaben ein Wert mit Einheit angegeben wird, so darf zwischen der Zahl und der Einheit kein Leerzeichen stehen!**
Man kann dem gesamten Dokument einen entsprechend breiten Rand verpassen:
`<body style ="margin-left: 4.3cm">`

```
<p style="text-decoration:overline">überstrichener Text </p>
<p style="text-decoration:underline">unterstrichen </p>
<p style="text-decoration:line-through">durchgestrichen </p>
```

Man kann einem Abschnitt eine exakte Position zuweisen. Dies macht Sinn, wenn der Abschnitt beispielsweise nur ein Bild enthält. Aber es funktioniert auch mit jedem anderen Inhalt. Bei der absoluten Positionsangabe ist mit *top* und *left* der Abstand vom oberen bzw. linken Seitenrand gemeint:

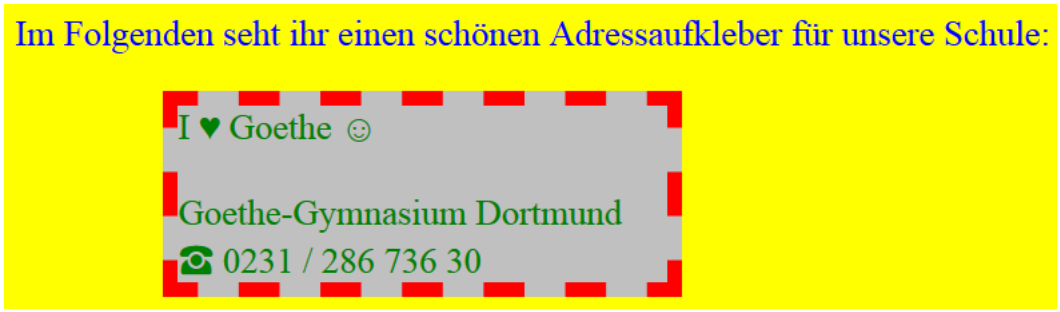
```
<p style="position: absolute; top:10px; left:20px">
Dies ist ein exakt absolut positionierter Text
</p>
```

Dabei besteht natürlich die Gefahr, dass sich an der angegebenen Stelle bereits ein anderer Text oder ein anderes Bild befindet.

Diese Gefahr besteht bei der relativen Positionsangabe normalerweise nicht. Hier ist mit *top* und *left* der Abstand vom vorherigen Objekt gemeint:

```
<p style="position: relative; top:10px; left:20px">
Dies ist ein exakt relativ positionierter Text
</p>
```

Aufgabe: Erstelle genauestens nachfolgenden Teil einer Webseite! Beachte die Farben! Der Adressaufkleber besitzt vom linken Rand einen kleineren Abstand als vom rechten Rand. Unter dem Adressaufkleber ist noch eine gelbe Zeile. Die gesamte Web-Seite ist nicht ganz so breit wie der Bildschirm.



Laufschrift

Mit der Anweisung **<marquee>** lässt sich eine Laufschrift erzeugen.
Beispiel:

```
<marquee behavior="alternate" direction="right"
width="300" scrollamount="30" scrolldelay="1000">
Informatik ist sch&ouml;n </marquee>
```

Attribut	Wert	Erklärung
loop	Ziffer	Anzahl Wiederholungen. Danach ist der Text gelöscht
behavior	alternate	Hin und her
	slide	Reinrollen und stehen bleiben
direction	right	Voreinstellung ist „left“
	down	
	up	
bgcolor	Farbname	
height	Pixel oder %	Platz für die Richtungen up/down
width	Pixel oder %	Platz für die Richtungen right/left
scrollamount	Zahl	Gibt an, um wie viel Pixel das Objekt bei jedem Schritt bewegt wird
scrolldelay	Zahl in Millisekunden	Dauer der Pause zwischen zwei Schritten

Aufgabe:
Erstelle eine sehr große Laufschrift mit blauem Hintergrund und roter Schrift, welche sich langsam, zentriert, horizontal hin und her bewegt.
Gleichzeitig soll sich eine zweite, große Laufschrift schneller, ebenfalls zentriert, vertikal hin und her bewegen.

Hyperlinks

Mit Verweisen – auch Hyperlinks genannt – kann man zu anderen Textstellen springen, üblicherweise entweder zu weiter unten stehenden Stellen im selben Dokument oder aber auch zu ganz anderen HTML-Seiten hin.

Das **Ziel** (auch *Anker* genannt) eines Links wird an der Zielstelle markiert durch **** Überschrift Kapitel Nr.2 **** .
Der *Anker* muss also einen Namen bekommen (hier: "Ziel-1"). Dieser **Zielname**, zu dem hingesprungen werden kann, darf keine Leerzeichen und keine deutschen Umlaute enthalten und als Sonderzeichen höchstens den Unterstrich. Im obigen Beispiel könnte man also zum Anfang des Kapitels 2 springen. Der Zielname selbst ist für den Leser der Webseite nicht sichtbar.
Der *Anker* muss nicht unbedingt einen Text einschließen. Es wäre also durchaus folgendes möglich und manchmal auch sinnvoll:
**** .

Falls man zu fremden Webseiten hin springen will, kann man das Ziel natürlich nicht als Anker markieren.

Eine alternative Möglichkeit, Sprungziele zu definieren, ist die Benutzung des Attributes ***id*=*"Zielname"*** innerhalb eines beliebigen HTML-Befehls.
Beispiele:

```
<h1 id="Ziel-1">...</h1>
<p id="Ziel2">...</p>
<h2 id="Kapitel-09">...</h2>
```

Insbesondere lässt sich auch im body-Tag ein Sprungziel definieren:

```
<body id="Anfang">
```

Soll nun in einem HTML-Dokument auf ein Ziel verwiesen werden, welches sich ebenfalls innerhalb dieses Dokumentes befindet, so reicht an der entsprechenden Absprungstelle (die als *Link* bezeichnet wird) die Anweisung
**** hier steht der sog. *HOTTEXT* ****
Das Doppelkreuz # wird nur für sog. interne Ziele benutzt. Damit bezeichnet man *Anker* **innerhalb** derselben Seite, auf der auch der zugehörige *Link* steht.
Das Schlüsselwort *href* bedeutet *hyper reference*.

Ein *Anker* (also das Sprungziel) kann gleichzeitig auch wieder ein *Link* sein (also Absprungstelle für einen neuen Sprung). Dafür müsste man die

entsprechenden Angaben schachteln:

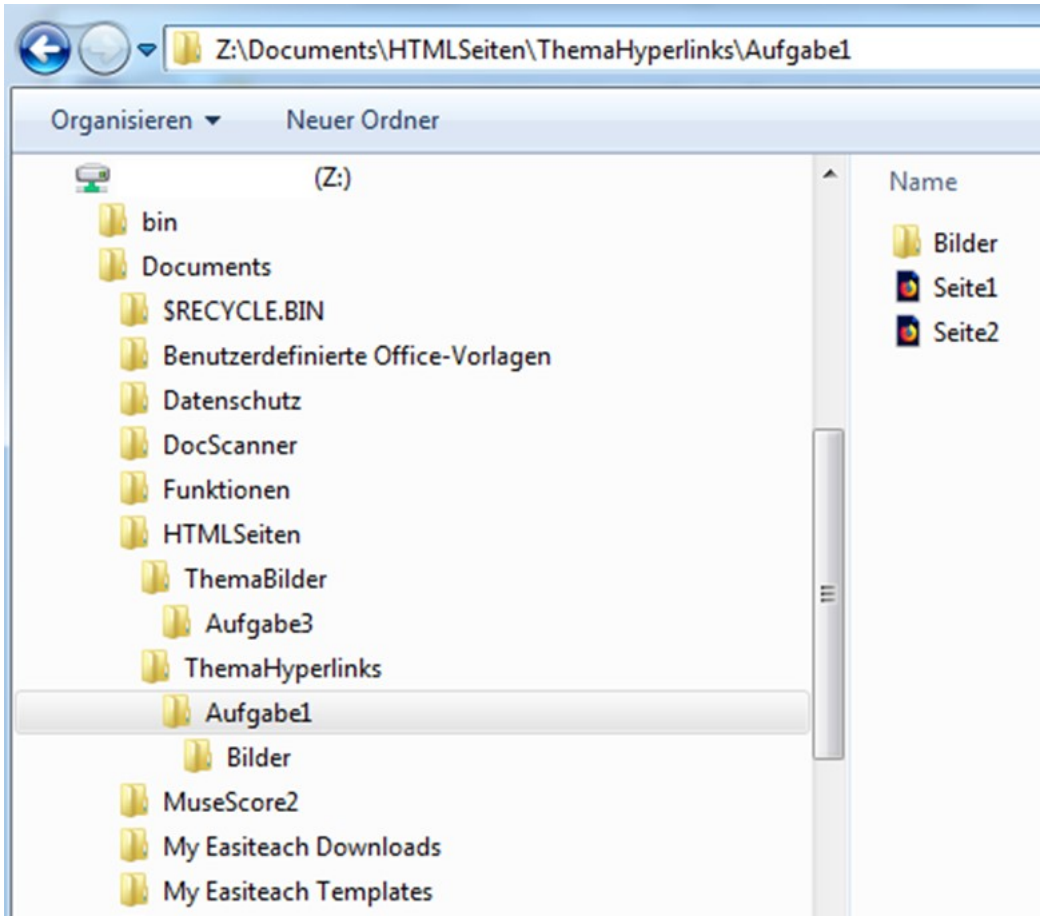
```
<a name="Ziel-1">
  <a href="#Zeile4">nach oben zurück</a>
</a>
```

Man kann auch Bilder als Absprungstelle *(Link)* benutzen:

```
<a href="#Ziel-1">
  <img src="Bildname.jpg" border="0">
</a>
```

Dient ein Bild als Link, so wird es in manchen Browsern standardmäßig immer mit einem Rahmen dargestellt. Die Rahmendicke wird durch die *border*-Anweisung festgelegt, border = 0 bedeutet dabei, dass kein Rahmen existiert.

Soll zu einem Ziel auf einer anderen HTML-Seite mit dem Namen *Seite2.html* hin gesprungen werden, so gibt es dafür mehrere unterschiedliche Anweisungsmöglichkeiten. Betrachte dafür zuerst im Windows-Explorer ein Beispiel für die mögliche Lage für die Ausgangs- und Zieldatei:



Angenommen, das aktuelle HTML-Programm sei die Datei namens *„Seite1.html“*. Innerhalb dieses Programms möchte man zu der HTML-Seite namens *„Seite2.html“* springen.

Wir setzen zunächst voraus, dass sich die beiden HTML-Seiten, wie in obiger Explorerdarstellung, in demselben Verzeichnis befinden.

****. ……****

Der *Anker* namens *Ziel-5* muss sich dann natürlich auf der *Seite2.html* befinden. Fehlt die Angabe #Ziel-5, so wird zum Anfang des neuen Dokumentes hin gesprungen:

****……****

****……****

Hierbei werden sog. **relative Pfade** angegeben. Das bedeutet, dass die Suche nach der *Seite2.html* in dem Unterverzeichnis beginnt, in welchem sich auch die Ausgangsdatei *Seite1.html* befindet. Eine relative Pfadangabe hat den Vorteil, dass man alle beteiligten HTML-Dokumente (mit beteiligten Verzeichnissen) gemeinsam an irgendeine andere Stelle im Speicherbereich des Computers bzw. des Servers verschieben kann und der Hyperlink weiterhin funktioniert

Die folgenden beiden Varianten sind sog. **absolute Pfadangaben**:

****……****

****……****

Die Suche nach der Zielseite *Seite2.html* beginnt dabei ganz oben im Stammverzeichnis Z. Hierbei hat man den Vorteil, dass die Ausgangsseite *Seite1.html* sich auch an einem anderen Ort befinden könnte. Hauptsache, die Zielseite *Seite2.html* befindet sich an dem im Link angegebenen Ort.

Beachte hierbei genau die zwei möglichen Bezeichnungen des Stammverzeichnisses Z ! Wenn man seine eigene Homepage auf einem fremden Server veröffentlichen will, sollte man grundsätzlich nur mit relativen Links arbeiten, da man nie genau weiß, unter welchem Pfad die gesamte Homepage dort abgelegt wird.

Möchte man in dem aktuellen Programm *„Seite1.html“* (im Ordner *Aufgabe 1*) einen Sprung zu einer Datei in dem Ordner *Aufgabe 3* machen, so kann man aus dem Ordner *Aufgabe 1* erst zwei Verzeichnisebenen zurückgehen, bevor man wieder bis in das Verzeichnis *Aufgabe 3* hinuntergehen kann. Dies geschieht so:
****……**.**
Auch dies bezeichnet man als **relative Adressierung**, weil die Suche nach der *Seite2.html* wieder in dem Ordner beginnt, in dem sich auch die Ausgangsseite *Seite1.html* befindet.

Alternativ wäre auch eine der beiden folgenden **absoluten Adressierungen** möglich:

<a href="/Z|/Documents/HTMLSeiten/ThemaBilder/
Aufgabe3/Seite3.html">……****

<a href="file:///Z|/Documents/HTMLSeiten/ThemaBilder/
Aufgabe3/Seite3.html">……****

Möchte man nur Verzeichnis-Ebenen tiefer gehen, so gibt man den entsprechenden Teilpfad an, z.B. im Dokument *Seite1.html*:
<img src="Bilder/Bildname">.

Natürlich kann man auch auf Internetseiten verweisen, z.B.
.....
Lesen Sie den
**** Spiegel ? ****

Bei sog. pdf-Dateien kann man sogar direkt eine bestimmte Seite anspringen:
<a href="http://archiv.ggdo.do.nw.schule.de/projekte/
InfoHomepage/Informatik/JavaGrundlagen.pdf#page=16">……
****

Normalerweise erscheint das neu angesprungene Dokument im selben Browserfenster. Das ist oft ärgerlich. Mit folgendem Zusatz wird ein zweites Browserfenster aufgebaut und das alte verbleibt im Hintergrund:
**** …… ****

Eventuell hat man eine ganze Reihe von Links. Möchte man, dass alle diese damit verknüpften Seiten in ein- und demselben Fenster dargestellt werden, so wählt man als Target einen beliebigen Namen für dieses, erst noch zu erschaffende Fenster:

**** …… ****

In Browsern ist oft eine Standardschriftgröße für *Links* voreingestellt. Allerdings kann man diese Schriftgröße auch selbst festlegen:

**** ……**.**

Man kann einen Hyperlink auch mit einer zusätzlichen sog. Quick-Info versehen:

**** GG ****

Bleibt die Maus nun etwa eine Sekunde lang über diesem Link stehen, so erscheint die entsprechende Information.

Aufgaben

1. Erstelle eine Seite, welche der Größe von mindestens drei Din-A-4-Seiten entspricht. Notfalls benutze entsprechend viele Zeilensprünge. Am Seitenanfang steht ein Link zum Seitenende und ein weiterer Link zur Seitenmitte. Am Seitenende steht ein Link zum Seitenanfang.

2. Erstelle drei unterschiedliche HTML-Seiten. Jede Seite enthalte einen kurzen Text. Auf jeder Seite steht ein *Link* zur nächsten Seite. Die letzte Seite verweist wieder auf die erste Seite.

Ein ***E-Mail-Link*** wird auf ähnliche Weise eingegeben:
.....
Bitte schreiben Sie an das
**** Goethe-Gymnasium****

Dieser Verweis vom Typ *"mailto:"* bewirkt, dass der Anwender beim Klicken auf diesen Verweis damit automatisch sein persönliches (auf seinem Rechner installiertes) Standard-E-Mail-Programm aufruft, in dem als Adressat schon die E-Mail-Adresse des Goethe-Gymnasiums eingetragen ist. Voraussetzung dafür ist natürlich, dass auf dem Rechner des Anwenders überhaupt ein E-Mail-Programm installiert ist. Außerdem sollte die angegebene E-Mail-Adresse natürlich auch existieren.

Zusätzlich zur Adresse kann man auch gleichzeitig schon den Betrefftext festlegen:

```
<a href="mailto: goethe-gymnasium@stadtdo.de
?subject= Anfrage bzgl. Klausurtermin"> Goethe-
Gymnasium</a>
```

Dabei darf hinter dem Fragezeichen und vor dem Gleichheitszeichen kein Leerzeichen stehen.

Als Hottext für den E-Mail-Link bietet sich natürlich auch immer ein kleines entsprechendes Bild an:


```
<a href="mailto: goethe-gymnasium@stadtdo.de">
  <img src="Bildname.gif" border="0" align="middle" />
  E-Mail an das Goethe-Gymnasium
</a>
```

Bilder

Die heutigen Browser sind in der Lage, viele gängige Bildformate darzustellen. Es sollte dennoch darauf geachtet werden, dass der Speicherbedarf für ein Bild nicht allzu groß wird, da sonst die entsprechende Ladezeit lästig wird.

Um eine Grafik einzufügen, genügt im einfachsten Fall die Anweisung **`<img src="Bilddateiname" />`**. Dabei ist *img src* die Abkürzung von *Image Source*. Der *img-Tag* ist ein sog. *Standalone-Tag*, d.h. er benötigt kein eigenes, abschließendes *Tag*. Stattdessen beendet man den *img-Tag* mit Leerzeichen und Slash. Durch das Leerzeichen „denken" ältere Browser, der nun folgende Slash wäre ein zusätzliches Attribut. Und weil sie dieses nicht kennen, ignorieren sie es einfach.

Beispiel: das folgende Bild zeigt meine Freundin
`<img src="freundin.gif" />`

Der Name der Bilddatei muss komplett, d.h. mit Anhängsel (im obigen Beispiel ***`.gif`***), angegeben werden. Es gibt unterschiedliche Formate für Bilddateien. Die bekanntesten sind etwa **`.gif, .jpg, .jpeg, .bmp oder .png`**
Leider ist der Windows-Explorer (welcher praktisch ein Inhaltsverzeichnis für sämtliche Dateien auf dem Computer darstellt) meistens so voreingestellt, dass er die angeblich bekannten Dateitypen (also die Anhängsel) nicht mehr darstellt. Da man für die HTML-Programmierung allerdings den vollständigen Namen einer Bilddatei angeben muss, sollte man den Windows-Explorer so einstellen, dass er die Anhängsel grundsätzlich immer mit angibt.
Dies ist in jedem Betriebssystem anders zu erreichen. In unserer Windows 7 Version, die wir 2019 in der Schule haben, funktioniert das so:
Rufe mit dem Windows-Explorer das betreffende Verzeichnis, in dem die Bilddatei steht, auf! Wähle oben im Reiter „Organisieren" die Option „Ordner- und Suchoptionen", wähle den Reiter „Ansicht" und entferne dort den Haken bei *„Erweiterungen bei bekannten Datentypen ausblenden"*!

Obiges Beispiel funktioniert allerdings nur, wenn die Bilddatei in demselben Verzeichnis steht wie deine HTML-Seite (in welche du das Bild einbinden willst).

Steht das Bild in einem anderen Verzeichnis, muss der entsprechende Pfadname angegeben werden. Als Trennstriche zwischen Verzeichnisnamen wird in HTML der normale *Slash* "/" benutzt und nicht etwa, wie im Windows-System, der *Backslash* "\" .

Beispiel: das folgende Bild zeigt meine Freundin

```
<img src = "Bilder/freundin.gif" />
```

Man kann sogar Bilder einfügen, deren Quelle direkt im Internet liegt, vorausgesetzt, man kennt den genauen Speicherort

```
<img src="http://www.HTML-seminar.de/bilder/HTML-seminar.jpg" />
```

Von Nachteil hierbei ist natürlich, dass man nie genau weiß, wie lange sich dieses Bild an der betreffenden Stelle im Internet befindet. Sicherer wäre es, dieses Bild auf die eigene Festplatte herunter zu laden. Außerdem muss in jedem Fall das Copyright des fremden Bildes entsprechend berücksichtigt werden!

Wenn man die *HTML*-Seite im Web veröffentlichen möchte, darf die Pfadangabe keine absoluten Angaben zu Windows-Laufwerken oder Ordnern haben, da der Speicherort des Bildes auf dem fremden Server im Allgemeinen nicht bekannt ist. Die folgende Pfadangabe funktioniert also nur, wenn sich die *HTML*-Seite auf dem eigenen Computer befindet:

```
<img src="file:///C|/HTML/Bilder/freundin2.gif" />
```

Leider gibt es manchmal aus verschiedenen Gründen Probleme, die Grafiken wie gewünscht anzuzeigen. Für solche Fälle gibt es die Möglichkeit, alternativ einen Text anzuzeigen:

```
<img src="freund.gif"  alt="Dies ist mein Freund" />
```

Die folgende Anweisung bewirkt, dass der Titel des Bildes auch angezeigt wird, wenn man die Maus über das Bild bewegt:

```
<img src="freund.gif" title="Dies ist mein Freund" />
```

Der *img-Tag* alleine kann ein Bild nicht innerhalb des Browserfensters zentrieren. Mit folgendem Trick gelingt dies jedoch trotzdem: Man packt das Bild in einen eigenen Absatz:

```
<p align="center">  <img src="freund.gif">   </p>
```

Wenn man möchte, dass rechts oder links um eine Grafik herum Text stehen soll, so muss man die Grafik links- oder rechtsbündig definieren:
**** Jetzt steht das Bild links vom gesamten nachfolgenden Text. Ohne das Attribut *align* findet kein vernünftiger Textfluss statt. Stattdessen würde der Text am unteren Rand des Bildes noch in derselben Zeile, in der auch das Bild steht, beginnen.

Die beiden Attribute **align="middle"** und **align="center"** sind gleichbedeutend und bewirken, dass nur die erste Zeile des folgenden Textes rechts vom Bild in der mittleren Bildhöhe dargestellt wird. Analog gibt es die Anweisungen **align="top"** bzw. **align="bottom"** .

Sehr oft möchte man nur wenige Zeilen rechts neben dem Bild haben und der restliche Text soll dann unterhalb des Bildes fortgesetzt werden. Dies lässt sich mit folgendem angepassten „Zeilensprung“ erreichen:
<br clear="all" /> erreichen.

Mit den Attributen *hspace* und *vspace* (für *horizontal* bzw. *vertical space*) lässt sich links und rechts bzw. oberhalb und unterhalb des Bildes ein Leerraum in Pixeln angeben, sodass der Text etwas Abstand vom Bild hat:
****

Wenn der Text links und rechts um die Grafik herum fließen soll, oder wenn die Grafik exakt an einer bestimmten Position stehen soll, so fügt man die Grafik in eine unsichtbare Tabelle ein. Näheres dazu steht im Kapitel über Tabellen.

Das Bild lässt sich auch mit einem Rahmen umgeben:
****
Wenn man keinen Rahmen wünscht, so gebe man *border="0"* an.

Die Breite eines Bildes lässt sich ebenfalls einstellen:
****
Dabei bezieht sich die Prozentangabe nicht auf die Größe des Originalbildes sondern auf die Breite des Browserfensters. Und die kann bei jedem Besucher der Homepage anders sein.
Man kann allerdings auch die Bildbreite auf eine konkrete Pixelanzahl festlegen,

indem man auf die Einheit Prozent verzichtet:

```
<img src="Bildname.gif"  border="20" width="20" />
```

In allen Fällen wird die Bildhöhe automatisch angepasst, damit das Bild nicht verzerrt wird. Man kann durch die Angabe von *height* auch zusätzlich die Bildhöhe beeinflussen. Beachte dann aber mögliche Verzerrungen!

Der Speicherplatzbedarf für dieses Bild wird durch die Größenangaben nicht geändert. Auch die Ladezeit wird dadurch nicht beeinflusst. Das Bild wird erst im Browserfenster mit den neu berechneten und gewünschten Maßen angezeigt.

Aufgaben

1. Erstelle eine HTML-Seite, welche drei Bilder (linksbündig, rechtsbündig, zentriert) und zusätzlichen Text enthält. Der Text soll rechts vom ersten, links vom zweiten und direkt unter dem dritten Bild stehen.

2. Das erste Bild soll keinen, das zweite einen dünnen und das dritte einen dicken Rahmen besitzen.

3. Jedes Bild soll einen zusätzlichen, kurzen Informationstext anzeigen.

4. Eines dieser Bilder soll als *Link* eingerichtet werden.

5. Erstelle eine HTML-Seite, welche mehrere sog. animierte Bilder vom Typ *gif* enthalten!

6. In eine Laufschrift (marquee-Anweisung) lässt sich auch ein Bild einbauen. Erzeuge so zwei animierte Bilder auf deiner HTML-Seite. Das erste Bild soll sich alternierend horizontal bewegen, das zweite Bild (zentriert) alternierend vertikal.

7. Schreibe zunächst auf, was das folgende Programm bewirken wird! Anschließend teste das Programm mit deinem Browser! Wie viel von deiner Vorhersage stimmte?

```
<html>
  <head>
    <title>Testseite</title>
  </head>
  <body>
    Dies ist eine Testseite!
    <br /><br />
    <p align = "center">
      <marquee  behavior="alternate" direction="right" width="300"
          scrollamount="30" scrolldelay="100"> Informatik ist sch&ouml;n
      </marquee>
    <p/>

    <p style="position: absolute; left:200px">
      <marquee  behavior="alternate" direction="down" height="300"
```

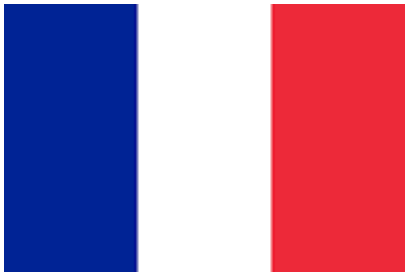
```
            scrollamount="30" scrolldelay="100">
          <img src="Asterix.gif"/>
          <br clear="all" />
          Asterix und seine Freunde
       </marquee>
     </p>

     <p style="position: relative; top:100px; left:20px">
        Dies ist relativ positionierter Text
     </p>
     <p style="position: absolute; top:500px; left:20px">
        Dies ist absolut positionierter Text
     </p>
  </body>
</html>
```

Hintergrundbilder

Mit Hilfe eines Attributes im *body-Tag* lässt sich für die gesamte Datei ein Hintergrundbild einstellen: **`<body background="Bilddateiname">`**
Falls dieses Bild kleiner als der Bildschirm sein sollte, füllt der Browser automatisch den gesamten Bildschirm mit diesem Bild auf, indem er es entsprechend oft wiederholt. Das funktioniert auch, wenn die Bildschirmbreite kein ganzzahliges Vielfaches der Bildbreite sein sollte. Im Internet findet man sehr viele solch kleine, kostenlose Bildchen, auch *Texturen* oder *Kacheln* genannt, welche sich für eine derartige Hintergrundgestaltung eignen.

Dies lässt sich folgendermaßen ausnutzen: Erstelle (zum Beispiel mit dem Windows-Hilfsprogramm namens *Paint.Net*) ein Bild, welches nur etwa 1cm hoch ist, aber über die gesamte Bildschirmbreite geht. Dieses schmale Bild soll im linken Drittel blau, im mittleren Drittel weiß und im rechten Drittel rot sein:

Benutze die Werkzeuge *Auswählen* und *Zuschneiden* von *Paint.Net*, damit dieses Minibild überall gleich hoch ist. Benutze dieses Bild als Hintergrundgrafik! Weil es schmal ist, wird es vom Browser automatisch entsprechend oft untereinander dargestellt, was zum Effekt hat, dass die ganze Webseite entsprechend in drei Hintergrundfarben aufgeteilt ist (Eingeweihte erkennen darin die Flagge eines europäischen Landes).



Hinweis: Schwarzen Text auf schwarzem Hintergrund sieht man schlecht!

Die folgende Anweisung sorgt dafür, dass das Hintergrundbild stehen bleibt, während der Seiteninhalt sich mit der Bildlaufleiste bewegt. Leider wird diese Anweisung (2017) aber nur vom Internet-Explorer verstanden:
`<body background="Bilddateiname" bgproperties = "fixed">`

Aufgabe: Erzeuge die deutsche Flagge als Hintergrundbild!

Bilddynamik

Mit der folgenden Anweisung wird ein Bild gewechselt, wenn man die Maus über dieses Bild bewegt:

```
<img  src="bild1.jpg"  border="0" name="bild"
    onmouseover="document.bild.src='bild2.jpg'";
    onmouseout="document.bild.src='bild1.jpg'" >
```

Die beiden Bilder müssen möglichst dieselben Ausmaße haben, damit es nicht zu unerwünschten Nebeneffekten kommen kann. Bedenke: Wenn das zweite Bild wesentlich kleiner sein sollte als das erste Bild, so wird zwar beim *onmouseover*-Effekt das erste Bild durch das zweite Bild ersetzt, aber gleichzeitig wird sofort das kleinere, zweite Bild praktisch von der Maus verlassen (in Wirklichkeit gar nicht erst berührt), sodass aufgrund der *onmouseout*-Anweisung das kleinere Bild sofort wieder von dem größeren Bild ersetzt wird.

Beachte weiterhin die unterschiedlichen Hochkommata und Anführungszeichen sowie das Semikolon!

Mit *document* wird die jeweils aktuelle HTML-Seite bezeichnet. Ein *document* enthält oft mehrere Objekte, z.B. Bilder, Tabellen oder Musikstücke. Diesen Objekten kann man einen Namen geben. Im obigen Beispielsbefehl erhält das entsprechende Image den Namen *„bild“*.

Aufgaben

1. Fast immer genügt im Internet ein Klick mit der rechten Maustaste auf ein Bild, und der Homepagebesucher hat anschließend die Möglichkeit, sich dieses Bild auf den eigenen Rechner herunterzuladen. Falls der Homepage-besitzer dies verhindern möchte, könnte er z.B., wie oben beschrieben, das eigentliche Bild durch ein anderes überdecken lassen, sobald sich die Maus über das eigentliche Bild bewegt. Gespeichert werden könnte dann nur das Überdeckungsbild. Programmiere dies! Achte darauf, dass die beiden Bilder möglichst exakt dieselbe Breite und Höhe besitzen! Was geschieht, wenn die beiden Bildgrößen deutlich unterschiedlich sind?

Untersuche anschließend auch, ob du (als dein eigener Besucher deiner gerade erstellten HTML-Seite) in der Lage bist, das gewünschte Bild herunterzuladen!

Imagemaps (Abbildungspläne)

Es ist möglich, Teile eines Bildes als Verweise (Links) zu markieren. Zum Beispiel könnte man in Stadtplänen durch Anklicken bestimmter Bezirke neue, vergrößerte Karten dieses Stadtteils aufrufen. Oder durch Anklicken eines Museums auf dem Stadtplan könnte man Informationen über dieses Museum abrufen. Oder durch Anklicken einer Person auf einem Klassenfoto zu deren Homepage gelangen, usw.

Nehmen wir an, wir hätten irgendein Bild auf unserer Web-Seite. Im Folgenden wird beschrieben, wie man auf diesem Bild sog. *"Hotzones"* (das sind die Bildbereiche, die beim Anklicken einen Sprung auf irgendein Ziel veranlassen) einrichtet.

Das Aufrufen des Bildes und die Verteilung von *Hotzones* in diesem Bild finden an verschiedenen Stellen im HTML-Quelltext statt. Natürlich muss **vor** der Darstellung des Bildes geklärt sein, wo dort die einzelnen *Hotzones* liegen sollen. Deshalb sollten möglichst direkt hinter dem *body-Tag* im HTML-Quelltext die entsprechenden Befehle für die *Hotzones* verschiedener Bilder stehen.
Die *Hotzones* können **u.a** die Form eines Rechteckes (shape="rect"), eines Kreises (shape="circle") oder eines beliebig großen Vielecks (shape = "poly") haben.

Bei rechteckigen *Hotzones* gibt man in *Pixeln* (*Pixel* = picture element) die Koordinaten zweier gegenüberliegender Rechteckpunkte an: zum Beispiel den Eckpunkt oben links und den Eckpunkt unten rechts. Bezüglich der Koordinaten gilt: der Ursprung liegt oben links. Die x-Werte weisen nach rechts, die y-Werte nach unten.

Mit Bildbearbeitungsprogrammen (z.B. mit dem Programm *Paint*, welches zum Windows-Zubehör gehört) lassen sich sehr einfach die Pixelkoordinaten einzelner Bildpunkte ermitteln.

Bei kreisförmigen *Hotzones* gibt man die beiden Koordinaten des Kreismittelpunktes und zusätzlich den Radius an.

Bei *Hotzones* in Vieleckform gibt man die Koordinaten aller Eckpunkte an. Mit Vielecken kann man eine *Hotzone* präziser festlegen (wenn die *Hotzone* z.B. ein Kopf auf einem Gruppenfoto sein soll).

Im Folgenden werden für ein Bild drei unterschiedliche *Hotzones* eingerichtet:

```
.....................
<body>
  <map name="Einteilung_Bild1">
     <area shape="rect" coords="20, 50, 70, 120"
         title="Hörde"  href="Seite2.htm">
     <area shape="circle" coords="150, 50, 20"
         title="Sölde"  href="Seite3.htm">
     <area shape="poly" coords = "200, 50, 200, 60,
         230, 50, 270, 20, 240, 10"  title="Asseln"
         href="Seite4.htm">
  </map>
  ....................
  <img src = "Dortmund.gif" usemap="#Einteilung_Bild1">
  ....................
</body>
```

Die *title*-Anweisung im obigen Beispiel bewirkt, dass der zugehörige Text kurz angezeigt wird, wenn die Maus an dieser Stelle wartet. Falls die *Hotzone* keine Sprungadresse beinhalten sollte, funktioniert **obige** *area*-Anweisung nicht.

Aufgaben

1. Lade von der Goethe-Homepage das aktuelle Foto des Lehrerkollegiums herunter und speichere es in deinem Verzeichnis. Die Köpfe von mindestens drei Personen sollen als *Hotzones* eingerichtet werden. Wenn der Mauszeiger auf einem dieser drei Köpfe ruht, soll der entsprechende Lehrername eingeblendet werden.
Verwende eine rechteckige, eine kreisförmige und eine vieleckige *Hotzone*!
Mit einem Bildbearbeitungsprogramm kannst du z.B. nur den Kopf einer Person ausschneiden, vergrößern und als neues Bild speichern. Klickt man nun im Gruppenfoto auf den Kopf dieser Person, so soll eine neue HTML-Seite erscheinen, welche den vergrößerten Kopf als Bild und weitere Informationen zur dargestellten Person enthält.

2. Erstelle mehrere HTML-Seiten, welche jeweils eine einzige Landkarte von Europa, Deutschland, Dortmund, Hörde oder Wellinghofen enthalten. Jede Karte soll eine entsprechend platzierte *Hotzone* haben, mit deren Hilfe man auf die nächste Karte gelangt. Zum Schluss sollte man auf ein Bild des Goethe-Gymnasiums gelangen.

Listen

Eine normale, nicht durchnummerierte (also **u**ngeordnete) **L**iste wird durch ** ** eingeleitet und mit ** ** beendet. Den einzelnen Aufzählungspunkten wird automatisch ein Symbol vorangestellt. Als Symbol kann man wählen **<ul type = "circle" | "square" | "disc" >**. Jedem Listenpunkt muss im Quelltext das Zeichen **<li>** vorangestellt sein, wobei *li* für ***l**ist **i**tem* steht.

```
<ul>
  <li> dieses ist der erste Eintrag  </li>
  <li> jetzt kommt der nächste </li>
  <li> und nun der letzte Eintrag </li>
</ul>
```

- dieses ist der erste Eintrag
- jetzt kommt der nächste
- und nun der letzte Eintrag

Hinweis: Leider ist die Darstellung in Firefox etwas anders als im Internet Explorer.

Geordnete Listen werden vom Browser automatisch durchnummeriert. Sie stehen zwischen **
** und **
** *(englisch: **o**rdered **l**ist).*

```
<ol>
   <li> Dieser Zeile wird automatisch die Ziffer 1
        vorangestellt </li>
   <li> zweite Zeile </li>
   <li> diese Zeile hat als Präfix automatisch die
        Ziffer 3 </li>
</ol>
```

1. Dieser Zeile wird automatisch die Ziffer 1 vorangestellt
2. zweite Zeile
3. diese Zeile hat als Präfix automatisch die Ziffer 3

Man kann auch den Anfangswert der Nummerierung festlegen:

```
<ol start = "4">
   <li> erste Zeile </li>
   <li> zweite Zeile </li>
   <li> dritte Zeile </li>
</ol>
```

4. erste Zeile
5. zweite Zeile
6. dritte Zeile

Möchte man die Listeneinträge nicht mit Zahlen durchnummerieren sondern etwa mit Buchstaben oder mit römischen Zahlen, so verwende man einen entsprechenden Type-Parameter:

```
<ol type="A"|"a"|"1"|"i"|"I">
```

Man kann auch den Anfangswert der Nummerierung festlegen:

```
<ol type="A" Start="4">          <--Beginn mit Buchstabe D  -->
   <li> Apfel </li>
   <li> Birne </li>
</ol>
```

D. Apfel
E. Birne

Listen können auch beliebig ineinander geschachtelt werden.

Aufgabe

1. Erstelle folgende geschachtelte Liste:

A. Naturwissenschaften
    - Physik
        - Atomphysik
        - Elektrizitätslehre
        - Optik
    - Biologie
        - Fauna
        - Flora
    - Chemie
        - schwarze Magie
        - weiße Magie

B. Fremdsprachen
    - Englisch
        - britisch
        - amerikanisch
    - Latein
    - Französisch

C. Gesellschaftswissenschaften

Tabellen

In einer HTML-Tabelle dürfen einzelne Tabellenzellen im Prinzip leer sein. Dabei gibt es jedoch ein Problem: viele Browser stellen dann die Tabelle nicht mehr richtig dar. Aus diesem Grunde sollte jede Tabellenzelle einen Inhalt haben. Als Inhalt für leere Zellen setzt man das nicht umbrechende Leerzeichen * * oder das normale Leerzeichen * * selbst ein.

<table> und **</table>** legen eine Tabelle fest. Wenn die Tabelle sichtbare Trennlinien enthalten soll, gibt man zusätzlich *border* an, also **<table border>.** Damit sind sowohl der äußere Rahmen als auch die inneren Gitternetzlinien gemeint. Dabei ist die Strichdicke für die Trennlinien auf 1 voreingestellt.

Fehlt die *border*-Anweisung, so erhält man eine sog. *blinde* Tabelle.

Muppet-Show	Sesamstrasse
Miss Piggy	Ernie
Kermit	Bert

```
<table border>
   <tr>
        <th> Muppet-Show </th>
        <th> Sesamstrasse </th>
   </tr>
   <tr>
        <td> Miss Piggy </td>
        <td> Ernie </td>
   </tr>
   <tr>
        <td> Kermit </td>
        <td> Bert </td>
   </tr>
</table>
```

Zwischen **<tr>** und **</tr>** werden die einzelnen Zeilen der Tabelle (table **r**ow) festgelegt. Die erste Zeile legt im Übrigen die Anzahl der Spalten für die gesamte Tabelle fest. **<th>** und **<td>** bestimmen die Inhalte (**t**able **d**ata) der einzelnen Zellen. Dabei wird **<th>** meist für Überschriften (= **t**able **h**ead)

benutzt. Diese werden standardmäßig zentriert und fettgedruckt hervorgehoben.
<td> leitet eine normale Tabellenzelle ein, standardmäßig linksbündig.

Die folgende Angabe der Strichdicke beeinflusst nur den äußeren Rahmen:

```
<table border = "15">
```

Muppet-Show	Sesamstrasse
Miss Piggy	Ernie
Kermit	Bert

```
<table border = "15">
   <tr>
       <th> Muppet-Show </th>
       <th> Sesamstrasse </th>
   </tr>
   .....
</table>
```

In der folgenden Tabelle fehlt ein Außenrahmen, aber alle inneren Gitterlinien sind vorhanden.

Weg s in m	1	2	4	7
Zeit t in s	3	6	12	21

```
<table  rules = "all">
   <tr>
      <td> Weg s in m </td>
      <td> 1 </td>
      <td> 2 </td>
      <td> 4 </td>
      <td> 7 </td>
   </tr>
   <tr>
      <td> Zeit t in s </td>
      <td> 3 </td>
      <td> 6 </td>
      <td> 12 </td>
      <td> 21 </td>
   </tr>
</table>
```

Die Anweisung *rules* beeinflusst die inneren Gitternetzlinien.

Genauer gilt:

rules = "none"	keine inneren Gitternetzlinien
rules = "all"	alle inneren Gitternetzlinien
rules = "rows"	Linien nur zwischen den Zeilen
rules = "cols"	Linien nur zwischen den Spalten

Im Folgenden werden einige Attribute im *Table-Tag* und deren Auswirkungen vorgestellt:

`<table width = "80%" border >` Die Prozentangabe bezieht sich hierbei auf die Bildschirmbreite. Alternativ könnte man die Tabellenbreite aber auch in Pixel angeben.

Muppet-Show	Sesamstrasse
Miss Piggy	Ernie
Kermit	Bert

Analog funktioniert das Attribut *height*.

Der Abstand des Zellinhaltes vom inneren Rand einer Zelle lässt sich so festlegen:

```
<table border  cellpadding = "15">
```

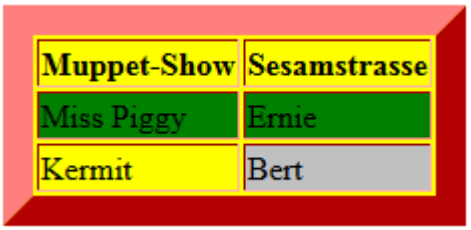
Muppet-Show	Sesamstrasse
Miss Piggy	Ernie
Kermit	Bert

Der Abstand der inneren Gitternetzlinien, welche etwa 1 Pixel breit sind, voneinander wird durch die Anweisung *cellspacing* bestimmt. Dieser hat standardmäßig den Wert von 2 Pixel, lässt sich aber natürlich ändern:

```
<table border = "1"  cellspacing = "10">
```

Muppet-Show	Sesamstrasse
Miss Piggy	Ernie
Kermit	Bert

Die äußere und innere Rahmenfarbe wird mit *bordercolor* festgelegt. Die Hintergrundfarbe lässt sich mit dem Parameter *bgcolor = #RGB* bestimmen, und zwar sowohl für die gesamte Tabelle oder nur für eine Zeile oder speziell nur für eine Zelle. Dabei sind die Variablen R, G und B die bereits bekannten zweistelligen Hexadezimalzahlen. Alternativ kann man natürlich auch einen englischen Farbnamen verwenden.




```
<table border="15" bordercolor="red"
     bgcolor="yellow">
   <tr>
       <th> Muppet-Show </th>
       <th> Sesamstrasse </th>
   </tr>
   <tr bgcolor = "green">
       <td> Miss Piggy </td>
       <td> Ernie </td>
   </tr>
   <tr>
       <td> Kermit </td>
       <td bgcolor = "silver"> Bert </td>
   </tr>
</table>
```

Eine weitere Möglichkeit, eine Tabelle graphisch zu verbessern, wäre die Verwendung eines Hintergrundbildes. Man kann ein Hintergrundbild entweder für die gesamte Tabelle oder nur für einzelne Zellen einrichten. Dabei ist allerdings darauf zu achten, dass der eigentliche Tabelleninhalt noch lesbar bleibt:


```
<table  border background = "Bildname.jpg">
.....
</table>
```

Tabellen lassen sich auch mit der *caption*-Anweisung beschriften. Diese sollte am besten direkt hinter dem *table*-Tag folgen.

Muppet-Show	Sesamstrasse
Miss Piggy	Ernie
Kermit	Bert

Stand: 29.09.2013

```
<table  border background = "Bildname.jpg">
  <caption align = "bottom"> Stand: 29.09.2013 </caption>
  <tr>
    <th> Muppet-Show </th>
    <th> Sesamstrasse </th>
   </tr>
.....
</table>
```

Dabei sind folgende *caption*-Werte möglich:

```
<caption align = "top">
<caption align = "left">
<caption align = "right">
<caption align = "bottom">
```

Ohne das Attribut *align* findet kein Textfluss um die Tabelle statt. Die gesamte Tabelle kann man im Browserfenster linksbündig, rechtsbündig oder zentriert ausrichten, z.B. mit **`<table align = "right">`**
Das Attribut *align* kann die Werte `"left"`, `"center"` und `"right"` annehmen.

Analog kann man fast alle Tabellenattribute auch nur für eine einzelne Zelle festlegen. Beispiel: **`<td height = "250" bgcolor = "red">.`**
Gibt man etwa die Höhe für eine einzige Zelle an, so gilt derselbe Höhenwert für alle Zellen in dieser Zeile.
Analog: die Definition der Spaltenbreite in einer Zelle hat zur Folge, dass alle Zellen in dieser Spalte diese Breite besitzen. Deshalb sollte man die Spaltenbreite immer in der ersten Zeile der Tabelle vornehmen.

Analog zur Horizontalausrichtung (links, mitte, rechts) gibt es auch ein Attribut namens *valign* (= vertical align, oben, mitte, unten). Dieses Attribut kann die Werte "top", "middle" und "bottom" annehmen. Betrachte dazu folgendes Beispiel:

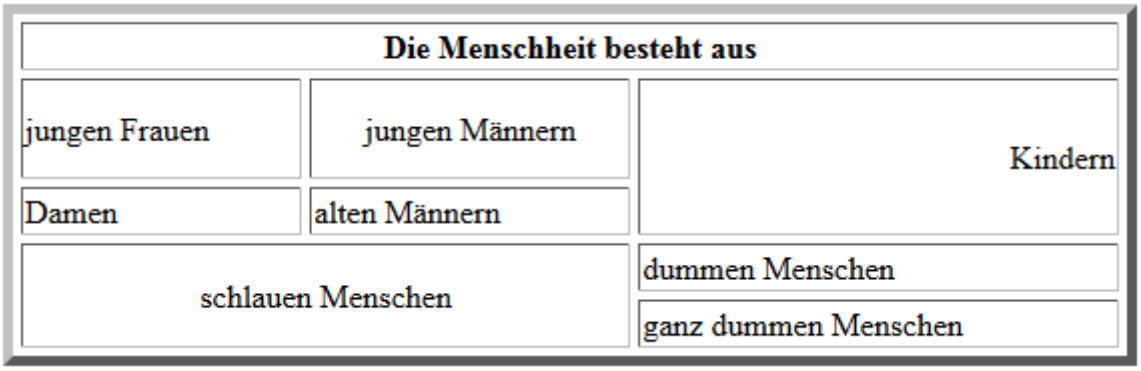

```
<table border  height = "120">
   <tr>
     <th valign = "top"> Muppet-Show </th>
   </tr>
   <tr>
     <td valign = "middle"> Miss Piggy </td>
   </tr>
   <tr>
     <td valign = "bottom"> Kermit </td>
   </tr>
</table>
```

Es ist vorteilhaft, die Anzahl, Breite und Ausrichtung der Spalten sofort hinter dem *Table-Tag* festzulegen. Die Abkürzung *colgroup* steht für column group = Spaltengruppe.

```
<table>
   <colgroup>
      <col width = "30%" align = "center">
      <col width = "70%" align = "right">
   </colgroup>
   <tr>.....</tr>
    ..................
</table>
```

Die Prozentangaben beziehen sich hier auf die Breite der Tabelle, nicht auf die Fensterbreite. Natürlich kann man die Spaltenbreite auch konkret in Pixeln angeben.

Man kann mehrere horizontal nebeneinander liegende Zellen zusammenfassen. Analog kann man auch mehrere vertikal untereinanderliegende Zellen zusammenfassen und es lässt sich sogar beides gleichzeitig anwenden:




```
<table border = "5" cellspacing = "4"  width = "90%">
  <tr>
     <th colspan= "3">Die Menschheit besteht aus</th>
  </tr>
  <tr>
     <td> jungen Frauen </td>
     <td align = "center"  height = "50"> jungen
         Männern </td>
     <td align = "right" rowspan = "2"> Kindern </td>
  </tr>
  <tr>
     <td> Damen </td>
     <td> alten Männern </td>
  </tr>
  <tr>
     <td align= "center" rowspan = "2" colspan = "2">
         schlauen Menschen </td>
     <td> dummen Menschen </td>
  </tr>
  <tr>
     <td> ganz dummen Menschen </td>
  </tr>
</table>
```

Möchte man, dass rechts neben der Tabelle der Text weiterfließt, so schreibt man noch zusätzlich in den ***TABLE**-Tag*: **`<table align = "left">`**

Schreibt man Text neben einer Tabelle, so ist oft das Attribut *clear* im *ZeilenumbruchTag* hilfreich. Dieser Zusatz verschiebt den Zeilenumbruch soweit nach unten, bis entweder ein Rand frei ist, **`<br clear = "left" />`** oder **`<br clear = "right" />`**, oder beide Ränder frei sind: **`<br clear = "all" />`**

Beispiel:
```
<table border  align = "right"  width="50%">
.............
</table>
Dieser Text steht
<br /> links neben der Tabelle
<br clear = "all">
Jetzt geht es unterhalb der Tabelle weiter.
```

Eine Tabellenzelle kann auch eine weitere Tabelle beinhalten.

Sog. blinde Tabellen – das sind Tabellen ohne sichtbare Gitternetzlinien - bieten sich beispielsweise an, wenn man einen Text mehrspaltig darstellen will.

Aufgabe 1 Erstelle eine blinde Tabelle mit zwei Zeilen und drei Spalten! Jede Zelle soll ein genau passendes Bild enthalten.	**Aufgabe 2** Erstelle eine Tabelle! Eine Zelle soll eine weitere Tabelle enthalten.	**Aufgabe 3** Erstelle eine Farbtabelle, bestehend aus fünf Zeilen und zwei Spalten! Jede Zelle stellt eine Farbe dar. Der Text in der Zelle enthält den Farbcode als Hexadezimalziffer, der Hintergrund entspricht genau dieser Farbe.

```
<table cellpadding = "20" width = "500"  >
  <tr>
    <td width="33%" valign = "top">
      <strong>Aufgabe 1 </strong><br />
      Erstelle eine blinde Tabelle mit zwei Zeilen und drei Spalten! Jede
      Zelle soll ein genau passendes Bild enthalten.
    </td>
    <td width="33%" valign = "top">
      <strong>Aufgabe 2 </strong><br />
      Erstelle eine Tabelle! Eine Zelle soll eine weitere Tabelle
      enthalten.
    </td>
    <td valign= "top">
      <strong>Aufgabe 3</strong><br />
      Erstelle eine Farbtabelle, bestehend aus fünf Zeilen und zwei Spalten!
      Jede Zelle stellt eine Farbe dar. Der Text in der Zelle enthält den
      Farbcode als Hexadezimalziffer, der Hintergrund entspricht genau
      dieser Farbe.
    </td>
  </tr>
</table>
```

Eine weitere Anwendung für blinde Tabellen wäre die Möglichkeit, einen relativ schmalen Text farbig zu hinterlegen:

Informatik ist das schönste Fach in der Schule. Außerdem ist unser Informatiklehrer wirklich ganz große Klasse. Später möchte ich auch Informatiker werden.

```
<table  bgcolor = "yellow" width = "300" >
   <tr>
     <td> Informatik ist das schönste Fach in der Schule. Außerdem ist
       unser Informatiklehrer wirklich ganz große Klasse. Später möchte ich
       auch Informatiker werden.
     </td>
   </tr>
</table>
```

Formulare

Es besteht die Möglichkeit, dass der Besucher deiner Homepage Daten in Formulare einträgt. Die einzelnen Formularinhalte könnte man später zum Beispiel mit Hilfe von *JavaScript* auswerten. Wenn sehr viele Besucher dasselbe Formular ausfüllen und als E-Mail abschicken, so gibt es für die Auswertung Hilfsprogramme, die oft in der Programmiersprache *Perl* oder in *PHP* geschrieben sind.

Es besteht also die Möglichkeit, dass die ins Formular eingetragenen Daten automatisch als E-Mail zu einer angegebenen Adresse geschickt werden. Das funktioniert allerdings nur, wenn der Besucher deiner Homepage auf seinem eigenen Rechner ein funktionierendes E-Mail-Programm installiert hat.

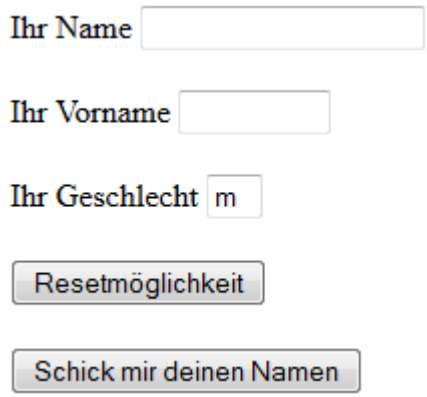
Im Folgenden werden wir uns auf die Erstellung von Formularen beschränken.

Jedes Formular wird durch **<form>** eingeleitet und mit **</form>** geschlossen. Diese Formulare müssen natürlich im Body-Teil der *HTML*-Seite stehen.

Formularelemente müssen nur dann zwingend zwischen **<form>** und **</form>** stehen, wenn das Formular auch an einen Server gesendet werden soll.
Für reine JavaScript-Berechnungen beispielsweise sind die beiden umgebenden <form>-Tags nicht notwendig.

Man kann in *HTML* zwar Formulare definieren und erstellen, für eine Auswertung oder Weiterverarbeitung der Eingaben benötigt man jedoch eine Programmiersprache wie beispielsweise *JavaScript* oder *PHP*.

Einzeilige Eingabefelder



Der *HTML*-Quelltext für das obige Formular lautet:

```
<form action = "mailto:w.wacker@t-online.de"
   method="post"  name="persoenlichesDatenformular">
   Ihr Name <input type = "text"  name="Lesername"
       size = "20"/> <br /> <br />
   Ihr Vorname <input type = "text"  name = "VN"
         size = "9" maxlength = "9" /> <br /> <br />
   <br /> <br />
   Ihr Geschlecht <input type = "text"  name ="G"
         size = "1"   value="m"/>
   <br /> <br />
   <input type = "reset" value =
          "Resetm&ouml;glichkeit"/>
   <br /> <br />
   <input type = "submit"  value = "Schick mir deinen
         Namen"/>
</form>
```

Das englische Wort *„to submit“* bedeutet „einreichen“.
Dem Empfänger kann man damit eine E-Mail mit dem Formularinhalt schicken. Wenn E-Mails verschickt werden, so werden die Daten für den Transport verschlüsselt. Dabei müssen z.B. die Daten von Bildern oder ganzen Dateien grundsätzlich anders verschlüsselt werden als reine Textdaten. Besteht der Formularinhalt praktisch nur aus Text, so setzt man im anfänglichen *form-Tag* zusätzlich das Attribut *enctype = "text/plain"* , also:

```
<form action = "mailto:w.wacker@t-online.de"
         method = "post"  enctype="text/plain" >
```

In diesem Fall erhält man beispielsweise eine E-Mail mit dem Inhalt "Lesername=Meier, VN=Katja, G=w".

Allerdings ist diese Verschlüsselungseinstellung auch der sog. Default-Wert, d.h., wenn diese Angabe fehlen sollte, so wird automatisch die Kodierung *enctype = "text/plain"* durchgeführt.

Sehr häufig werden Formulare gar nicht als E-Mail verschickt. Sie werden stattdessen oft benutzt, um z.B. mithilfe von *JavaScript* direkt die Eingaben des Besuchers auf dieser HTML-Seite auszuwerten und die weitere Gestaltung dieser HTML-Seite abhängig von den gemachten Eingaben zu gestalten. In diesem Fall fehlt natürlich sowohl die *action*-Angabe im *form-Tag* als auch das *submit*-Eingabefeld.

Alle Eingabefelder werden durch den *input-Tag* bestimmt. Beachte, dass alle *input-Tags* als sog. *stand-alone-Tags* intern geschlossen werden müssen. Es gibt verschiedene Typen von Eingabefeldern, die durch das Attribut *type* festgelegt werden, z.B. für die Eingabe von Texten (type = „text“), von Zahlen (type = „number“), von Passwörtern (type = „password“), von Zeitangaben (type = „date“) und weitere mehr.

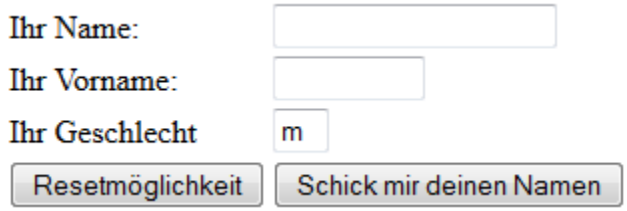
Der sog. Default-Wert für das *type*-Attribut ist *text*. Ein fehlendes *type*-Attribute ist also gleichbedeutend mit type = „text“.

Jedes Eingabefeld kann und sollte einen Namen erhalten (ebenso wie das Formular selbst, denn man kann auch mehrere Formulare auf einer einzigen Web-Seite haben), damit man bei der Auswertung des Formulars auch direkt auf dieses Eingabefeld zugreifen kann.

Das einzeilige Text-Eingabefeld im obigen Formular für den Lesernamen wird mit der sog. optischen Anzeigenlänge (= Anzahl der Zeichen) *size* auf eine sichtbare Breite von 20 Zeichen eingestellt. Allerdings kann der Benutzer auch mehr als 20 Zeichen eintragen; dabei kann man mit dem Attribut *maxlength* eine Begrenzung für die maximale Zeichenzahl angeben. Wenn *maxlength* größer sein sollte als *size*, wird gescrollt.

Mit *value* kann man schon sog. Default-Werte eingeben, die sich aber überschreiben lassen.

Obiges Formular sieht strukturierter aus, wenn man es in eine blinde Tabelle einbettet:



```
<form action = "mailto:w.wacker@t-online.de" method = "post">
<table>
   <tr>
    <td> Ihr Name:</td>
    <td> <input type = "text" name="Lesername"  size = "20"/>
    </td>
   </tr>
   <tr>
     <td> Ihr Vorname:</td>
     <td> <input type = "text"  name = "VN"  size = "9"
               maxlength = "9" /> <td>
   </tr>
   <tr>
     <td> Ihr Geschlecht </td>
     <td> <input type = "text"  name ="G" size = "1"
              value = "m"/> </td>
   </tr>
   <tr>
     <td> <input type = "reset" value =
                  "Resetm&ouml;glichkeit"/> </td>
     <td>  <input type = "submit"  value = "Schick
              mir deinen Namen"/> </td>
   </tr>
</table>
</form>
```

Auch ein Passwortfeld ist möglich. Dafür schreibt man etwa

```
<input type = "password" name = "Geheimeingabe"
    size = "25" maxlength = "20" />
```

Hierbei wird man bei der Eingabe vor neugierigen Blicken geschützt, da bei der Eingabe statt der Werte Sternchen erscheinen.

Betrachte folgendes Beispiel zur Eingabe von Zahlen:

```
<input name="groesse" type="number" min="100"
                    max="220" step="0.1" value="175"> cm
```

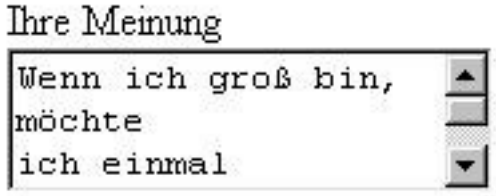
Im obigen Beispiel kann man Zahlen zwischen 100 und 220 eingeben. Die Eingabe wird sofort nach Beendigung überprüft. Man kann die Eingabewerte hier in 0.1-Schritten erhöhen. Fehlt die *step*-Angabe, so wird *step*=“1“ angenommen. Mit *value* wird ein Wert vorbelegt.

Folgendes Eingabefeld entspricht einem einstellbaren Schieberegler:

```
<input name="Intelligenz" type="range" min="50"
                     max="150" value="90"/>
```

Mehrzeilige Eingabefelder

Ein **textarea**-Element ist ein mehrzeiliges Eingabefeld (*textarea = Textbereich*). Es kann noch im Browser über die rechte untere Ecke in seiner Größe verändert werden. Man kann seine Größe aber auch über die Attribute *rows* und *cols* festlegen.




```
<form>
   Ihre Meinung <br />
   <textarea  name = "Bemerkungen"  rows="3"
          cols="20"  wrap = "soft"> </textarea>
</form>
```

Die Anweisung *wrap="soft"* sorgt dafür, dass ein automatischer Zeilenumbruch stattfindet. Ältere Browser vergessen sonst das Umbrechen des Textes. Das Attribut *cols* bestimmt die Anzahl der Zeichen pro Zeile, also die Breite des Textfeldes. Mit *rows* wird die Anzahl der anzuzeigenden Zeilen eingestellt. Allerdings kann der Benutzer unabhängig von dieser Voreinstellung weitere Zeilen eintragen. Falls der Feldinhalt nicht mehr sichtbar ist, wird der Rollbalken aktiv.

Man kann ein mehrzeiliges Eingabefeld mit Inhalt vorbelegen. Im Gegensatz zu den einzeiligen Eingabefeldern hat **<textarea>** kein `value`-Attribut. Um mehrzeilige Eingabefelder mit Text vorzubelegen, notiert man den gewünschten Text einfach als Elementinhalt zwischen den Tags **<textarea …..> und </textarea>**. Leerzeichen und Zeilenumbrüche im Quelltext werden als solche dargestellt.

Alle Eingabefelder lassen sich auch mit der Eigenschaft *readonly* erstellen. Diese können vom Besucher der Seite nicht überschrieben werden (und sind damit strenggenommen gar keine Eingabefelder mehr). Beispiel:

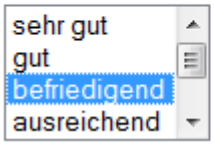
```
<input type = "textarea"  name="Lizenz"  readonly/>
Mit dem Absenden dieses Formulars erkläre ich mich mit den
Lizenzbedingungen einverstanden.</textarea>
```

Derartig geschützte Eingabefelder werden für Standardvorgaben benutzt.

Auswahlfelder

Im Folgenden werden sog. Auswahlfelder in Formularen dargestellt:

Wie gefällt euch dies ?



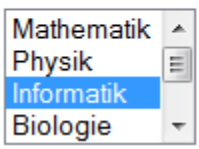
Das obige Auswahl-Feld erhält man durch

```
<form>
   <H2>Wie gef&auml;llt euch dies ? </H2>
   <select name = "Urteil"   size = "4">
      <option> sehr gut </option>
      <option> gut </option>
      <option selected> befriedigend </option>
      ..........................................
      <option> ungen&uuml;gend </option>
   </select>
</form>
```

Ein Auswahlfeld wird durch **<select>**...**</select>** realisiert. Die obige Angabe *size="4"* bewirkt, dass in der Auswahlbox nur vier Zeilen angezeigt werden. Die Anzahl der Optionen kann jedoch beliebig groß sein. Die weiteren Zeilen sind dann mit Hilfe des Scroll-Balkens erreichbar. Wird die Größenangabe *size* = "1" gewählt, so spricht man auch von einer sog. Drop-Down-Liste.
Die Breite einer Auswahlliste wird in Abhängigkeit des längsten Listeneintrags gewählt und geschieht automatisch.

Wichtig bei Auswahlfeldern ist, dass der Besucher normalerweise nur eine von mehreren Möglichkeiten auswählen kann. Dies lässt sich jedoch mit dem Attribut *multiple* im *select-Tag* ändern. Die Mehrfachauswahl geschieht dann durch das Halten der [*Strg*]-Taste und dem Anklicken der jeweiligen Listenelemente. Ein dezenter Hinweis für den Surfer auf diese Technik sollte nicht fehlen.

Wähle mit Hilfe der [Strg]-Taste zwei Leistungskurse!

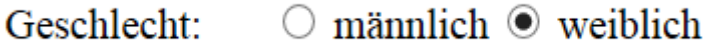


```
<form>
   <h2>W&auml;hle mit Hilfe der [Strg]-Taste zwei
          Leistungskurse! </h2>
   <select name = "Fach"  size = "4"  multiple>
      <option> Mathematik </option>
      <option> Physik </option>
      <option selected> Informatik </option>
      <option> Biologie </option>
      <option> Chemie </option>
      <option> Englisch </option>
   </select>
</form>
```

Weiterhin ist es möglich, den „Absendewert“ eines Eintrags zu verändern. Normalerweise wird beim Senden bzw. beim Auswerten des Formulars der Text des ausgewählten Eintrags übertragen. Mithilfe des Attributes *value* kann man innerhalb des *option-Tags* einen anderen Wert setzen:

```
<form>
   <h2>W&auml;hle mit Hilfe der [Strg]-Taste zwei
          Leistungskurse! </h2>
   <select name = "Fach"  size = "4"  multiple>
      <option value = "M"> Mathematik </option>
      <option value = "Ph"> Physik </option>
      <option value = "In" selected> Informatik
          </option>
      <option value = "Bi"> Biologie</option>
      <option value = "Ch"> Chemie </option>
      <option value = "E"> Englisch </option>
   </select>
</form>
```

Eine weitere Möglichkeit, "Eins aus vielen" auszuwählen, erhält man mit sog. **Radio-Button**s:




```
<form>
  Geschlecht:         
  <input type = "radio" name="Geschlecht"
      value = "m" /> m&auml;nnlich
  <input type = "radio" name="Geschlecht"
      value = "w" checked /> weiblich
</form>
```

Damit mehrere Buttons zu einer Gruppe gehören, müssen sie denselben Namen haben. Außerdem wird der Wert (=die Aussage, *value*) des jeweiligen Radio-Buttons gleich direkt im Tag mit eingetragen. Beachte: der Wert des Buttons ist in der Variablen *value* gespeichert, nicht in dem Text hinter dem Button! Mit dem Attribut *checked* im *input-Tag* kann man dem Surfer schon eine Auswahl vorschlagen.

Mit sog. **Check-Boxen** kann man auch Mehrfachauswahlen treffen:

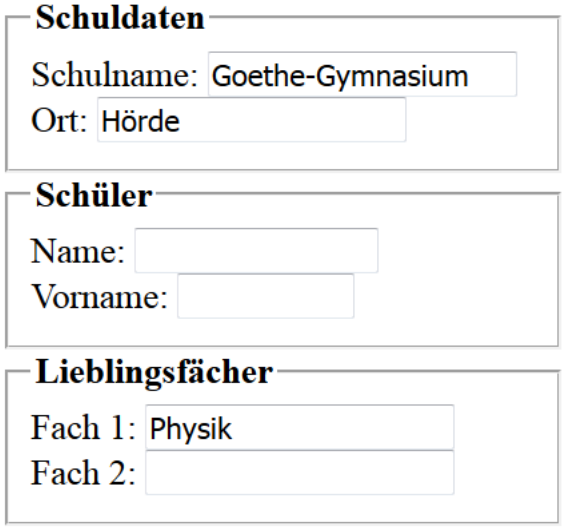

```
<form>
   Welche Filme finden Sie gut?  <br />
   <input type = "checkbox" name = "Film" value="W"/>
     Winnetou <br />
   <input type ="checkbox" name="Film"  value="T"
      checked/>
     Titanic <br />
   <input type ="checkbox" name="Film"  value="HP"/>
     Harry Potter <br />
   <input type ="checkbox" name="Film"  value="Mar"/>
     Marienhof<br />
</form>
```

Hier gilt ebenso wie bei Radio-Buttons: Damit mehrere Checkboxen zur selben Gruppe gehören, müssen sie denselben Namen haben.

Mit dem Attribut *checked* im *input-Tag* kann man dem Surfer schon eine Auswahl vorschlagen.

Gruppierung von Formularteilen

Um die Übersichtlichkeit eines Formulars zu verbessern, lassen sich die einzelnen Formularelemente zu Gruppen zusammenfassen. Das folgende Formular wird in drei Gruppen (*fieldsets*) eingeteilt. Jede Gruppe besitzt eine eigene Überschrift (*legend*). Die Gruppen sind durch eine Linie voneinander getrennt.




```
<form>
  <fieldset>
    <legend> <strong> Schuldaten </strong> </legend>
    Schulname: <input type = "text"  name="Schulname"
        value = "Goethe-Gymnasium"  readonly/> <br />
    Ort: <input type = "text"  name="Ort"  value =
            "H&ouml;rde"  readonly/> <br />
  </fieldset>
  <fieldset>
   <legend> <strong> Sch&uuml;ler </strong> </legend>
   Name: <input type = "text"  name="Schuelername"
              size="15"/> <br />
   Vorname: <input type = "text"  name="Vorname"
              size="10"/> <br />
  </fieldset>
  <fieldset>
    <legend> <strong> Lieblingsf&auml;cher </strong>
    </legend>
    Fach 1: <input type = "text"  name="Fach1"  value
                = "Physik"/> <br />
    Fach 2: <input type = "text"  name="Fach2"
              size="20"/> <br />
  </fieldset>
</form>
```

Aufgabe

Erstelle ein Formular, in welches der Benutzer (ein Schüler unserer Schule) mehrere Informationen eintragen soll. Das Formblatt soll „ordentlich“ aussehen (Einsatz von blinden Tabellen?).

Eingabe von Name, Vorname, Klasse

Angabe der Lieblingsfächer. Dabei sind Mehrfachangaben möglich.

Der Schüler soll in freier Form seine Meinung über unsere Schule eintragen können.

Der Schüler soll eine Note für die Qualität seines naturwissenschaftlichen (Physik, Biologie, Chemie zusammengenommen) Unterrichts und analog jeweils eine Note für die Qualität seines fremdsprachlichen und gesellschaftswissenschaftlichen Unterrichts (mit möglichen Noten als Auswahl) eingeben können.

Der Schüler soll mit Radio-Buttons unter 10 möglichen Lehrernamen den „Master of Desaster“ auswählen können.

Der Schüler gibt den Namen seines Lieblingslehrers ein, aber so, dass keine weitere zuschauende Person diesen Namen lesen kann!

Natürlich muss das Formular eine Reset-Möglichkeit haben.

Das Formular soll ein Eingabefeld erhalten, in welches man das angeblich interessanteste Fach in der Schule eingeben kann. Dieses Fach soll als Voreinstellung den Eintrag „Informatik“ haben. Natürlich darf dieser Eintrag nicht überschrieben werden können.

FRAMES

Man kann den Bildschirm in mehrere Bereiche, sog. *Frames* einteilen. Jeder Frame entspricht einer ganz normalen HTML-Datei. Zusätzlich benötigt man eine weitere Datei, in welcher festgelegt wird, wie die Frames auf dem Bildschirm verteilt werden sollen.
Um also ein Frameset mit zwei Frames aufzubauen, benötigt man drei Dateien. Die erste Datei definiert nur das Frameset. Beispiel:

```
<html>
   <head>
      <title> Ein erstes Framebeispiel </title>
   </head>
   <frameset  rows = "20%, 80%"   border = "5">
      <frame  src = "ObererRahmen.HTML" />
      <frame  src = "UntererRahmen.HTML" />
   </frameset>
</html>
```

Wie man sieht, ist in dieser Datei kein *body-Tag* nötig. Auf den *head-Teil* hätte man ebenso verzichten können. Auf dieser Seite wird festgelegt, dass zwei Rahmen untereinander (wie Zeilen, engl. *rows*) angelegt werden. Die Prozentangaben beziehen sich auf die Bildschirmgröße. Alternativ kann man statt der Prozentangaben auch absolute Werte in Pixel angeben:

```
   <frameset  rows = "100, *"   border = "5" >
```

Hierbei bedeutet das Sternchen, dass der zweite, untere Frame sich der restlichen Höhe des Bildschirms anpasst.

Die beiden HTML-Seiten für die beiden Frames brauchen übrigens keinen Titel in ihrem jeweiligen Head-Teil. Dieser würde sowieso nicht angezeigt. Hingegen wird der in der Frame-set-Datei angegebene Titel angezeigt.

Natürlich kann man auch mehr als zwei Frames untereinander anordnen.

Zusätzlich kann man (im *Frame**Set**-Tag*) die Dicke der Framebegrenzungen angeben.

Möchte man die Frames nebeneinander haben, so ersetze man in dem Frameset-Tag die Angabe *rows* durch *cols* (engl. columns = Spalten).

Man kann Frames auch verschachteln. Dafür müsste man die für einen Rahmen vorgesehene HTML-Datei nur durch eine weitere Frameset-Datei ersetzen (zu welcher natürlich dann auch weitere Frame-Dateien gehören).

Normalerweise kann ein Betrachter die Ausmaße eines Frames (durch Ziehen mit der Maus) ändern. Möchte man dies verhindern, so fügt man im *frame-Tag* noch den Befehl *noresize* hinzu. Beispiel:

```
<frame  src = "UntererRahmen.HTML"  noresize>
```

Benachbarte Frames können dann allerdings auch nicht bewegt werden. Bei unsichtbaren Framerahmen (border = "0") ist dieses Attribut jedoch nicht nötig. Hier kann man sowieso nicht die Größe der Frames verändern.

Normalerweise beginnt der Inhalt eines Fensters einige Pixel vom Rand des Fensters entfernt. Diesen Abstand kann man im *frame-Tag* einstellen:

```
<frame  src = "UntererRahmen.HTML"
            marginwidth = "10"   marginheight = "25">
```

Unter *marginwidth* ist damit der linke und rechte Abstand zum Fensterrahmen gemeint, mit *marginheight* wird der obere und untere Abstand bestimmt. Interessant ist dabei natürlich der Einstellungswert `"0".`

Laden von HTML-Seiten in bestimmte Frames hinein

Man kann einen Frame benennen, sodass er als Zielort für andere HTML-Dateien verfügbar ist. Zum Beispiel möchte man manchmal, dass nach einem Mausklick auf einen Link (Verweis) im oberen Rahmen die zugehörige HTML-Seite im unteren Rahmen dargestellt wird. Die Benennung des Zielrahmes erfolgt schon im Frame-Tag (in der Frameset-Seite):

<frame src = "UntererRahmen.HTML" name="Frame_unten">

Nun kann man bei Verweisen zu anderen HTML-Dateien festlegen, dass diese z.B. im unteren Frame angezeigt werden. Dies wird sofort bei der Angabe des Sprungzieles gemacht. Zwei Beispiele:

```
<a  href = "Testseite.HTML"
                target ="Frame_unten">zur Testseite</a>
<a  href = "Klassenfoto.bmp"
                  target = "Frame_unten">zum Bild</a>
```

Fehlt die Zielangabe, so erscheint die Testseite in dem Rahmen, in dem sich auch der Link befindet. Der ursprüngliche Rahmen wird dabei natürlich überschrieben und kann nur mit dem [Zurück]-Pfeil des Browsers wiedergeholt werden.

Gibt man dem Attribut *target* den Wert *_blank*, also ***target = "_blank"***, so wird das Frameset verlassen und die Testseite bzw. das Klassenfoto in einem neuen, großen Browserfenster geöffnet und die alte Seite bleibt im Hintergrund erhalten.

Aufgaben

1. Erstelle eine HTML-Seite, welche zwei Frames untereinander enthält. Jeder Frame soll ein Bild zeigen.
2. Erstelle eine HTML-Seite, welche zwei Frames nebeneinander enthält. Jeder Frame soll ein Bild zeigen.
3. Erstelle eine HTML-Seite, welche aus drei Frames besteht. Die obere Bildschirmhälfte enthält einen Frame, die untere enthält zwei Frames nebeneinander. Jeder Frame soll ein Bild zeigen.
4. Erstelle eine HTML-Seite, welche aus zwei Frames besteht, die nebeneinander dargestellt werden. Im kleineren, linken Frame stehen nur fünf *Links*. Diese *Links* sollen bestimmen, welches Bild im rechten Frame dargestellt wird.

Multimedia

Mithilfe des Einbettungs-Tags *<object>* kann man Multimedia-Dateien in die eigene HTML-Seite integrieren. Bei Verwendung dieses *Tags* laufen die für die Multimedia-Datei benötigten, zusätzlichen Programme (die sog. *PlugIns*) grundsätzlich im Hintergrund ab und die erzeugten Musik- oder Videoausgaben werden direkt in das HTML-Dokument integriert.

Leider kann man nicht davon ausgehen, dass auf dem Rechner des Surfers immer alle notwendigen *PlugIns* installiert sind. So kann es durchaus vorkommen, dass das Abspielen von Multimedia-Dateien z.B. in Firefox funktioniert, im Internet-Explorer aber nicht oder umgekehrt.

Es kann auch durchaus vorkommen, dass eine Multimedia-Datei beim Besucher nicht abgespielt wird, weil in dessen Browser (im Menü *Einstellungen*) das Ausführen von eingebetteten Objekten nicht erlaubt wird.

Dateien, welche Videoclips für Homepages enthalten, werden mit dem Anhängsel *"*.avi"* bezeichnet. Es gibt unterschiedliche Audio-Datei-Formate. Die bekanntesten haben die Datei-Anhängsel *"*.mid*", "*.wav" oder "*.mp3"* .

<u>Kopiere</u> nun eine Multimediadatei in dein Homepageverzeichnis! Gib anschließend sinngemäß eine der beiden folgenden Anweisungen:

```
<object data = "abanda.mid"   height = "100"  width =
     "200" autostart = "false"> </object><br />

<object data = "BlaueNacht.mp3"  height = "100"
    width = "200"  autostart = "false"> </object>
```

Bei dem Besucher deiner Homepage wird daraufhin jeweils ein Hilfsprogramm zum Öffnen dieser Datei gestartet (falls seine Browserinternen Einstellungen gerade dies nicht verhindern). Je nach *PlugIn* erscheint dann z.B. eine Recorder-Tastatur. Anklicken mit der rechten Maustaste eröffnet weitere Möglichkeiten.

Der Autostart-Parameter sorgt dafür, dass die Multimediadatei automatisch gestartet wird. Das wirkt sehr effektvoll beim Aufruf einer HTML-Seite.

Folgende beiden Anweisungen sorgen im Jahr 2013 dafür, dass sowohl im Internet-Explorer als auch in Firefox das entsprechende Musikstück abgespielt wird:

```
<object data = "abanda.mid" height = "100"  width =
"200" autostart = "true"> </object><br />

<object width="200" height="30"
classid="clsid:05589FA1-C356-11CE-BF01-00AA0055595A">
    <param name="filename" value="abanda.mid">
</object>
```

Mit dem Attribut *classid* referenziert man die Implementierung des gewünschten ActiveX-Controls (*classid = class identifier = Klassenbezeichner*). Die Angabe besteht aus der festen Zeichenfolge clsid: - gefolgt von der Bezeichner-ID. Diese ID muss man kennen. Im obigen Beispiel wird ein recht bekanntes ActiveX-Control referenziert, nämlich dasjenige, das unter Windows zum Abspielen von Multimedia-Dateien zuständig ist. Es bindet den Media-Player von Windows in den Bereich des definierten Objekts ein. Mit classid="CLSID:05589FA1-C356-11CE-BF01-00AA0055595A" bindet man also ein ActiveX-Control ein, das es erlaubt, Sound- und Videodateien aller bekannten Formate wie WAV, AU, MID, MP3, AVI, MPEG usw. abzuspielen.
Dies funktioniert allerdings nur im Internet-Explorer.

Im *object-Tag* kann man auch noch bestimmen, wie oft z.B. ein Musikstück wiederholt werden soll:
<object..... loop="false">

Statt *loop="false"* könnte man auch *loop="true"* oder *loop* = „3“eingeben.

Es lassen sich auch Verweise auf alle derartigen Multimediadateien anlegen. Diese werden von allen Browsern richtig erkannt. Beispiel:

<a href="Musikdateititel.mid"> Musik ? </A>
Hierbei öffnet sich automatisch das zum Abspielen vorhandene PlugIn auf dem Rechner des Besuchers.

Meta-Tag

Als Metainformationen bezeichnet man allgemein Daten, die Informationen über andere Daten enthalten. Alle Metatags haben ihren Platz im HTML-Code zwischen **<head>** und **</head>**. Damit setzt man u.a. Informationen für Suchmaschinen, z.B. Angaben über den Verfasser der Seite, letztes Bearbeitungsdatum oder Schlüsselwörter im Text.

```
<meta name="author" content="Anna Lyse">
<meta name="date" content="2016-07-16T08:39:23+12:00">
```

Durch diese Angabe wird der Autor des Dokuments ausgewiesen. Des Weiteren wird in einer standardisierten Form ein Datum angegeben.

```
<meta name="description" content="Dieses Dokument
enthält Informationen über die wichtigsten
Schulfächer überhaupt!">

<meta name="keywords" content="Physik, Mathematik,
Informatik">
```

Wenn man ein HTML-Dokument zwar im Web anbieten, aber trotzdem verhindern möchte, dass dieses Dokument über öffentliche Suchdienste auffindbar sein soll, kann man eine entsprechende Anweisung als Meta-Information für Suchprogramme notieren:

```
<meta name="robots" content="noindex, nofollow">
```

Mit **<meta name="robots" content="noindex">** verbietet man einem Suchprogramm, Inhalte aus der HTML-Datei an seine Suchdatenbank zu übermitteln (*robots = Suchprogramme, noindex = keine Indizierung*).

Mit **<meta name="robots" content="nofollow">** kann man einem Suchprogramm verbieten, Hyperlinks, die in diesem Dokument vorhanden sind, zu folgen (*nofollow = nicht folgen*). Inhalte aus der aktuellen HTML-Datei darf es jedoch an seine Suchdatenbank übermitteln. Diese beiden Angaben kann man auch so wie im obigen Beispiel kombinieren.

Meta-Tags können auch auch Informationen für den Browser beinhalten:

Mit der folgenden Angabe wird der europäische Zeichensatz definiert. Durch die Angabe kommen die Browser und Suchmaschinen auch mit Umlauten zurecht.

```
<meta charset="utf-8" />
```

Automatische Weiterleitung

Mit Hilfe eines Meta-Tags im *Head*-Teil einer HTML-Datei kann man dafür sorgen, dass die aufgerufene HTML-Datei nach einstellbarer Zeit automatisch eine andere HTML-Datei aufruft. Natürlich kann man beliebig viele Dateien derart miteinander verketten. Insbesondere mit Hilfe von Frames kann man so auch eine Dia-Show organisieren.

Zwei HTML-Dateien sollen sich gegenseitig aufrufen. Die erste sei *Test1.html*, die zweite sei *Test2.html* . Die Abkürzung ***url*** bedeutet ***U****niform* ***R****essource* ***L****ocator* und gibt die genaue Adresse einer Datei an.

Der Inhalt von *Test1.html* sei

```
<html>
   <head>
      <title> erste Seite </title>
      <meta  http-equiv="refresh"  content="3;
           url=test2.HTML">
   </head>
   <body bgcolor="#FF0000">
      Dies ist Seite 1
   </body>
</html>
```

Der Inhalt von *Test2.html* sei

```
<html>
   <head>
      <title> zweite Seite </title>
      <meta  http-equiv="refresh"  content="6;
           url=test1.HTML">
   </head>
   <body bgcolor="#00FF00">
      Dies ist Seite 2
   </body>
</html>
```

Natürlich kann man auch auf eine Seite im Internet weiterleiten, zum Beispiel mit der Anweisung

```
<meta http-equiv="refresh" content="6; url=http://stern.de">
```

Der oben gezeigte *Meta-Tag* ruft nach einer Verweildauer, welche durch die Zahl hinter *content* in Sekunden festgelegt wird, automatisch die Seite auf, deren Adresse hinter *url* steht. **Beachte die etwas ungewöhnliche Stellung der Anführungszeichen!**
Wird die Wartezeit auf 0 gesetzt, so erfolgt die Weiterleitung sofort. Dies ist besonders problematisch: Nutzt der Besucher nämlich die Zurück-Navigations-funktion (Back-Button), wird er direkt wieder nach vorne katapultiert. Somit hängt er auf einer Seite fest. Das verärgert den Benutzer natürlich sehr!

Aufgabe:
Erstelle eine kleine Dia-Show, indem du 5 verschiedene HTML-Seiten mit jeweils einem Bild erstellst. Nach jeweils 3 Sekunden soll immer die nächste Seite aufgerufen werden. Nach dem letzten Bild bleibt die entsprechende Seite stehen. Die letzte Seite enthält zusätzlich einen Link, mit dessen Hilfe man auf die Startseite deiner Homepage zurückspringen kann.

Datei von Original-Adresse laden.
Jeder Browser speichert die aufgerufenen HTML-Seiten in einem speziellen Speicherbereich mit schnellem Zugriff auf dem lokalen Rechner. Dieser Speicherbereich wird Browser-Cache genannt. Web-Seiten werden im Web auf so genannten Proxy-Servern zwischengespeichert. Das ist dann ein so genannter Proxy-Cache. Ein Nachteil ist jedoch, dass dem Anwender möglicherweise Daten angezeigt werden, die gar nicht mehr aktuell sind, weil auf der Originaladresse mittlerweile neue Daten liegen. Beispielsweise ändert sich im Laufe eines Vormittags häufig der auf einem Rechner angezeigte Vertretungs-plan für Schüler. Da wäre es schlecht, wenn den ganzen Tag lang nur die allererste Version dieses Vertretungsplanes gezeigt werden würde.
Man kann mit Hilfe einer Meta-Angabe erzwingen, dass die Daten nicht aus einem Cache-Speicher geholt werden, sondern von der Original-Datei aus.

Mit `<meta http-equiv="expires" content="0">` wird veranlasst, dass diese HTML-Datei in jedem Fall von der Originaladresse geladen wird (*expires = fällig werden*).

Cascading Style Sheets (CSS)

HTML ist für die Struktur eines Dokumentes zuständig - CSS dagegen ist für das Aussehen (Design) verantwortlich. CSS regelt die Darstellung von HTML-Elementen: CSS-Eigenschaften beschreiben die Schriftfamilie, Schriftgröße und -farben, die Größe und den Hintergrund von Elementen und die Platzierung von Elementen im Browserfenster. Damit überschreibt und erweitert CSS die Darstellung von HTML-Elementen.

Der Aufbau einer CSS-Anweisung ist folgendermaßen:

```
Selektor  {
     Eigenschaft 1: Wert;
     Eigenschaft 2: Wert;
     usw.
}
```

Wichtig: Jede Deklaration (= Eigenschaft-Wert-Kombination) muss mit einem Semikolon abgeschlossen werden. Das Semikolon ist also kein Trennzeichen sondern ein Abschlusszeichen.
Der *Selektor* ist das jeweilige HTML-*Tag*, das in CSS ohne die *Tag*-Klammern geschrieben werden muss.

CSS-Anweisungen kann man im Head-Teil unterbringen zwischen den *Tags* **<style>** und **</style>** . Beispiel:

```
<style>
  h1 {
      color: red;
      background-color: black;
  }

  p {
      max-width: 500px;
  }
</style>
```

Obige Anweisung würde u.a. bewirken, dass sämtliche h1-Überschriften in diesem Dokument in roter Schrift auf schwarzem Hintergrund erfolgen würden. Beachte: Keine Anführungszeichen!

Eigenschaften und ihre Werte werden immer klein geschrieben. Wenn der Selektor ein HTML-Tag-Selektor ist wie in diesem Beispiel, wird der Selektor ebenfalls immer klein geschrieben. Wenn ein Wert eine Maßangabe enthält wie *width: 500px* in diesem Beispiel, wird die Maßeinheit **ohne Leerzeichen** direkt hinter den Wert gesetzt.

Falls obige Farbwahl nur für eine einzige, ganz bestimmte Überschrift gelten soll, so schreibt man diese Farbwahl am besten nur in die betreffende Überschrift mit hinein. Die komplette CSS-Information für diesen Tag wird dabei in Anführungszeichen geschrieben:

```
<h1 style="color: red; background-color: black;">
Überschrift</h1>
```

Man spricht in diesem Fall auch von *Inline-stylesheet*.

Falls CSS-Anweisungen sehr umfangreich sein sollen, so lagert man die CSS-Anweisungen üblicherweise in eine eigene Datei (mit beliebigem Namen) aus. Das *style-Tag* wird in einer externen Style-Sheet-Datei **nicht** notiert. Zusätzlich muss man allerdings im Head-Teil einer jeden HTML-Seite ankündigen, dass diese externe Seite eingebunden werden muss. Im Folgenden wird diese Seite *design.css* genannt:

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

Farben mit CSS

Für Farben werden entweder die englischen Namen (white, gray, black …) verwendet oder sie werden in hexadezimaler Schreibweise (z.B. für rot = #FF0000) oder als dezimaler RGB-Code, z.B. für rot = `rgb(255,0,0)`, angegeben. Beachte die Raute vor der Hexadezimalzahl!

Möchte man nun in CSS einem Text eine Farbe zuweisen, gibt es die Anweisung: `color: Farbwert;`

Natürlich kann auch eine Hintergrundfarbe zugewiesen werden. Das klappt bei den meisten Elementen wie z.B. Überschriften, Absätzen, Aufzählungen und DIV- bzw. SPAN-Bereichen.
Der Aufbau des Befehls für die Hintergrundfarbe in CSS lautet:
`background-color: Farbwert;`

Aufgabe: Erzeuge nachfolgende Ausgabe!
Hinweis: am einfachsten ist es, die gesamte Ausgabe mit einem einzigen div-Bereich zu realisieren.

Auffallend ist, dass die Hintergrundfarbe hier den gesamten Bereich bis nach rechts füllt. Hier sieht man die Auswirkung von Block-Elementen.

Aufgabe:
Bearbeite den folgenden Quellcode so, dass für die fett hervorgehobenen Bereiche über den HTML-Befehl **** die Hintergrundfarbe gelb sein wird. Alle kursiven Textstellen **<i>** sollen hellgrün dargestellt werden.

```
<p><strong>Naturwissenschaften sind toll.</strong>
Sie ergänzen sich <i>super</i>. Mit <i>Farbe</i> wird
alles <strong>interessanter!</strong> </p>
```

Rahmen mit CSS

Für jedes Element kann ein Rahmen angezeigt werden. Ein Rahmen benötigt folgende 3 Angaben, damit dieser angezeigt wird:

- Rahmenfarbe
- Strichstärke des Rahmens
- Art des Rahmens (z.B. durchgängig oder gepunktet usw.)

Für diese 3 Rahmen-Angaben gibt es die entsprechenden CSS-Anweisungen:

- border-color
- border-width
- border-style

Man sollte immer alle 3 Werte angeben. Ohne die Angabe der Rahmenfarbe wird i.d.R. weiß als Rahmenfarbe genutzt. Und ein weißer Rahmen auf weißem Hintergrund kann einen schon in den Wahnsinn treiben, wenn man ihn sucht. Durch folgende Angaben erhält man also einen grünen Rahmen:

```
border-color: green;
border-width: 5px;
border-style: solid;
```

Als Kurzschreibweise wird auch Folgendes akzeptiert:
`border: green 5px solid;`

Hier sind alle 3 Angaben (in beliebiger Reihenfolge) notwendig – zur Trennung der einzelnen Werte wird einfach ein Leerzeichen gemacht.

CSS bietet sehr viele verschiedene Längeneinheiten. Es gibt z.B. aus der Buchdruckerkunst hergeleitete Einheiten wie point (**`pt`**) und pica (**`pc`**), andere sind bekannter, so wie centimeter (**`cm`**), inch (**`in`**) oder Prozent (**`%`**). Prozentangaben beziehen sich immer auf die Größe des sog. Containers. Speziell für CSS wurde die Einheit **`px`** erfunden. Diese Einheit **`px`** wurde definiert um klein, aber noch sichtbar zu sein. Sie hängt teilweise von der verwendeten Hardware ab.
Eine relative Längeneinheit ist **`em`** , welche von der Größe der Schriftart abhängt.
Das Verhältnis der absoluten Einheiten ist wie folgt:
1in = 2.54cm = 25.4mm = 72pt = 6px
Es gibt keine Einschränkung, welche Einheit wo verwendet werden kann.

Für die Wahl der Rahmenart gibt es u.a. folgende Möglichkeiten:

- solid = durchgezogen
- double = doppelt
- none = kein Rahmen (unsichtbar)
- hidden = kein Rahmen (unsichtbar)
- dotted = gepunktet
- dashed = gestrichelt
- groove = 3D-Effekt
- ridge = 3D-Effekt
- inset = 3D-Effekt
- outset = 3D-Effekt

Überschrift Nummer 1

Dies ist der erste Abschnitt

Überschrift Nummer 2

Man kann auch die vier Seiten eines Rahmens einzeln festlegen. Dies geschieht durch die Angabe der Position des Rahmens: top, right, bottom und left.

```
border-top-style: ...;
border-right-style: ...;
border-bottom-style: ...;
border-left-style: ...;
```

Analoges gilt auch für die Attribute **`color`** und **`width`**. Oder, besser noch in Kurzschreibweise: **`border-top: green 20px solid;`**

Farbe und Schriftartgestaltung

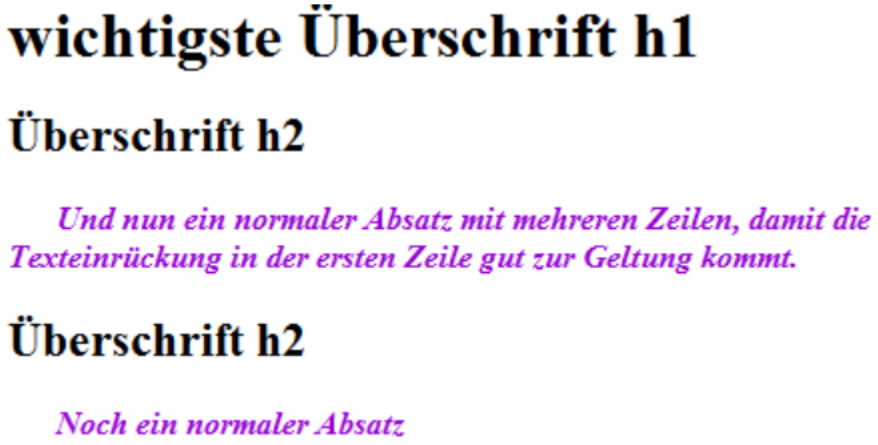
Für die Gestaltung der Schrift werden folgende CSS-Befehle benötigt:

`color:#FF9F00;`
Farbe: orange, kann entweder als Hex-Wert:"#FF9F00" oder als (englischer) Farbname angegeben werden: "orange", also "color:orange;"
`font-size:28pt;`
Schriftgröße: hier 28pt
`font-family:arial, "lucida console", sans-serif;`
Schriftart: wenn vorhanden Arial, wenn nicht vorhanden, dann "lucida console" und wenn diese nicht vorhanden, dann eine vorhandene serifenlose Schrift.
`font-weight:bold;`
Die Schriftstärke, zur Auswahl stehen normal|bold|bolder|lighter

`font-style:italic;`
italic = kursive
oblique = schräggestellt
normal = normal

`text-indent:1.5cm;`
Einrückung der ersten Zeile eines Textblockes um den vorgegebenen Abstand.

Aufgabe: erstelle folgende Ausgabe:



Beachte, dass in obiger Ausgabe alle Absätze gleich formatiert sind!

Zeilenhöhe und Abstände
`line-height:1.5em;`
Bei der Angabe der Maßeinheit ist die Wahl einer relativen Maßeinheit (em) sinnvoll, damit bei Änderung der Schriftgröße der Zeilenabstand automatisch passend angezeigt wird. Daher wird im Beispiel mit 1.5em gearbeitet. Bitte den Dezimalpunkt beachten!
`letter-spacing:0.3em;`
Der Abstand zwischen den einzelnen Buchstaben sollte relativ angegeben werden, also in em

`word-spacing:0.5em;`
Der Abstand zwischen den einzelnen Wörtern sollte relativ angegeben werden, also in em

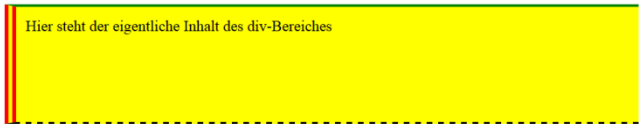
Box-Modell

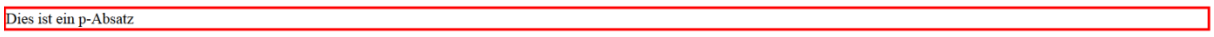
Für jedes Element wird eine rechteckige Fläche, eine sog. box (engl. Box = Schachtel), in CSS reserviert.

Für jede Box lassen sich folgende Eigenschaften festlegen:

- **Inhalt:** für den Inhalt kann eine Breite (**`width`**) und eine Höhe (**`height`**) definiert werden
- **Innenabstand (`padding`):** definiert den Platz zwischen Inhalt und Rahmen
- **Hintergrundfarbe (`background-color`):** Es kann eine Hintergrundfarbe festgelegt werden
- **Hintergrundbild (`background-image`):** Es kann ein Hintergrundbild festgelegt werden, das die Hintergrundfarbe überdeckt
- **Rahmen (`border`):** dem Rahmen kann eine Stärke mitgegeben werden, die Strichart und eine Farbe (der Rahmen kann auch unsichtbar sein)
- **Außenabstand (`margin`):** Abstand zu anderen Elementen

Im Folgenden wird ein div-Bereich entsprechend eingerichtet:






```
<div style="background-color:yellow;
            width:  400px;
            height: 100px;
            border-top: 3px green solid;
            border-right: 2px red hidden;
            border-bottom: 2px black dashed;
            border-left: 12px red double;
            padding: 10px;
            margin: 60px;
">

Hier steht der eigentliche Inhalt des div-Bereiches
</div>
```

```
<p style="border:3px red solid; ">Dies ist ein p-
Absatz </p>
```

Genauso wie beim Rahmen können für den Außenabstand verschiedene Abstände definiert werden:

- **margin-top: Wert;**
- **margin-right: Wert;**
- **margin-bottom: Wert;**
- **margin-left: Wert;**

Genauso können für den Innenabstand verschiedene Abstände definiert werden:

- **padding-top: Wert;**
- **padding-right: Wert;**
- **padding-bottom: Wert;**
- **padding-left: Wert;**

In einer verkürzten Form können auch alle 4 Seiten definiert werden:
"margin:20px 35px 15px 60px;"
Das bedeutet: 20px für oben, 35 px rechts, 15px unten, 60px links - gelesen wird wie die Uhrzeit, man fängt oben an.

Die verkürzte Schreibweise funktioniert sowohl für:

- margin
- padding
- border

CLASS und ID - Bezeichner für CSS-Elemente

Eine Klasse ist in CSS einfach gesagt eine Gruppierung von Eigenschaften, die einmal definiert wird und in der Regel mehrmals, sogar für unterschiedliche Html-Elemente, verwendet wird.
Im folgenden Beispiel wird eine rote Schrift auf gelbem Hintergrund als eigene Klasse definiert und für die beiden Html-Elemente **** und **<i>** angewandt:

Das wichtigste Fach in der Schule ist **Informatik** , na klar!
Aber auch *Mathematik* ist nicht schlecht!

```
<html>
  <head>
    <style>
      .rotaufgelb {
          background-color: yellow;
          color: red;
      }
    </style>
  </head>
  <body>
    <br /><br />
    Das wichtigste Fach in der Schule ist
    <strong class="rotaufgelb"> Informatik </strong>,
    na klar!<br />
    Aber auch <i class="rotaufgelb">Mathematik </i>
    ist nicht schlecht!
  </body>
</html>
```

Hinweis: für Klassen wird in der CSS-Definition vor dem Namen ein Punkt geschrieben. Und die Namen sind „case sensitiv“ – sprich Groß- und Kleinschreibung macht einen Unterschied. Also am besten alles klein schreiben.

Im Gegensatz zu Klassen können ID-Bezeichner nur einem einzigen HTML-Element zugeordnet werden. Für IDs wird in der CSS-Definition vor dem Namen eine Raute geschrieben. Auch hier sind die Namen „case sensitiv“ – sprich Groß- und Kleinschreibung macht einen Unterschied.

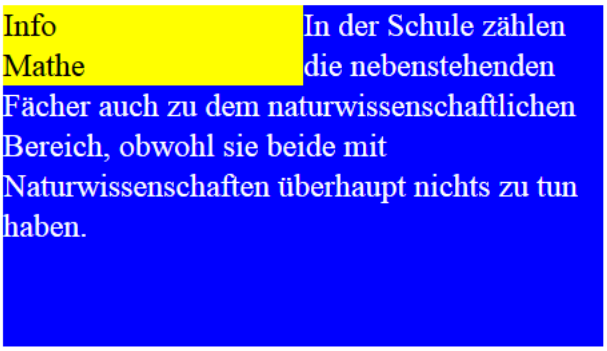
Das Fach **Musik** ist auch ganz nett!

```
<html>
  <head>
    <style>
```

```
        #weissaufblau {
           background-color: blue;
           color: white;
        }
    </style>
  </head>
  <body>
    <br />
    Das Fach
    <strong id="weissaufblau"> Musik </strong>
    ist auch ganz nett!<br />
  </body>
</html>
```

Man kann einem HTML-Element auch gleichzeitig sowohl eine Klasse als auch eine ID zuordnen.

Die CSS-Anweisung float



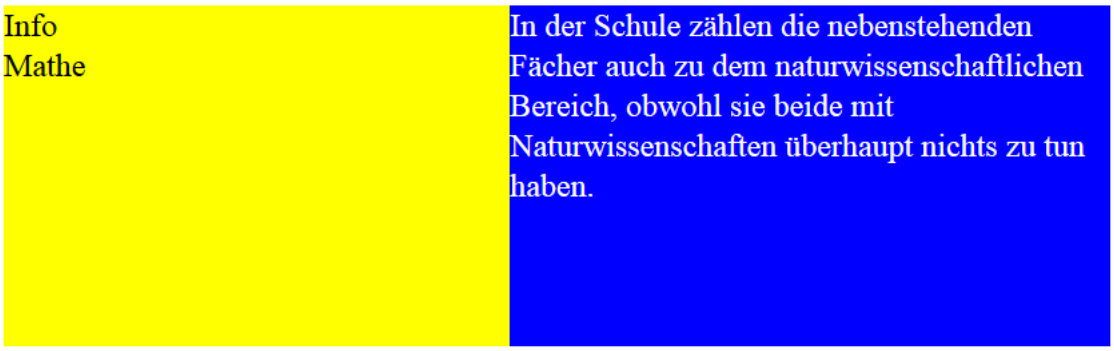
Vom Sinn her sind beide Fächer reine Geisteswissenschaften.

```
<html>
  <head>
    <meta http-equiv="content-type"
       content="text/html; charset=utf-8" />
    <style>
      #Absatz1 {
        float:left;
        background-color:yellow;
        width: 150px;
      }

      #Absatz2 {
         background-color:blue;
         color: white;
         width:  300px;
         height: 170px;
      }
    </style>
  </head>

  <body>
    <br />
    <div id="Absatz1">
      Info <br />
      Mathe
    </div>
    <div id="Absatz2">
       In der Schule zählen die nebenstehenden Fächer
       auch zu dem naturwissenschaftlichen Bereich,
       obwohl sie beide mit Naturwissenschaften
       überhaupt nichts zu tun haben.
    </div>
```

```
    Vom Sinn her sind beide Fächer reine
    Geisteswissenschaften.
  </body>
</html>
```

Wenn man den Abstand des zweiten Absatzes zum linken Seitenrand festlegt, so lässt sich leicht eine 2-Spalten-Darstellung realisieren:

```
<style>
  #Absatz1 {
     float:left;
     background-color:yellow;
     width: 20%;
     height: 170px;
  }

  #Absatz2 {
     background-color:blue;
     color: white;
     width:  300px;
     height: 170px;
     margin-left: 20%
  }
</style>
```

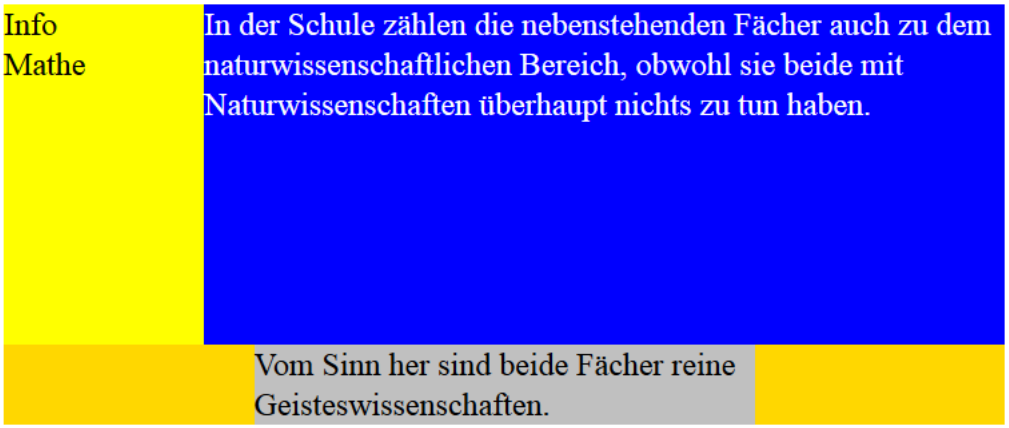


Vom Sinn her sind beide Fächer reine Geisteswissenschaften.

Mit der folgenden CSS-Anweisung kann man einen Bereich auch zentrieren. das auto besagt, dass auf beiden Seiten der Platz automatisch aufgeteilt wird.

```
#Absatz3  {
   background-color:silver;
   width:300;
   margin: 0 auto;
}
```

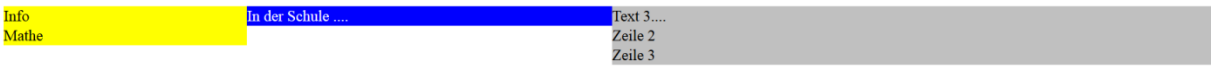
Umschließt man alle drei Bereiche mit einem weiteren, sog. Elternbereich, so lässt sich auch Folgendes realisieren:




```
<html>
  <head>
    <meta http-equiv="content-type"
        content="text/html; charset=utf-8" />
    <style>
      #Absatz0  {
         background-color:gold;
         width:500;
      }

      #Absatz1 {
        float:left;
        background-color:yellow;
        width: 20%;
         height: 170px;
      }

      #Absatz2 {
         background-color:blue;
          color: white;
         width:  400px;
         height: 170px;
          margin-left: 20%
      }

      #Absatz3  {
         background-color:silver;
         width:250;
         margin: 0 auto;
      }
    </style>
```

```
  </head>

  <body>
    <br />
    <div id="Absatz0">
      <div id="Absatz1">
        Info <br />
        Mathe
      </div>

      <div id="Absatz2">
        In der Schule …….
      </div>

      <div id=Absatz3>
         Vom Sinn …..
      </div>
    </div>
  </body>
</html>
```

3-spaltiges Layout in CSS



Äußerst wichtig bei dem zugehörigen HTML-Code ist, dass man im <body>-Teil zuerst den rechtsbündigen Absatz schreibt und danach erst den mittleren Absatz! Beachte auch, dass die margin-Angaben im Uhrzeigersinn jeweils den Abstand zum Seitenrand bestimmen!

```
<html>
  <head>
    <meta http-equiv="content-type"
        content="text/html; charset=utf-8" />
    <style>
      #Absatz1 {
        float:left;
        width:20%;
        background-color:yellow;
      }

      #Absatz2 {
        margin:0 50% 0 20%;
        background-color:blue;
        color:white;
      }

      #Absatz3  {
        float:right;
        width:50%;
        background-color:silver;
      }
    </style>
  </head>

  <body>
    <div id="Absatz1">
      Info <br />
      Mathe
    </div>

    <div id="Absatz3">
      Text 3....<br />
      Zeile 2<br />
```

```
      Zeile 3
    </div>

    <div id="Absatz2">
       In der Schule ....
    </div>

  </body>
</html>
```

Hintergrundbilder

Im HTML-Code kann das Bild eine ID bekommen, wenn die Einstellungen nur für dieses Bild gelten sollen oder eine Klasse (class), wenn man mehrere Bilder mit derselben Größe, Ausrichtung und Eigenschaften hat.

Im folgenden Beispiel wird das Bild durch die CSS-Definition rechts ausgerichtet, mit einem Abstand nach links von 20 Pixel und dass der Text das Bild umfließt.

```
<style>
  #rechts {
    float:right;
    margin-left: 20px;
    border:1px solid red;
 }
</style>
…..

<img id="rechts" src="bilder/meineFreundin.jpg" />
Hier kommt Text, der das Bild links umfließt und
automatisch umgebrochen wird.
…..
```

Um das Bild in den Hintergrund zu packen, gibt es mehrere Möglichkeiten. Die weit verbreiteste Möglichkeit ist, es in der CSS-Datei einen Bereich zu Definieren, der für das komplette HTML-Dokument gilt - also der Tag html bzw. body.

Beispiel: Das Bild wird nun gesetzt und wiederholt sich automatisch.

```
<style>
  html, body {
    background:url(bilder/meineFreundin.jpg);
  }
</style>
```

Im Folgenden wird das Bild rechts oben (right top) gesetzt und wiederholt sich nicht:

```
<style>
  html, body {
    background:url(bilder/meineFreundin.jpg)
      no-repeat right top;
  }
</style>
```

Absolute Positionierung

Im Folgenden wird ein Absatz exakt positioniert:

```
<style>
  #kasten {
    background-color: orange;
    position: absolute;
    width: 250px;
    height: 175px;
    top: 150px;
    left: 0;
  }
</style>
…..
<div id="kasten">…….</div>
```

Die Angaben *top* und *left* geben den Abstand des Bereiches zum oberen bzw. zum linken Bildschirmrand an. Natürlich kann man genauso gut Angaben für *bottom* bzw. *right* geben.
Bei dem Wert 0 muss nicht unbedingt eine Maßeinheit mitgegeben werden.