« htmlcss.pdf »

HTML und CSS

Eine Einführung

von Herbert Paukert

Version 24.a - 2024

Dieses Werk ist urheberrechtlich geschützt.

Kopien daraus bedürfen der Einwilligung des Autors.

Eigenverlag

www.paukert.at

In dieser Einführung werden ausgewählte Elemente und Befehle
aus der mächtigen Vielfalt von HTML und CSS dargestellt . . . . .

Teil A: HTML-CSS-Kurzanleitung

WWW-Internetseiten werden mittels **HTML** (HyperText Markup Language, einer eigenen Skript-Sprache) erstellt. Ein HTML-Dokument enthält neben dem eigentlichen Text Steuercodes (Tags), welche die unterschiedlichen Elemente einer Seite, wie Zeichen- und Absatzformate, Überschriften, eingefügte Grafiken oder Verweise (Links) definieren. Jene Elemente, die man hervorheben möchte, markiert man mit solchen Steuercodes. Entsprechende Programme, so genannte WEB-Browser, die ein HTML-Dokument geliefert bekommen, erkennen und interpretieren diese Steuerzeichen dann richtig.

Zur Erstellung von HTML-Seiten genügt ein einfacher Texteditor wie **NotePad** von Microsoft oder auch **HtmlEdit** vom Autor. Dabei werden alle Zeichen (auch die deutschen Umlaute und das scharfe ß) in einen erweiterten ASCII-Code kodiert, was durch die Anweisung <META charset="ISO-8559-1"> bewirkt wird. Verwendet man jedoch komplexere Editoren wie **NotePad++** oder **EditPadLite** werden entsprechend der Anweisung <META charset="UTF-8"> alle Zeichen im Unicode kodiert. Im Gegensatz zum ASCII-Code, der nur 1 Byte für jedes Zeichen besitzt, stehen jetzt für jedes Zeichen 2 Bytes zur Verfügung. Mit diesen 256*256 = 65536 Bytes können alle Sprachen der Welt kodiert werden. Um dabei die deutsche Sprache zu erhalten muss am Anfang das Tag <html lang = "DE"> geschrieben werden. Manche Editoren stellen dem eigentlichen Unicode noch drei Erkennungsbytes voran (BOM = Byte Order Mark).

HTML-Befehle (Tags) werden durch spitze Klammern markiert. Fast alle Befehle bestehen aus einem **einleitenden** und einem **abschließenden** Tag. Das einleitende Tag kann außer dem Befehl noch Attribute und Werte enthalten. Dem abschließenden Tag wird ein Slash (/) vorangestellt, es markiert das Ende des Befehls. Groß- und Kleinschreibung wird bei HTML-Tags nicht unterschieden und mehrfache Leerzeichen werden ignoriert. Der grundlegende Aufbau einer HTML-Seite sieht folgendermaßen aus:

```
<!DOCTYPE html>
<HTML> oder <HTML lang="DE">
<HEAD>
<TITLE> . . . . . </TITLE>                                  beschreibt den Seiteninhalt
<META charset="ISO-8859-1">                                 älterer Zeichensatz (mit Umlauten)
   oder
<META charset="UTF-8">                                      bestimmt den Zeichensatz „Unicode“
<META NAME="description" CONTENT="Beschreibung">            beschreibt den Seiteninhalt
<META NAME="author" CONTENT="Autor">                        nennt den Autor
<META NAME="keywords" CONTENT="Stichwörter">                bezeichnet Stichwörter
<META NAME="viewport" CONTENT="width=device-width, initial-scale=1.0">
                                                            responsive Screen-Anpassung
```

nicht sichtbarer Dokumenten-Kopf

```
</HEAD>

<BODY>
```

sichtbarer Dokumenten-Körper

```
</BODY>
</HTML>
```

Die META-Anweisungen im Dokumenten-Kopf dienen der allgemeinen Beschreibung der HTML-Seite. Der Inhalt des Dokumenten-Körpers kann aus Texten, Bildern, Sounds und Videos bestehen. Diese Elemente werden in ihrem Erscheinungsort (position, left, top), in ihrer Größe (size, width, height) und in ihrer Farbe (color) durch einschlägige Befehle bestimmt.

Unterstützt wird **HTML** dabei durch die besondere Auszeichnungsprache **CSS** (Cascading Style Sheets). Deren Befehle bestimmen die Art und Weise (style) der Erscheinung der HTML-Elemente. Die Angabe von Stilanweisungen kann im Dokumenten-Kopf zwischen den Tags **<style>** und **</style>** stehen.

Neben CSS kann noch **JavaScript** in eine HTML-Seite eingebunden werden. Das ist eine eigenständige Programmiersprache, welche auf die HTML-Objekte und deren Styles zugreifen und sie auch verändern kann. Die JavaScript-Befehle müssen immer zwischen den Tags **<script>** und **</script>** stehen.

Das Dokument-Objekt-Modell (DOM)

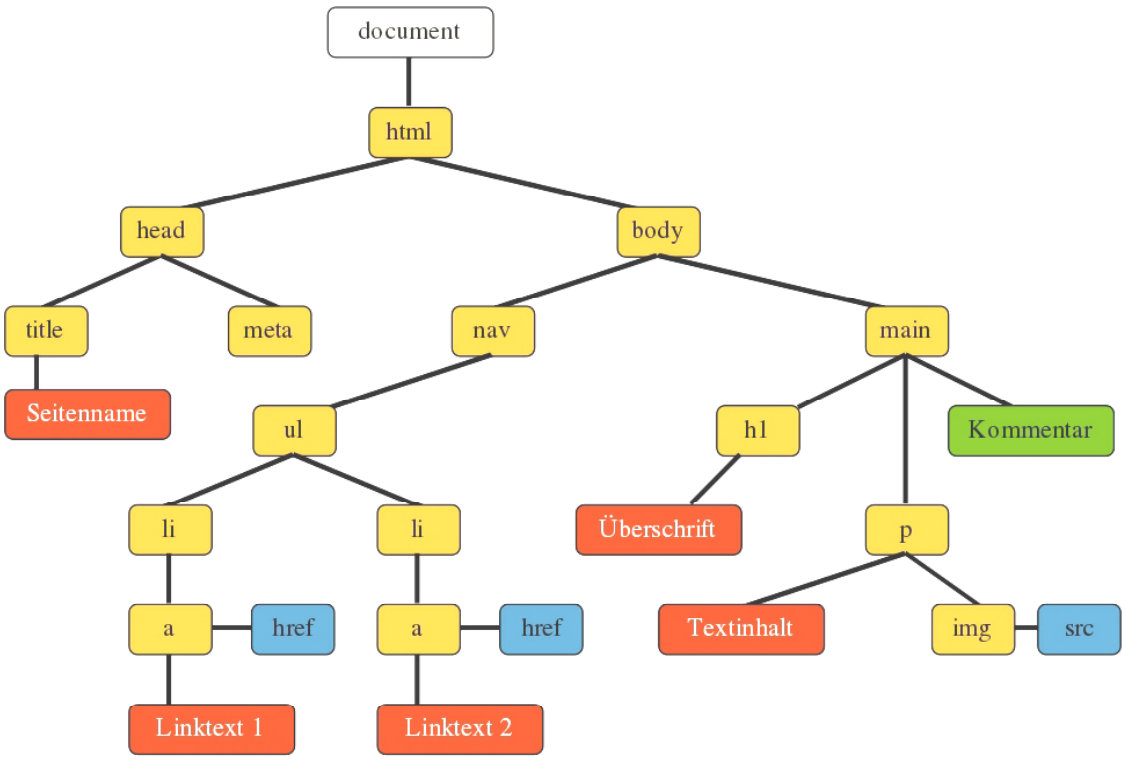
Ein Web-Browser auf dem Rechner des Benutzers (Client) stellt eine Verbindung mit dem Rechner des Anbieters (Server) über einen Internet-Link her und schickt an den Webserver eine Anfrage. Der Webserver antwortet mit einer Empfangsbestätigung und sendet an den Client ein HTML-Protokoll zurück.

Weil JavaScript im Web-Browser des Client integriert ist und auf dessen Rechner dann ausgeführt wird, ist JavaScript eine clientseitige Programmiersprache. Davon zu unterscheiden sind die serverseitigen Programmiersprachen (beispielsweise PHP), die nur am Server-Rechner laufen.

Der erhaltene HTML-Code liegt im Browser zunächst nur als Text vor. Während der Browser über das Netz den Code empfängt, wird durch einen so genannten Parser der Code schrittweise verarbeitet, d.h. in eine bestimmte Speicherstruktur im Arbeitsspeicher des Rechners übergeführt. Diese Speicherstruktur besteht aus zusammenhängenden Bündeln von Informationen, welche auch Objekte genannt werden. Die verschiedenen Objekte bilden eine Struktur, welche mit einem Wurzelknoten beginnt und sich dann baumartig zu weiteren Element-Knoten verzweigt. Der Aufbau dieser Objektstruktur ist durch das Document-Object-Model (DOM) geregelt. Das oberste Element (Wurzelknoten, root) trägt den Namen „***document***“, welches das HTML-Dokument im Browserfenster abbildet. Das „***document***“ enthält nun weitere Objekte: Zunächst „***HTML***“, „***HEAD***“, „***BODY***“, und dann beispielsweise Textfelder, Listen, Schaltflächen (buttons), Bilder (images), usw. Jedes Objekt besitzt bestimmte Eigenschaften (Attribute) und bestimmte Methoden (Funktionen). Die Attribute von übergeordneten Objekten werden auf untergeordnete Objekte vererbt.

An den einzelnen Objekten kann der Benutzer bestimmte Ereignisse (events) auslösen, beispielsweise mit der Maus auf einen Schalter klicken (click) oder auf der Tastatur eine Taste drücken (keypress).

Die Programmiersprache JavaScript kann nun auf die verschiedenen Objekte zugreifen und auch mögliche Ereignisse registrieren und mit bestimmten Unterprogrammen (Funktionen) darauf reagieren. Diese Behandlung von Ereignissen (event handling) ermöglicht erst die Interaktion von Benutzer und HTML-Dokument.



Schematische Darstellung eines DOM-Baumes mit HTML-Objekten.

Die wichtigsten HTML Elemente

Allgemeine Elemente	
<html> </html>	Erzeugt ein HTML Dokument
<head> </head>	Für Meta Informationen, z.B. Seitentitel.
<title> </title>	Name der Website. Wird im Browser oben angezeigt.
<body> </body>	Definiert den sichtbaren Teil der Website.
Formatierung	
	Für Absätze (Paragraphen). Nach einem Absatz wird automatisch ein Zeilenumbruch eingefügt.
	Definiert einen Abschnitt.
</ span>	Definiert einen Abschnitt in einer Zeile.
 	Zeilenumbruch.
<hr>	Fügt eine horizontale Linie ein.
Text	
 ... 	Fügt Überschriften ein. H1 ist die wichtigste (größte) Überschrift, H6 ist die kleinste Überschrift.
** **	**Bestimmt die Schriftart.**
** **	**Bestimmt die Größe des Textes.**
** **	**Bestimmt die Farbe des Textes.**
	Hebt Text **fett** hervor.
	Hebt Text **fett** hervor.
	Hebt Text *kursiv* hervor.
<u > </u >	Stellt Text unterstrichen dar.
Bilder	
<img>	Fügt ein Bild ein. Es muss immer das Attribut **src="..."** angegeben werden. Das Attribut **alt="..."** sollte angegeben werden. Die Größe kann über die Attribute **width="..."** und **height="..."** definiert werden.
Links	
<a> 	Fügt einen Link ein. Es muss immer das Attribut **href="..."** angegeben werden. Wenn ein Link in einem neuen Fenster geöffnet werden soll, kann man das Attribut **target="_blank"** nutzen.
Listen	
	Unsortierte Liste mit Markierungszeichen.
	Sortierte Liste mit Nummerierung.
	Linienelement zwischen Start-Tag und Ende-Tag einer Liste.

Tabellen

<table border="1">	Beginn einer Tabelle (nur zum Testen immer border="1" setzen)
<thead>	Bereich für den Kopf beginnt
<tr>	hier fängt die erste Reihe für die Kopfzeile an
<th>Überschrift</th>	hier fängt die erste Zelle für die Kopfzeile (Überschriften für Spalte)
<th>Ü. Spalte 2</th>	zweite Überschrift für zweite Spalte
</tr>	erste Reihe für Überschriften beenden
</thead>	Bereich für Kopf zu Ende
<tfoot>	Bereich für die Fußzeile (kommt immer vor Inhalt!)
<tr>	Beginn der Reihe für **Fußzeile**
<td> Fuß Z1 </td>	Zell-Anfang, Inhalt Fußzeile 1, Zell-Ende
<td> Fuß Z2 </td>	Zell-Anfang, Inhalt Fußzeile 2, Zell-Ende
</tr>	Ende der Reihe für die Fußzeile
</tfoot>	Bereich für Fuß zu Ende
<tbody>	Bereich für Inhalt beginnt
<tr>	hier fängt die erste Reihe mit **Inhalt** an
<td>Zelle 11 </td>	hier fängt die erste Zelle an, dann der eigentliche Inhalt und die Zelle beenden
<td>Zelle 12 </td>	hier fängt die zweite Zelle an, dann der eigentliche Inhalt und die Zelle beenden
</tr>	erste Reihe mit Inhalt beenden
<tr>	nächste Reihe mit Inhalt
<td>Zelle 21 </td>	erste Zelle, zweite Reihe
<td>Zelle 22 </td>	zweite Zelle, zweite Reihe
</tr>	zweite Reihe mit Inhalt beenden
	Hier können noch beliebig viele weiteren Reihen folgen
</tbody>	Bereich für Inhalt zu Ende
</table>	Tabelle beenden

Formulare

Formulare werden für Eingabe, Ausgabe und Senden von Daten verwendet. Der Formularinhalt wird von den Tags **<form>** und **</form>** eingeschlossen. **<form action= "Zieladresse" method="get">** definiert, wohin die Daten gesendet werden, d.h. an Internetadressen (**url** = uniform resource locator). Wenn das Formular seine Daten an sich selbst (**self**) senden soll, sind neben **<form>** keine weiteren Attribute nötig. Der Formularinhalt kann aus verschiedenen Kind-Elementen bestehen:

(1) **<input type="text" name="…" value="…" size="…" maxlength="…">**
Einzeiliges Text-Feld mit Namen, Anfangswert, Länge und maximaler Länge.

(2) **<textarea name="…" cols="…" rows="…"> </textarea>**
Mehrzeiliges Text-Feld mit Namen, Spaltenanzahl und Zeilenanzahl.

(3) **<input type="submit" value="Senden">** Sendet das Formular an die angegebene Zieladresse.
<input type="button" value="Link" onClick = "location.href='Zieladresse'"> Sendet nach einem Mausklick das Formular an die angegebene Zieladresse.
<input type="reset" value="Reset"> Eine Schaltfläche zum Zurücksetzen des Formulars.

(4) **radio-, checkbox-, select- und option-**Elemente zur Auswahl von verschiedenen Möglichkeiten.

Formulare

<form action="" method=""> <!-- Hier kommen nun die gewünschten Formularelemente --> **</form>**		Zwischen diesen TAGs werden die einzelnen Formularfelder eingeschlossen **action=""** -> entweder Seite, die aufgerufen werden soll und der der Inhalt der Felder übergeben wird, oder Mailadresse in Form "mailto:donald@duck.ent" (nicht ratsam wegen SPAM!) **method="post|get"** – entweder post oder get
<input **type="text"** name="" value="" size="" maxlength="" />	Text-Box	**Text-Box** **name=""** -> Name des Feldes (für Variablenübergabe) **value=""** -> Vorgabewert (normalerweise leer) **size=""** -> Größe der Anzeige **maxlength=""** -> maximale Länge des Feldes
<input **type="hidden"** name="" value="" size="" maxlength="" />		versteckte Feld
<textarea name="" cols="10" rows="60"></textarea>	Text-Area	**Text-Area** Textblock für Eingabe von viel Text – **cols=""** -> wie viele Spalten angezeigt werden **rows=""** -> Anzeige der Anzahl Reihen
<input **type="radio"** name="" value="v" checked="checked" />	Radio-Button Frau | Herr	**Radio-Button** Wird gerne verwendet, wenn es um entweder-oder-auswahlen geht (z. B. Herr oder Frau) **name=""** -> Name des Feldes (für Variablenübergabe) pro Auswahl eine input-Zeile mit demselben Namen! value="v" -> Vorgabewert ist hier wichtig, da der Nutzer nur anklicken kann (Bsp: value="Herr")
<input **type="checkbox"** name="" checked="checked" />	Check-Box	**Check-Box** Anklickbox – über **checked="checked"** kann die Box bereits angeklickt sein
<select name="" size="3" multiple="multiple"> <option value="1" selected="selected">1</option> <option value="2">2</option> <option value="3">3</option> </select>	List-Box 1 2 3	**List-Box** Pull-Down Auswahl **size="3"** -> Größe der Anzeige **multiple="multiple"** -> ob mehrere Elemente angewählt warden können **<option ...** -> die einzelnen Elemente **selected="selected"** -> vorselektiertes
<input **type="Submit"** name="" value="speichern" />	Submit-Button speichern	Button zum Anklicken und Absenden des Formulars **value="speichern"** -> Der value wird als Beschriftung der Schaltfläche angezeigt

---

CSS-Befehle Funktion (Parameter)

Schrift

```
font-family          Schriftart  (Arial, Times New Roman, etc.)
font-size            Schriftgröße  (Prozent oder Pixel (px) )
color                Schriftfarbe  (red, green, blue usw., oder hexadezimal "#RRGGBB" )
font-variant         Schriftvariante  (normal, small-caps)
font-weight          Schriftgewicht  (normal, bold, bolder, lighter)
font-style           Schriftstil  (normal, oblique, italic)
```

Textgestaltung

```
text-align           Textausrichtung  (left, right, center, justify (Blocksatz))
line-height          Zeilenabstand  (Prozent oder Pixel (px) )
text-decoration      Textgestaltung  (underline, overline, line-through, blink)
word-spacing         Wortabstand  (Prozent oder Pixel (px) )
letter-spacing       Zeichenabstand  (Prozent oder Pixel (px) )
text-indent          Texteinrückung  (Prozent oder Pixel (px) )
text-transform       Textart  (capitalize, uppercase, lowercase, none)
```

Links

```
A:link               Farbangabe des Link-Tags ( color = …; )
A:visited            Besuchter Link ( color = …; )
A:hover              Link bei Mausberührung ( color = …; )
A:active             Angeklickter Link ( color = …; )
```

Bilder

```
background-color     Hintergrundfarbe  (red, green, blue usw., oder hexadezimal "#RRGGBB" )
background-image     Hintergrundbild URL("name.jpg")
background-attachement  Hintergrundbild fixiert (fixed) oder gescrollt (scroll)
background-repeat    Kacheln  (repeat, repeat-x, repeat-y, no-repeat)
```

Ränder

```
padding              Innenabstand allseitig  (Prozent oder Pixel (px) )
padding-top          Innenabstand oben  (Prozent oder Pixel (px))
padding-left         Innenabstand links   (Prozent oder Pixel (px))
padding-bottom       Innenabstand unten  (Prozent oder Pixel (px))
padding-right        Innenabstand rechts  (Prozent oder Pixel (px))

border               Dicke der Rahmenlinie allseitig  (thin, medium, thick oder Pixel (px))
border-top-width     Dicke der Rahmenlinie oben  (thin, medium, thick oder Pixel (px))
border-left-width    Dicke der Rahmenlinie links  (thin, medium, thick oder Pixel (px))
border-bottom-width  Dicke der Rahmenlinie unten  (thin, medium, thick oder Pixel (px))
border-right-width   Dicke der Rahmenlinie rechts  (thin, medium, thick oder Pixel (px))

border-style         Rahmentyp  (none,dotted,dashed,solid,double,groove,ridge,inset,outset)
border-color         Rahmenfarbe  (Farbname oder Hexadezimal)

margin               Außenabstand allseitig (Prozent oder Pixel (px) )
margin-top           Außenabstand oben (Prozent oder Pixel (px) )
margin-left          Außenabstand links (Prozent oder Pixel (px) )
margin-bottom        Außenabstand unten (Prozent oder Pixel (px) )
margin-right         Außenabstand rechts (Prozent oder Pixel (px) )

width                Rahmenbreite  (auto, Prozent, Pixel (px))
height               Rahmenhöhe  (auto, Prozent, Pixel (px))
```

LINKS

(1) **** Hinweisender Text ****

(2) **** Hinweisender Text ****

Bei Variante (2) wird das HTML-Dokument in einem neuen Fenster geöffnet.

Pseudoklassen

Folgende Formate werden verwendet, damit ein Feedback bei Links erreicht werden kann.

Beispiel mit CSS-Code:

Sehr wichtig ist die Reihenfolge (Faustformel dazu: LoVe HAte)

```
#navi a:link      { color:blue;     text-decoration:none; }
#navi a:visited   { color:black;    text-decoration:line-through; }
#navi a:focus     { color:green;    text-decoration:overline; }
#navi a:hover     { color:red;      text-decoration:overline; }
#navi a:active    { color:orange;   text-decoration:underline; }
```

Im HTML-Code dann

```
<div id="navi">
  <ul>
    <li><a href="index.htm">Startseite</a></li>
    <li><a href="ueber-mich.htm">Über mich</a></li>
    <li><a href="termine.htm">Termine</a></li>
    <li><a href="impressum.htm">Impressum</a></li>
  </ul>
</div>
```

link	Der normale Anzeige für einen Link
visited	Einen bereits vom Nutzer besuchter Link
focus	Den Zustand von **focus** erreicht man, wenn man über die TAB-Taste (öfters drücken) zu einem Link wandert. Dieser kann dann mit der Return gewählt werden.
hover	Mit der Maus über einen Link fahren
active	Gerade aktiver Link

Absolute/relative Positionierung

position	Art der Positionierung: relative | static | fixed | absolute;
position: absolute;	weiter notwendige Angaben left | right top | bottom width height
z-index	**Beispiel:** z-index: 1; Reihenfolge der Elemente (wird oft bei absolut positionierten Elementen benötigt) Je höher die Nummer, desto weiter im Vordergrund

Kurzschreibweisen

Für einige CSS-Formate gibt es die Möglichkeit, diese in einer Kurzschreibweise zusammenzufassen. Dadurch wird es übersichtlicher und weniger Code.

Kurzschreibweise in Beispielen

border:1px solid red;	Rahmen: border-width border-style border-color;
background: #ff00ee url(bild.jpg) fixed no-repeat right top;	background-color: #ff00ee; background-image: url(bild.jpg); background-attachment: fixed; background-repeat: no-repeat; background-position: right top;
padding: 35px 30px 20px 15px;	Leserichtung wie bei Uhr (oben, rechts, unten, links) Kurzschreibweise für: padding-top: 35px; padding-right: 30px; padding-bottom: 20px; padding-left: 15px;
padding: 35px 18px;	Wenn oben und unten gleich sind (wie auch rechts und links) Kurzschreibweise für: padding-top: 35px; padding-right: 18px; padding-bottom: 35px; padding-left: 18px;
padding: 8px;	Wenn alle 4 Seiten gleich sind Kurzschreibweise für: padding-top: 8px; padding-right: 8px; padding-bottom: 8px; padding-left: 8px;
margin:	Desselbe wie bei padding
font:	font-size und font-family sind Pflichtangaben, wobei die font-family immer die letzte Angabe ist. font: 12pt Arial, sans-serif; wenn auch die line-height mitgegeben wird, dann wird diese mit / von der Schriftgröße getrennt: font:12pt/1.6em serif;

CSS3

Responsives Webdesign: automatische Anpassung der Größe der HTML-Objekte an den jeweiligen Geräte-Bildschirm (device-screen) von desktops, laptops, tablets und phones.

<META NAME="viewport" CONTENT="width=device-width, initial-scale=1.0">

Zusätzliche Media-Query – Anweisungen in CSS3:

@media (not | only) ***mediatype*** **and (*expression*) { CSS-Code }**

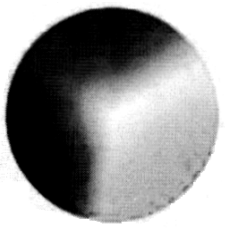
Beispiel:

```
@media screen and (min-width: 600px) and (max-width: 900px) {
    body { background-color: lightblue; font-size: 24px; border: 8px solid black; }
}
```

CSS3

Verschiedene CSS3-Anweisungen funktionieren nur in aktuellen Browserversionen!

Farbangaben CSS3

color: #00ff00;	alte Form der Farbangabe
color: rgb(0, 255, 0);	Farbangabe anhand RGB-Zahlen (dezimal)
color: rgb(0%, 100%, 0%);	Farbangabe in RGB anhand Prozenten
color: hsl(300, 100%, 60%);	Farbangabe im HSL-System H = hue = Farbton S = saturation = Sättigung L = lightness = Helligkeit H als Winkel von 0 – 360 S in Prozent von 0 – 100% L in Prozent von 0 – 100% 

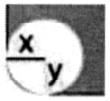
CSS3: Transparenz

opacity:	0 ... 1 Grad der Durchsichtigkeit (vererbt die Eigenschaft!)
color: rgb**a**(..., ..., ..., 0.5)	0 ... 1 | als 4 Wert zur Farbe als Transparenz
color: hsl**a**(..., ..., ..., 0.5)	0 ... 1 | als 4 Wert zur Farbe als Transparenz

CSS3: Schatten

box-shadow: **X**px **Y**px Schattenverlauf Farbe;	Nun kann ein Schatten einem Element mitgegeben werden
box-shadow: **X**px **Y**px Schattenverlauf Farbe **,** **X**px **Y**px Schattenverlauf Farbe;	mehrere Schatten (Trennung erfolgt über Komma)
box-shadow: 10px 10px 100px gray; **-webkit-box-shadow:** 10px 10px 100px gray; **-moz-box-shadow:** 10px 10px 100px gray;	für verschiedene Browserhersteller!

CSS3: abgerundete Ecken

border-radius: **X**px , **Y**px;	Zum abrunden von Ecken – ist nur 1 Wert angegeben, gilt dieser für X und Y 
border-top-left-radius: border-top-right-radius: border-bottom-left-radius: border-bottom-right-radius:	Für jede Ecke einzeln definierbar
border-radius: 40px; -moz-border-radius: 40px; -webkit-border-radius: 40px; **Besonderheiten Firefox:** -moz-border-radius-topleft: 40px; -moz-border-radius-topright: 40px; -moz-border-radius-bottomleft: 40px; -moz-border-radius-bottomright: 40px;	für verschiedene Browser **Besonderheiten Webkit-Browser:** -webkit-border-radius: 40px 30px; /*Leerzeichen!*/

CSS3: Schatten bei Text

text-shadow: Xpx Ypx Schattenverlauf Farbe;	Beispiel: text-shadow: 5px 10px 8px orange;
text-shadow: -5px -5px 6px green, 5px 10px 6px red;	Mehrfarbiger Schatten

CSS3: Transformationen

transform: **rotate(X**deg);	**rotieren (drehen)** Beispiel: transform: rotate(**X**deg); WICHTIG: kein Leerzeichen nach rotate!! positiver Wert = Drehung im Uhrzeigersinn negativer Wert = Drehung gegen Uhrzeigersinn 
transform: rotate(30deg); -moz-transform: rotate(30deg); -ms-transform: rotate(30deg); -o-transform: rotate(30deg); -webkit-transform: rotate(30deg);	**rotate für verschiedene Browser**
transform-origin: X Y;	**Mittelpunkt für die Drehung festlegen** transform-origin: 0 0; /* obere linke Ecke */ transform-origin: 100% 0; /* obere rechte Ecke */ transform-origin: 100% 100%; /* untere rechte Ecke */ transform-origin: 50% 0; /* mitte oben */ 
transform: scale(wert); transform: scale(x, y);	**skalieren** bei Angabe eines Wertes wird x wie y vergrößert bei Unterschiedlicher Angabe von x wie y findet eine "Verzerrung" statt 
transform: scale(1.4); -moz-transform: scale(1.4); -ms-transform: scale(1.4); -o-transform: scale(1.4); -webkit-transform: scale(1.4);	**scale für verschiedene Browser**

transform: skew(wert);
transform: skew(x, y);
	schräg, (wind)schief, verdrehen
	Beispiel:
	transform: skew(20deg);
	transform: skew(20deg, 30deg);
	 

transform: skew(0, 20deg); -moz-transform: skew(0, 20deg); -ms-transform: skew(0, 20deg); -o-transform: skew(0, 20deg); -webkit-transform: skew(0, 20deg);	**skew für verschiedene Browser**

transform: translate(wert);
transform: translate(...px, ...px);
	CSS3 translate: umsetzen, verrücken, versetzen
	transform: translate(80px);
	transform: translate(80px, 50px);
	
	bei Angabe eines Wertes wird nur auf x versetzt angezeigt. Bei Angabe von x wie y findet in beiden Richtungen eine entsprechende versetzte Anzeige statt.

Browser-Spezifikationen: ***moz***: Mozilla-Firefox, ***ms***: Microsoft, ***o***: Opera, ***webkit***: Apple-Safari, …

CSS: Styles und Selektoren

(1) Globale Styles:

```
<!DOCTYPE html>
<head>
<style>

   Hier stehen die CSS-Definitionen im Dokumenten-Kopf

</style>
</head>
<body>
```

(2) Inline-Styles:

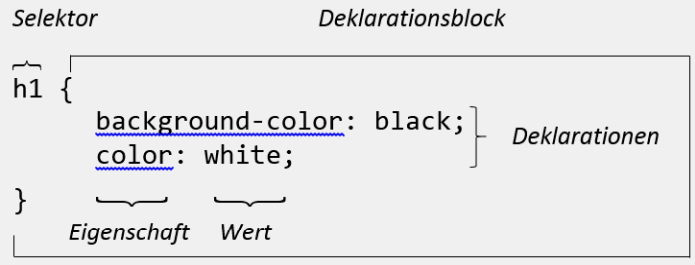
Hier stehen die CSS-Definitionen im Dokumenten-Körper.
Diese Form ist nicht optimal, weil so Inhalt (HTML) mit Präsentation (CSS) gemischt wird.
Ein Inline-Stylesheet wirkt sich nur auf ein Element aus. Beispiel:

```
<body>
  ...
  <img src="bild.jpg" width="240" height="160" alt="Beschreibung"
            style="border: 4px solid red">
 ...
</body>
```

(3) Externe Stylesheets:

Einbinden von externen Stylesheet-Dateien auf einer HTML-Seite im Dokumenten-Kopf.
Beispiel: **<link rel="stylesheet" href="myStyle.css"/>.**

Allgemeine Formatierungsregel für CSS-Befehle:



Beispiele:

```
h1 { font-size:20px; color:red; }

body {
   background-image: url("myPicture.jpg”);
   background-color: #D2B48C;
   margin-left: 20%;
   margin-right: 20%;
   padding: 10px 10px 10px 10px;
   font-family: "Times Roman", Arial;
}
```

Hinweis: Schriftnamen, die Leerzeichen enthalten, sollten in Anführungszeichen gesetzt werden!

Drei Arten der Selektoren-Bildung

(1) Durch Verweis auf einen Objekt-Namen [HTML-Element].
(2) Durch Verweis auf einen Identitäts-Bezeichner von Objekten [id].
(3) Durch Verweis auf einen Klassen-Bezeichner von Objekten [class].

Der einfachste Selektor besteht nur aus dem Namen des HTML-Elements, das formatiert werden soll, beispielsweise ein Absatz im Text (p):

```
p { color: red; }
```

Es kann auch ein Selektor für mehrere HTML-Elemente festgelegt werden:

```
p, h3 { color: red; }
```

HTML-Elemente können über ihre Identitätsbezeichnung **ID** oder über ihre **Klassen**-Bezeichnung angesprochen werden. Dazu werden die HTML-Tags um das Attribut **id="idname"** oder das Attribut **class="classname"** erweitert. Der Unterschied ist, dass mit **class** mehrere Elemente ausgezeichnet werden können, dagegen bezieht sich **id** immer nur auf ein bestimmtes Element pro HTML-Seite.

Für Klassen wird in der CSS-Definition vor dem Namen ein Punkt (**.**) geschrieben.
Für ID wird in der CSS-Definition vor dem Namen das Zeichen **#** geschrieben.

CSS-Definitionen im Dokumenten-Kopf:

```
#hinweis { font-style: bold; }
.bemerkung { font-style: italic; }
```

Entsprechende Objekte im Dokumenten-Körper:

```
<div id="hinweis">
     In der europäischen Union werden die Probleme größer.
     . . . . . .
</div>

<div class="bemerkung">
     In der europäischen Union werden die Probleme größer.
     . . . . . .
</div>
```

Es können auch mehrere Klassen-Selektoren für ein HTML-Element festgelegt werden:

```
p.eins { color: red; }
p.zwei { color: blue; }
```

Links können als Pseudo-Klassen - basierend auf ihrem jeweiligen Zustand – ausgezeichnet werden:

```
a:link { color: blue;  /* noch nicht besuchter Link */ }
a:visited { color: orange;  /* bereits besuchter Link */}
a:hover { color: green;  /* mit der Maus berührter Link */}
a:active { color: red; /* gerade besuchter Link */}
```

Vererbung

Viele CSS-Eigenschaften wirken nicht nur auf die vom Selektor ausgewählten HTML-Elemente, sondern werden auch an die „Nachkommen“ dieser Elemente vererbt. Wird beispielsweise im <**body**> die Schriftfarbe auf rot gesetzt, dann sind alle Textelemente, die sich im HTML-Körper befinden, ebenfalls rot.

Schriften (fonts)

font-family: Schriftart

Angabe der Schriften mit zusätzlicher, generischer Ersatzschrift (durch Beistriche getrennt).
Fonts, die Leerzeichen enthalten, werden in Anführungszeichen gesetzt.

Beispiel 1: font-family: Calibri, Arial, sans-serif; (Schriften ohne Serifen)
Beispiel 2: font-family: "Times New Roman", serif; (Schriften mit Serifen)
Beispiel 3: font-family: "Courier New", monospace; (Nicht proportionale Schriften)

font-size: Schriftgröße

px	**Pixel**
%	**Größe in Prozent in Relation zu einer anderen Schrift**
em	**Skalierungsfaktor für die Schriftgröße in Relation zu einer vorgegebenen Schriftgröße**

<**small**> kleingedruckter Text </**small**>

font-weight: Schriftstärke

Folgende Angaben sind möglich:
inherit (Schriftstärke des Elternelements)
lighter (dünner als im Elternelement)
normal (normale Schriftstärke)
bold (fette Schriftstärke)
bolder (fetter als im Elternelement)

font-style: Schriftstil

Folgende Angaben sind möglich:
inherit (Schriftstil des Elternelements)
normal (normaler Schriftstil, Voreinstellung)
italic (kursiver Schriftstil)
oblique (schräg gestellter Schriftstil)

line-height (Höhe der Textzeile)

Alle Schriftattribute können in einem einzigen Tag gesetzt werden.

Die Reihenfolge der Eigenschaften ist wichtig. Es müssen aber nicht alle Eigenschaften angegeben werden, nur **font-family** und **font-size** sind verpflichtend. Nicht angeführte Eigenschaften werden automatisch auf **normal** gesetzt. Die (optionale) Zeilenhöhe muss direkt nach der Schriftgröße und mit einem Schrägstrich angegeben werden.

font: [font-family] [font-style] [font-variant] [font-weight] [font-size]/[line-height];

Beispiel:
font: Arial, sans-serif italic small-caps bold 1.2em/1.6;

Beispiel:
<body> { font.size = 15px; }
<p {font-size: 1.2em; }> … </p>

In dem letzten Beispiel ergibt sich für den Absatz eine Schriftgröße von 15 * 1.2 = 18px.

Einschub über Zeichensätze

ASCII-Tabelle:

0		1		2		3		4		5		6		7	
8	BS	9		10	LF	11		12	FF	13	CR	14		15	
16		17		18		19		20		21		22		23	
24		25		26	EOF	27	ESC	28		29		30		31	
32		33	!	34	"	35	#	36	$	37	%	38	&	39	'
40	(	41	)	42	*	43	+	44	,	45	-	46	.	47	/
48	0	49	1	50	2	51	3	52	4	53	5	54	6	55	7
58	8	57	9	58	:	59	;	60	<	61	=	62	>	63	?
64	@	65	A	66	B	67	C	68	D	69	E	70	F	71	G
72	H	73	I	74	J	75	K	76	L	77	M	78	N	79	O
80	P	81	Q	82	R	83	S	84	T	85	U	86	V	87	W
88	X	89	Y	90	Z	91	[	92	\	93	]	94	^	95	_
96	`	97	a	98	b	99	c	100	d	101	e	102	f	103	g
104	h	105	i	106	j	107	k	108	l	109	m	110	n	111	o
112	p	113	q	114	r	115	s	116	t	117	u	118	v	119	w
120	x	121	y	122	z	123	{	124	|	125	}	126	~	127	
128		129	ü	130	é	131	â	132	ä	133	à	134	å	135	ç
136	ê	137	ë	138	è	139	ï	140	î	141	ì	142	Ä	143	Å
144	É	145	æ	146	Æ	147	ô	148	ö	149	ò	150	û	151	ù
152	ÿ	153	Ö	154	Ü	155	¢	156	£	157	¥	158	₧	159	ƒ
160	á	161	í	162	ó	163	ú	164	ñ	165	Ñ	166	ª	167	º
168	¿	169	⌐	170	¬	171	½	172	¼	173	¡	174	«	175	»
176	░	177	▒	178	▓	179	│	180	┤	181	╡	182	╢	183	╖
184	╕	185	╣	186	║	187	╗	188	╝	189	╜	190	╛	191	┐
192	└	193	┴	194	┬	195	├	196	─	197	┼	198	╞	199	╟
200	╚	201	╔	202	╩	203	╦	204	╠	205	═	206	╬	207	╧
208	╨	209	╤	210	╥	211	╙	212	╘	213	╒	214	╓	215	╫
216	╪	217	┘	218	┌	219	█	220	▄	221	▌	222	▐	223	▀
224	α	225	ß	226	Γ	227	π	228	Σ	229	σ	230	µ	231	τ
232	Φ	233	Θ	234	Ω	235	δ	236	∞	237	ø	238	ε	239	∩
240	≡	241	±	242	≥	243	≤	244	⌠	245	⌡	246	÷	247	≈
248	°	249	•	250	·	251	√	252	ⁿ	253	²	254	■	255	

Beispiel: Codierung des Strings „**Eva**“ mithilfe der ASCII-Tabelle im 10er-, 16er- und 2er-System.

E = 69, v = 118, a = 97

$69 = 4 * 16^1 + 5 * 16^0 = 45h = \underset{4h}{0100}\ \underset{5h}{0101} = 1 * 2^6 + 1 * 2^2 + 1 * 2^0$

$118 = 7 * 16^1 + 6 * 16^0 = 64h = \underset{7h}{0111}\ \underset{6h}{0110} = 1 * 2^6 + 1 * 2^5 + 1 * 2^4 + 1 * 2^2 + 1 * 2^1$

$97 = 6 * 16^1 + 1 * 16^0 = 61h = \underset{6h}{0110}\ \underset{1h}{0001} = 1 * 2^6 + 1 * 2^5 + 1 * 2^0$

E = 69 = 45h = 01000101
v = 118 = 76h = 01110110
a = 97 = 61h = 01100001

Drückt man auf eine Taste der Tastatur, dann erkennt der Computer den Bytewert des Zeichens. Eine intern gespeicherte Codetabelle (ASCII, American Standard Code for Information Interchange) wandelt bei der Bildschirm- oder Druckerausgabe diesen Code in das grafische Erscheinungsbild des Zeichens um. Im westeuropäischen Sprachraum verbreitete sich die Codetabelle ISO-8850-1, die eine modifizierte ASCII-Tabelle ist und u.a. auch die deutschen Umlaute enthält. Der Nachteil ist die 1-Byte-Darstellung, die nur 256 verschiedene Zeichencodes erlaubt. Dieser Nachteil wurde durch die Einführung des 2-Byte-Unicodes UFT-8 behoben, der jetzt 256*256 = 65536 Zeichen darstellen kann (genug für alle Sprachen der Welt). In HTML gibt es zusätzlich für ganz bestimmte Zeichen so genannte Sondercodes (auch Escape-Sequenzen genannt), beispielsweise:

Ä	`&Auml;`	Ü	`&Üuml;`	&	`&`	¶	`&pi;`
ä	`&auml;`	ü	`&uuml;`	"	`&quot;`	∫	`&int;`
Ö	`&Ouml;`	ß	`&szlig;`	<	`<`	Σ	`&sum;`
ö	`&ouml;`	€	`&euro;`	>	`>`	space	` `

`<SUB>, </SUB>`	Beginn und Ende von tiefgestelltem Text.
`<SUP>, </SUP>`	Beginn und Ende von hochgestelltem Text.

Der Farbwert eines Zeichens kann auch hexadezimal codiert werden mit **#RRGGBB**, wobei jeweils 2 Bytes für die Grundfarben rot (R), grün (G) und blau (B) zur Verfügung stehen.

HTML-Elemente positionieren

Elemente können durch Verwendung der Eigenschaft **position** aus dem normalen Elementfluss entfernt werden und an eine beliebige andere Stelle gesetzt werden.

Mit **position: absolute** kann man Elemente losgelöst vom Textfluss positionieren, an eine Stelle, die mit meistens **top** und **left** festgelegt wird. Auch Größenangaben oder Abstände, wie **width** und **height** oder wie **margin** und **padding**, sind möglich.

Mit **position: relative** wird die Position nur um die angegebenen Werte (+ oder -) verschoben.

```
h1 {
   position: absolute;
   top: 40px;
   left: 100px;
}
```

float setzt Elemente an die linke oder rechte Seite eines Blocks, währen der restliche Text um das Element herumfließt. Mit **float** können die Elemente nebeneinander angeordnet werden.

```
img { float: left; }
```

Wenn Textabsätze mit links oder rechts schwebenden Bildern nicht so hoch sind wie das Bild, entstehen in aufeinander folgenden Textblöcken Treppenstufen. Um Treppen beim Umfließen von Bildern zu vermeiden, muss mit **clear** das Umfließen beendet werden, so dass die Elemente wieder untereinander stehen.

```
img { float: left; }
h2  { clear: left; }
```

Im normalen Elementfluss sind die Elemente (bzw. ihre Boxen) nicht positioniert oder gefloatet. Mit der Eigenschaft **display** wird festgelegt, in welcher Art von Box ein Element erscheint. Elemente mit **display: inline;** erzeugen eine oder mehrere **Inline-Boxen**. Inline-Boxen verlaufen auf einer Zeile horizontal in der Schreibrichtung der verwendeten Sprache. Inline-Boxen können Innen- und Außenabstände sowie Rahmen besitzen.

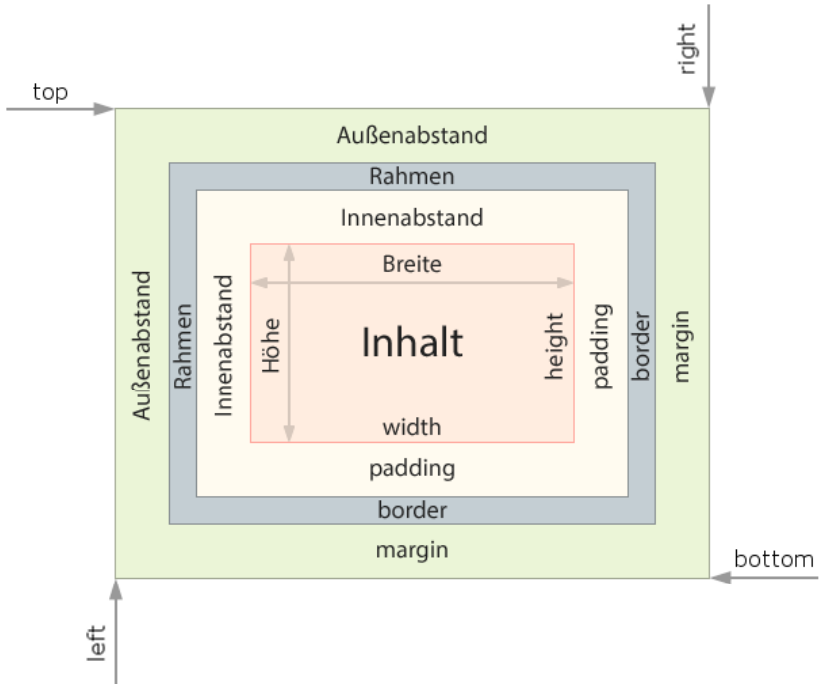
Block-Boxen nehmen die gesamte Breite des Elternelementes ein. Sie sind so hoch, wie ihr Inhalt. Dadurch entsteht ein zusammenhängendes Rechteck, das aussieht wie eine Box, und das genau hat dieser Darstellungsart ihren Namen gegeben.

Das klassische Box-Modell

CSS behandelt eine Webseite so, als wäre jedes darin enthaltene Element in einer unsichtbaren Box eingeschlossen. Eine Box besteht aus einem Inhalt (content), dem diesen Inhalt umgebenden Innenraum (**padding**), der Außenkante des Paddings (Rahmen, **border**) und dem unsichtbaren Bereich um diesen Rahmen (Rand, **margin**), der die Elemente voneinander trennt.

Außenabstand: margin-top, margin-right, margin-bottom, margin-left
Innenabstand: padding-top, padding-right, padding-bottom, padding-left

Beispiele:	`margin-left: 5px;`
	`padding-top: 10px;`
	`margin: 10px 5px 10px 5px;`
	`padding: 10px 5px 10px 5px;`



Durch die Angabe von **margin** oder **padding** ohne Zusatz von **top, right, bottom** oder **left** können die Abstände für alle vier Positionen festgelegt werden

Eine Angabe: für alle vier Abstände gilt derselbe Wert.
Zwei Angaben: 1. Wert für top und bottom. 2. Wert für right und left.
Drei Angaben: 1. Wert für top, 2. Wert für right und left. 3. Wert für bottom.
Vier Angaben: 1. Wert für top. 2. Wert für right. 3. Wert für bottom. 4. Wert für left.
(Merkhilfe: Uhrzeigerbewegung)

Treffen zwei untereinanderliegende Ränder (margins) aufeinander (collapsing), dann verschmelzen sie so, dass nur der größere Rand (margin) dargestellt wird.

Maßangaben:

X px = X Bildpunkte (Pixel)
X % = X Prozent relativ zum umgebenden Element
X em = X mal so groß wie das jeweilige Element

Mit **margin: auto** können Block-Elemente zentriert werden. Dazu muss aber eine feste Breite des Eltern-Elements definiert sein. Mit **text-align: center** können definierte Bereiche der Seite zentriert werden.

Über die Eigenschaft **border** kann die Rahmendicke, der Rahmentyp und die Rahmenfarbe festgelegt werden. Es ist auch möglich, Angaben für Rahmendicke, Rahmenfarbe und Rahmentyp für einzelne Seiten des Elements zu machen.

border-top definiert einen Wert für oben,
border-right für rechts,
border-bottom für unten,
border-left für links.

Beispiele:	`border: 1px solid red;`
	`border-bottom: thick double blue;`

Spezielle Strukturelemente von HTML5

Folgende Elemente steuern den grundsätzlichen Aufbau (layout) einer Internetseite mit HTML5, d.h. die Seitenstrukturierung:

body	**gesamter Inhaltsbereich**
header	**einleitende Inhalte**
nav	**Seitennavigation**
main	**Hauptinhalt**
section	**thematische Gruppierung von Inhalten**
article	**einzelne in sich geschlossene Inhaltsbereiche**
aside	**ergänzende Hinweise, Randbemerkungen**
footer	**informative Inhalte etwa Impressum, Copyright, Autor**
address	**Kontaktinformationen**

Der **header** enthält den sichtbaren Kopfbereich einer Webseite oder eines Teils einer Seite, der für einleitende Inhalte gedacht ist (Firmenlogos, Links zum Impressum oder zur Kontaktseite, …).

```
<header>
   <img src="/images/logo.png" alt="Firmenlogo">
   <h1>Herzlich Willkommen!</h1>
</header>
```

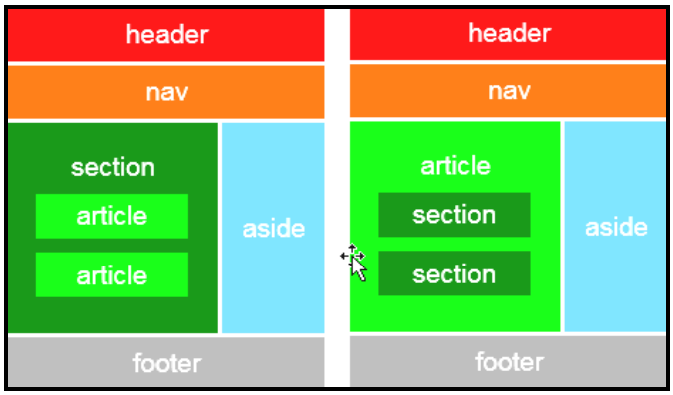
Das **nav**-Element umschließt Navigationsleisten und Menüs, wobei es neben einer unsortierten Liste mit den entsprechenden Links auch eine Überschrift oder ähnliches enthalten kann.

```
<nav>
   <h2>Navigation</h2>
      <ul>
         <li><a...> Link 1</a></li>
         <li><a...> Link 2</a></li>
         <li><a...> Link 3</a></li>
      </ul>
</nav>
```

Das **section**-Element dient zur Unterteilung von Inhalten nach inhaltlichen Gesichtspunkten.
Es enthält zumeist eine thematische Gruppierung von Inhalten – sehr oft mit einer Überschrift.

Das **article**-Element stellt einen in sich geschlossenen Abschnitt eines Dokuments dar, vergleichbar mit einem Zeitungsartikel. Der Unterschied zum **section**-Element ist der, dass die Inhalte für sich alleine stehen. Es sind sozusagen isolierte Beiträge wie Blog-Postings oder Webseiten-Inhalte. Dies bedeutet, dass ein oder mehrere **article**-Elemente in einem themenbezogenen **section**-Element optimal platziert werden können. Das **section**-Element sollte verwendet werden, wenn es auf einer Seite tatsächlich verschiedene Themenbereiche gibt, die sich zusammenfassen lassen.

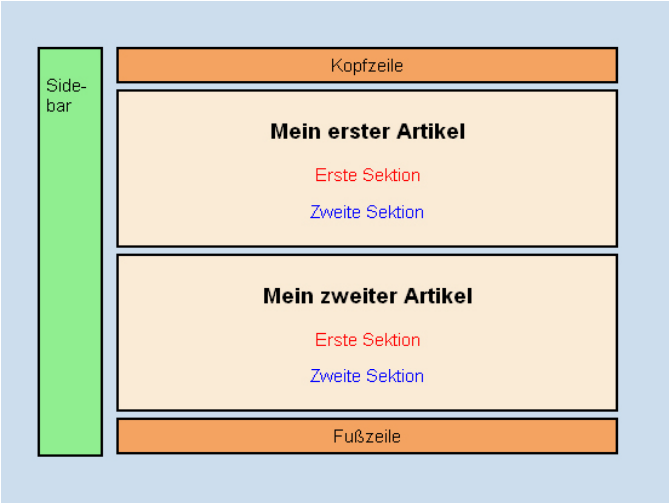
Es kann aber auch in **articles** thematische Gruppierungen von Inhalten mit einer Überschrift geben. Für solche Fälle sind **section**-Elemente innerhalb eines **article**-Elements die richtige Wahl. Die nachfolgende Grafik veranschaulicht diese beiden Varianten einer Seitenstrukturierung.



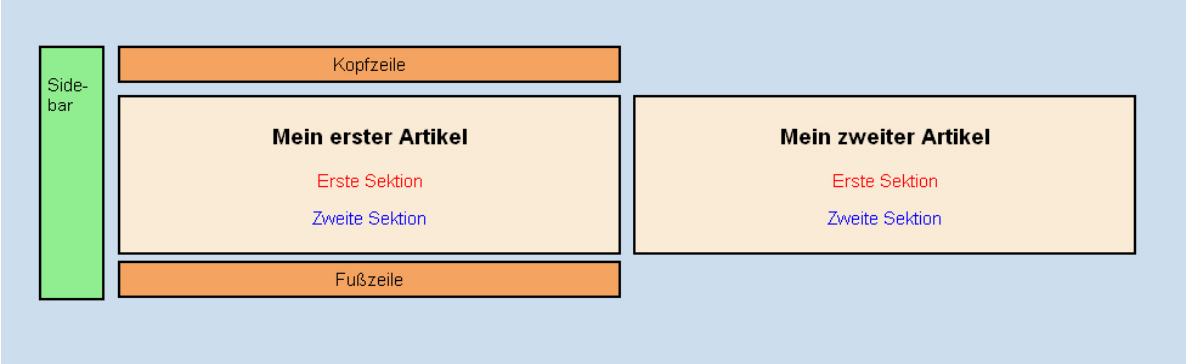
Das **aside**-Element umschließt laut Spezifikation Abschnitte einer Seite, deren Inhalte nur in einem indirekten Zusammenhang mit dem umgebenden Inhalt stehen. Dies sind zum Beispiel Randbemerkungen, Fußnoten oder Links zu weitergehenden Webseiten. Die Platzierung des **aside**-Elementes neben dem Hauptinhalt erfolgt über CSS.

Das **footer**-Element enthält alle Informationen, die in Webseiten am Ende stehen: Autor, Hinweise zum Urheberrecht, ein Link zum Impressum. Dabei kann ein **footer** letztes Element der Seite, aber auch das Ende eines **article**-Elementes sein.

Die folgende Grafik ist ein Snapshot des Programms „**boxen.html**“, das auf der nächsten Seite aufgelistet ist. Die Position des Sidebars ergibt sich aus den Maßangaben der restlichen HTML-Elemente.



Man erhält die folgende Grafik, wenn man im Programm „**boxen.html**“ die Kommentarklammern /* und */ entfernt. Dadurch werden die beiden Artikel horizontal gefloatet.



```
<!DOCTYPE html>
<html>
<head>
<title> Boxen  </title>
<meta charset="ISO-8859-1">
<meta name="author" content="Paukert Herbert">
<meta name="description" content=" Boxen ">

<style>
body{ background-attachment: fixed; font-family: arial; font-size: 15px;
      background-color: #CCDDEE; color: black; margin-left: 100px; margin-top: 50px;}
section.sec1 { color: red; }
section.sec2 { color: blue; }
.headfoot {
   /* clear: left; */
   width: 400px;
   padding: 5px;
   border: 2px solid black;
   background-color: sandybrown;
   margin: 5px;
   text-align: center;
}
.art {
   /* float: left; */
   width: 400px;
   padding: 5px;
   border: 2px solid black;
   background-color: antiquewhite;
   margin: 5px;
   text-align: center;
}
.sidebar {
   position: absolute; top:50px; left:40px;
   width: 40px;
   height: 335px;
   /* height: 200px; */
   background-color: lightgreen;
   padding: 5px;
   border: 2px solid black;
   text-align: left;
}
</style>
</head>

<body>
  <header class = "headfoot">
    Kopfzeile
  </header>
  <article class = "art">
    <h3> Mein erster Artikel </h3>
    <section class = "sec1">
      <p> Erste Sektion </p>
    </section>
    <section class = "sec2">
      <p> Zweite Sektion </p>
    </section>
  </article>
  <article class = "art">
    <h3> Mein zweiter Artikel </h3>
    <section class = "sec1">
      <p> Erste Sektion </p>
    </section>
    <section class = "sec2">
      <p> Zweite Sektion </p>
    </section>
  </article>
  <footer class = "headfoot">
    Fußzeile
  </footer>
  <aside class = "sidebar">
    <br>
    Side-<br>
    bar<br>
    <br>
  </aside>
</body>
</html>
```

Das Animations-System

Eine Vielzahl von **CSS-Anweisungen** dient der Animation.
Dabei werden **HTML-Elemente** dynamisch verändert.
Animationen können mit oder ohne **JavaScript** erzeugt werden.
Darüber handelt dieses letzte Buchkapitel.

• CSS 2D Transforms

translate(Xpx, Ypx) verschiebt ein Element um X Pixel horizontal und Y Pixel vertikal.
translateX(Npx) verschiebt ein Element um N Pixel horizontal.
translateY(Npx) verschiebt ein Element um N Pixel vertikal.
rotate(±Ndeg) rotiert ein Element um N Winkelgrade in oder gegen den Uhrzeigersinn.
scale(X, Y) ändert die Element-Größe um das X-fache horizontal und das Y-fache vertikal.
scaleX(N) ändert die Element-Größe um das N-fache horizontal.
scaleY(N) ändert die Element-Größe um das N-fache vertikal.
skew(Xdeg, Ydeg) stellt ein Element um einen Winkel schräg - bezogen auf eine Achse.
skewX(Ndeg) stellt ein Element um einen Winkel schräg - bezogen auf die x-Achse.
skewY(Ndeg) stellt ein Element um einen Winkel schräg - bezogen auf die y-Achse.
matrix(p1,p2,p3,p4,p5,p6) kombiniert 6 Transform-Methoden: p1 = scaleX(), p2 = scaleY(), p3 = skewX(), p4 = skewY(), p5 = translateX(), p6 = translateY().

Beispiel:
```
div {
   transform: translate(50px,100px);
   transform: rotate(-90deg);
   transform: scale(2,2);
}
```

• CSS Transitions

Die **transition**-Methode ermöglicht eine sanft-fortschreitende Änderung eines Merkmals.
transition-property: Merkmal ; transition-duration: (Milli)Sekunden;
transition-delay: (Milli)Sekunden; transition-timing-function: Speedcurve;
Für Speedcurve gibt es sechs Parameter (ease, ease-in, ease-out, ease-in-out, linear, bezier) .
Sie bestimmen das Geschwindigkeits-Verhalten der Merkmalsänderung. Beispielsweise wird mit „ease“ der Anfang und das Ende der Merkmalsänderung verzögert.

Beispiel:
```
div {
   transition-property: width;
   transition-duration: 2s;
   transition-delay: 2s;
   transition-timing-function: ease;
}
```

• CSS Animations

Damit werden Animationen ermöglicht auch ohne Verwendung von JavaScript.
Bei allen Animationen wird der CSS-Stil von Elementen graduell verändert.
Dabei muss zuerst ein **Keyframe** der Animation festgelegt werden. Dadurch wird die graduelle Änderung des aktuellen Stils zu einem neuen Stil festgelegt.

Die **animation**-Methode selbst kennt ähnliche Eigenschaften wie die **transition**-Methode (animation-duration, animation-delay und animation-timing-function). Zusätzlich gibt es:
animation-name verweist auf einen keyframe-Bezeichner.
animation-iteration-count bestimmt die Anzahl der Durchführungen („n“ oder „infinite“).
animation-direction bestimmt die Ausrichtung: normal (vorwärts), reverse (rückwärts), alternate-reverse (vorwärts und rückwärt).
animation-fill-mode bestimmt den Zustand vor bzw. nach der Animation (forwards, backwards).

Beispiel 0:
```
@keyframes mycolor {
   from { background-color: red; }
   to { background-color: blue; }
}

div {
   width:  100px;
   height: 100px;
   background-color: red;
   animation-name: mycolor;
   animation-duration: 2s;
   animation-timing-function: ease;
}
```

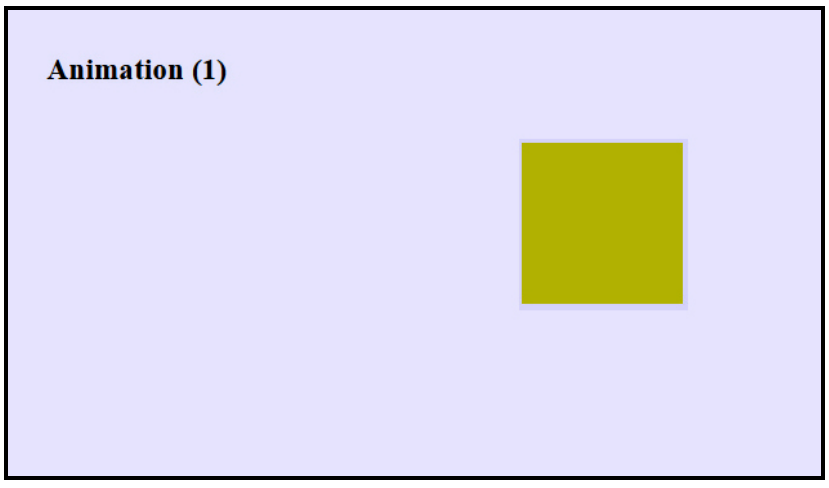
Beispiel 1:
```
@keyframes mymove {
   0%    { background-color: red; left:0px; top:0px; }
   25%   { background-color: green; left:200px; top:0px; }
   50%   { background-color: blue; left:200px; top:200px; }
   75%   { background-color: green; left:0px; top:200px; }
   100% { background-color: red; left:0px; top:0px; }
}

div {
   width:  100px;
   height: 100px;
   position: relative;
   background-color: red;
   animation: mymove 5s infinite;
}
```

Im Beispiel (0) wird ein **div**-Element als quadratische Box (100px) in der Farbe rot definiert.
Eine Animation verändert dann die Farbe durch Verweis auf ein entsprechendes Keyframe.

Im Beispiel (1) wird ein **div**-Element als quadratische Box (100px) in der Farbe rot definiert.
Eine Animation verändert dann die Farbe und die Position durch Verweis auf ein Keyframe.
Dabei wird die Animations-Methode in Kurzform angeschrieben.
Im Keyframe ist anstelle von „from“ – „to“ der Änderungsverlauf in Prozenten angegeben.

Dieses Beispiel (1) wird in ähnlicher Form im Programm „***move01.html***“ realisiert.

„move01.html“



```
<!DOCTYPE html>
<html>
<head>
<title>Animation (1)</title>
<meta charset="ISO-8859-1">
<meta name="viewport" content="width=device-width, initial-scale=1.0;" />
<meta name="author" content="H.P">

<style>
body {
  background: #EEEEFF;
  margin-left: 50px;
}

div {
  width: 100px;
  height: 100px;
  background: red;
  position: relative;
  animation: mymove 5s infinite;
}

@keyframes mymove {
  0%   {top: 0px; left: 100px; background: red;}
  25%  {top: 0px; left: 300px; background: blue;}
  50%  {top: 300px; left: 300px; background: yellow;}
  75%  {top: 300px; left: 100px; background: green;}
  100% {top: 0px; left: 100px; background: red;}
}
</style>
</head>

<body>
  <h3>Animation (1)</h3>
  <br>
  <div></div>
  <br>
</body>
</html>
```

• Der Aufruf von Animationen in JavaScript

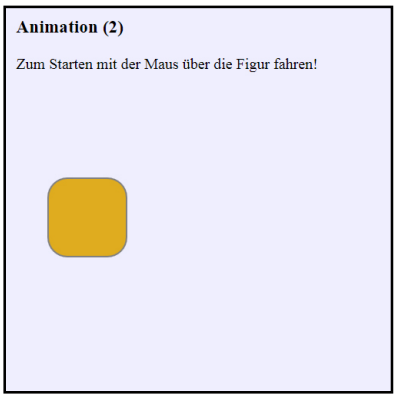
Die **element.animate()**-Methode bindet CSS-Eigenschaften ein, deren Werte dann dynamisch verändert werden. Dabei kann ein Array von Objekten aus CSS-Eigenschaften und deren Werten (d.h. Schlüssel-Wert-Paare) verwendet werden.

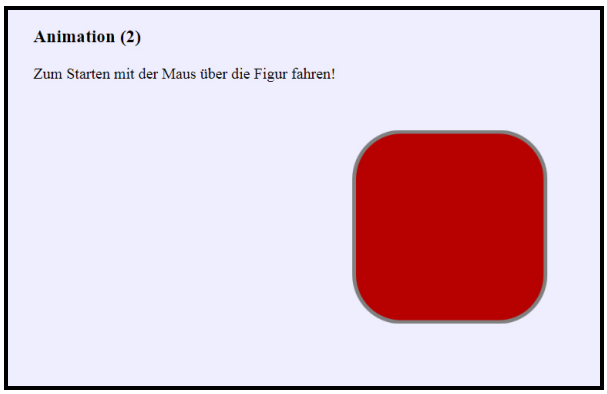
Beispiel:
```
element.animate ( [
   { transform : 'translateX(   0px)', color: 'black'; },
   { transform : 'translateX( 100px)', color: 'red'; },
   { transform : 'translateX(   0px)', color: 'black'; },
   ], {
        duration: 3000,
        delay: 2000,
        interations: infinite
     }
);
```

Eine Animation ist ein Ereignis, welches an einem HTML-Element stattfindet. Das Ereignis wird mit einem entsprechenden Event-Handler verarbeitet. Event-Handler-Funktionen können in JavaScript auf unterschiedliche Art codiert werden. Die Zuordnung eines HTML-Elements zu einer JavaScript-Variablen kann mit ***element = document.getElementById('name');*** erfolgen.

Grundsätzlich kann der gesamte JavaScript-Code in eine **anonyme, selbstausführende Funktion** verpackt werden, welche die entsprechenden Unterfunktionen enthält. Diese Funktion kann nach der Ladung des HTML-Dokuments mit dem „**DOMContentLoaded**"-Ereignis aufgerufen werden.

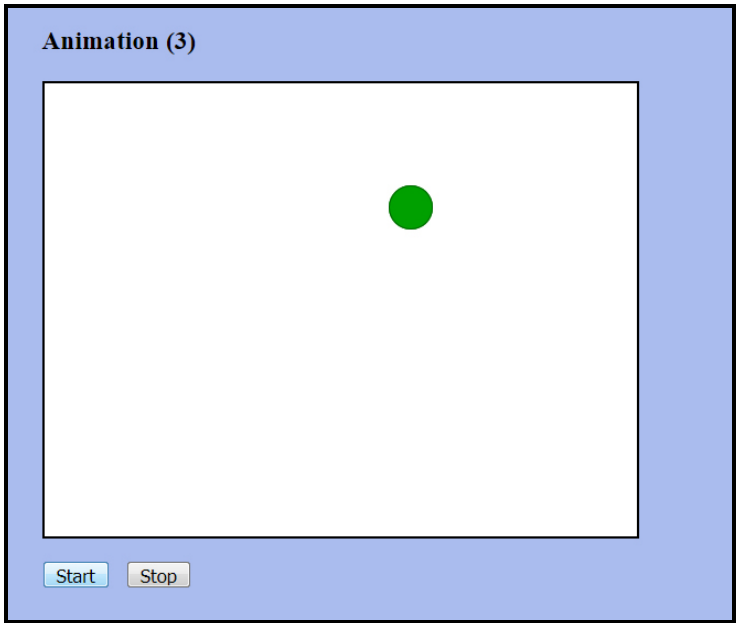
„move02.html“






```
<!DOCTYPE html>
<html>
<head>
<title>Animation (2)</title>
<meta charset="ISO-8859-1">
<meta name="viewport" content="width=device-width, initial-scale=1.0;" />
<meta name="author" content="H.P">

<style>
  body {
    background: #EEEEFF;
    margin-left: 50px;
  }
  #figur {
    position: absolute;
    left: 50px;
    top: 150px;
    width:  50px;
    height: 50px;
    border-radius: 25%;
    background: #DFAC20;
    border: 4px solid gray;
  }
</style>

<script>

  // Aufruf der Funktion "init()" sofort nach Ladung der HTML-Seite
  document.addEventListener("DOMContentLoaded", function() { init(); } );

  function init() {
    var kreis = document.getElementById('figur');
    kreis.onmouseover = function() {
        kreis.animate([
          { transform: 'translate(  0px)' + 'scale(1.0)', background: '#DFAC20' },
          { transform: 'translate(500px)' + 'scale(3.0)', background: 'red' },
          { transform: 'rotate(45deg)' },
          { transform: 'translate(  0px)' + 'scale(1.0)', background: 'black' }
          ],
          { duration: 3000, iterations: 2 }
        );
    };
  }

</script>

</head>

<body>
  <h3>Animation (2)</h3>
  <p>Zum Starten mit der Maus über die Figur fahren!</p>
  <div id="figur"></div>
</body>
</html>
```

„move03.html“




```
<!DOCTYPE html>
<html>
<head>
<title>Animation (3)</title>
<meta charset="ISO-8859-1">
<meta name="viewport" content="width=device-width, initial-scale=1.0;" />
<meta name="author" content="H.P">

<style>
body {
  background: #AABBEE;
  margin-left: 50px;
}
div {
  width:  500px;
  height: 400px;
  background:  white;
  border: 2px solid;
}
#circle {
  background: green;
  left: 50px;
  top:  50px;
  width:  40px;
  height: 40px;
  border-radius: 50%;
  border: 3px solid gray;
}
</style>

<script>

  // Array-Liste von Transformationen
  var move = [
    { transform: 'translate(0, 0)', background: 'green' },
    { transform: 'translate(460px, 360px)', background: 'red' },
    { transform: 'translate(200px, 0px)', background: 'red' },
    { transform: 'translate(0px, 0px)', background: 'green' }
  ];

  function run(){
    var ball = document.getElementById('circle');
    ball.animate(move, {
      duration: 2500,
      iterations: 100,
      fill: 'forwards'
    });
  }
```

```
  function halt() {
    var ball = document.getElementById('circle');
    ball.animate(move, {
      duration: 0,
      iterations: 0,
      fill: 'forwards'
    });
  }

</script>

</head>

<body>
  <h3>Animation (3)</h3>
  <div>
    <div id="circle"></div>
  </div>
  <br>
  <button id="btn1" onclick="run();">Start</button>
    
  <button id="btn2" onclick="halt();">Stop</button>
</body>
</html>
```

Teil B: Verschiedene DEMO-Programme

Als geschichtliche Anmerkung sei noch erwähnt, dass im Jahr 2013 die HTML-Version **HTML 5** im Internet eingeführt wurde, vorher war **HTML 4** der Standard. HTML 5 bietet im Vergleich zu HTML 4 eine größere und funktionsmächtigere Vielfalt an Befehlen. Jedoch ist HTML 5 abwärts kompatibel, so dass auch ältere Befehle aus HTML 4 verstanden werden, beispielsweise der **** Befehl. Damit kann jener Text zwischen **** und **** entsprechend dem Attribut (*color, size*) ausgezeichnet werden. In HTML 5 wird diese Textauszeichnung mit entsprechenden CSS-Befehlen erreicht. Ein anderer alter Befehl aus HTML 4 ist der ** ** Befehl. Damit werden zwischen ** ** und ** ** liegende Elemente zentriert. Auch das Zentrieren kann in HTML 5 mit CSS-Befehlen ausgeführt werden.

Zur ersten einführenden Demonstration seien drei Musterseiten programmiert (*demo1.html, demo2.html* und *demo3.html*). Die erste Seite enthält einige HTML-Objekte und die Verweise (*links*) auf die beiden anderen Seiten. Die zweite Seite enthält in einer Tabelle (*table*) einen Monatskalender. Die dritte Seite enthält einen einfachen Rechner, der mit nur wenigen JavaScript-Befehlen realisiert ist. Der Leser möge die einzelnen Befehle in der Kurzanleitung nachlesen und dann analysieren, vor allem den Syntax der globalen Style-Befehle im Dokumenten-Kopf. Sie verweisen immer auf ein bestimmtes HTML-Objekt und ordnen diesem dann spezielle Attribute zu.

Inhaltsübersicht über die 15 aufgelisteten HTML-Programme:

demo1.html	Schriften, Bilder und Links
demo2.html	Beispiel einer Tabelle
demo3.html (*)	Formelauswertung mit JavaScript
demo4.html	Schriftauszeichnung mit Blocksatz
demo5.html	Formulare senden
demo6.html (*)	Formulare empfangen
demo7.html (*)	Beispiel einer Auswahlbox
demo8.html	Beispiel einer Auswahlliste
demo9.html	Multimedia mit Text, Bild und Ton
demo10.html	Multimedia mit Video
villon.html	Ein Gedicht von Francois Villon
juan.html	Die Geschichte von Don Juan
show.html (*)	Eine zeitgesteuerte Bildershow
galerie.html (*)	Eine einfache Bildergalerie
home.html	Entwurf einer einfachen Homepage

Die fünf mit (*) versehenen Programme enthalten einen kurzen JavaScript-Code zwischen den Tags <script> und </script>.

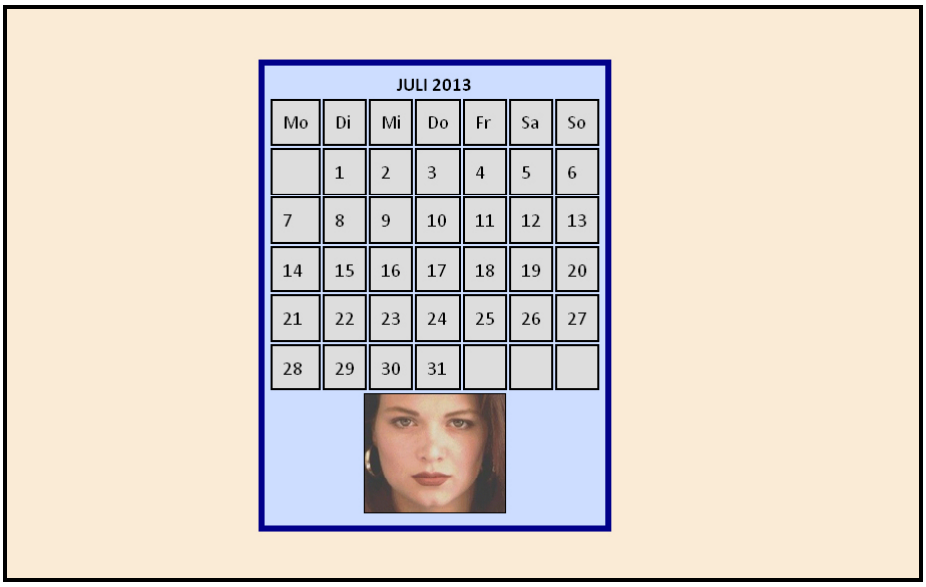
demo1.html – Schriften, Bilder und Links


```
<!DOCTYPE html>
<html>
<head>
<title> Erste Demoseite  </title>
<meta charset="ISO-8859-1">
<meta name="author" content="Paukert Herbert">
<meta name="description" content=" Erste Demoseite ">

<style>

body {
  background-image: url("back.jpg"); background-color:#CCDDEE; color:black;
  font-family:Calibri,Arial; font-size:19px; font-weight:normal;
  margin: 50px; padding-left:50px;
}
a:link {color: blue;}
a:visited {color: blue;}
a:hover {color: yellow;}
a:active {color: yellow;}

</style>
```

```
<body>
<a name="top"></a>
<a href="#bottom">Springe zum Seitenende "BOTTOM"</a>
<br><br>
Normale Textzeile<br>
<p style="color:red; font-weight:bold;font-style:italic;">Rote, fette, schräge Textzeile</p>
<span style="font-family:Arial">Textzeile in ARIAL</span><br>
<span style="font-family:Times Roman">Textzeile in TIMES ROMAN</span><br>
<span style="font-family:Courier New">Textzeile in COURIER NEW</span><br>
<br>
<a href="frau.jpg" target="_blank"><img src="frau.jpg" height="120px" border="1">
  Klick auf das Bild!
</a>
<br><br>
<a href="demo2.html">Springe zur externen Seite "demo2.html"</a>
<br><br>
<a href="demo3.html">Springe zur externen Seite "demo3.html"</a>
<br><br>
<span style="font-size: 19px">Textzeile in Schriftgröße "normal"</span><br>
<span style="font-size: 0.75em">Textzeile in Schriftgröße "0.75%"</span><br>
<span style="font-size: 1.25em">Textzeile in Schriftgröße "125%"</span><br>
<span style="font-size: 1.50em">Textzeile in Schriftgröße "150%"</span><br>
<span style="font-size: 1.75em">Textzeile in Schriftgröße "175%"</span><br>
<span style="font-size: 2.00em">Textzeile in Schriftgröße "200%"</span><br>
<span style="font-size: 19px">Textzeile in Schriftgröße "normal"</span><br>
<br>
<a href="#top">Springe zum internen Seitenanfang "TOP"</a>
<br>
<a name="bottom"></a>
</body>
</html>
```

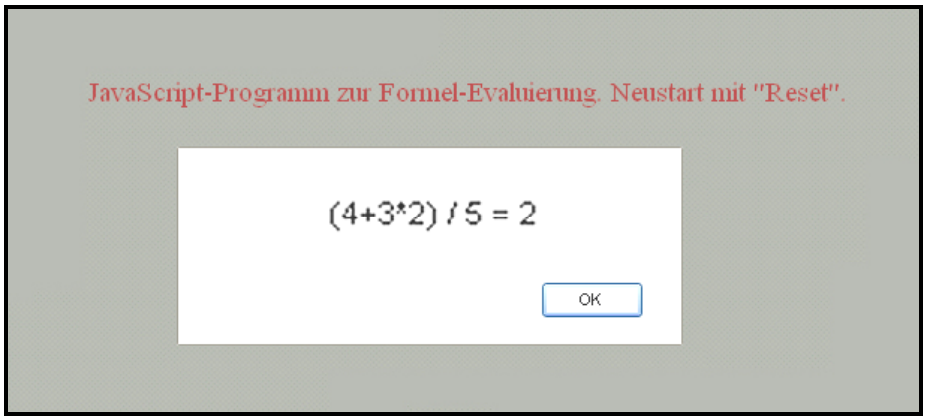
Hinweise:
Das Programm enthält für verschiedene Textzeilen **inline-Styles** zur Textauszeichnung.
Die Schriftgröße bezieht sich mittels **%** **em** auf die normale Schriftgröße.
Die HTML-Tags **** Textzeile . . . **** dienen zur einfachen Textmarkierung.

demo2.html – Beispiel einer Tabelle



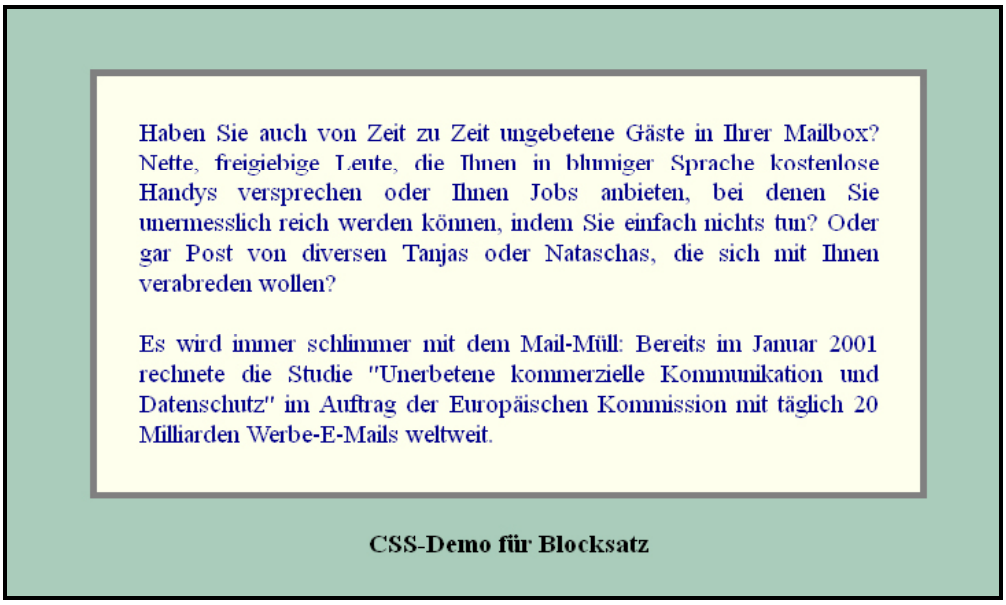

```
<!DOCTYPE html>
<html>
<head>
<title> Zweite Demoseite  </title>
<meta charset="ISO-8859-1">
<meta name="author" content="Paukert Herbert">
<meta name="description" content=" Zweite Demoseite ">

<style>
   body  {
         background-color:antiquewhite; color:black;
         font-family:Calibri,Arial; font-size:19px; font-weight:normal;
         margin: 25px; padding-left:25px;
   }
   table { background-color:#DDEEFF; border: 6px solid darkblue; padding:5px; margin: auto; }
   td { background-color:#DDDDDD; border: 2px solid black; padding:10px; }
</style>

</head>

<body>
<br>
<br>
<table>
  <th colspan=7>JULI 2013</th>
  <tr>
     <td>Mo</td> <td>Di</td> <td>Mi</td> <td>Do</td> <td>Fr</td> <td>Sa</td> <td>So</td>
  </tr>
  <tr>
     <td></td> <td>1</td> <td>2</td> <td>3</td> <td>4</td> <td>5</td> <td>6</td>
  </tr>
  <tr>
     <td>7</td> <td>8</td> <td>9</td> <td>10</td> <td>11</td> <td>12</td> <td>13</td>
  </tr>
  <tr>
     <td>14</td> <td>15</td> <td>16</td> <td>17</td> <td>18</td> <td>19</td> <td>20</td>
  </tr>
  <tr>
     <td>21</td> <td>22</td> <td>23</td> <td>24</td> <td>25</td> <td>26</td> <td>27</td>
  </tr>
  <tr>
     <td>28</td> <td>29</td> <td>30</td> <td>31</td> <td></td> <td></td> <td></td>
  </tr>
  <th colspan=7> <img src="frau.jpg" width="140" border="1"> </th>
</table>
</body>
</html>
```

demo3.html – Einfache Formelauswertung mit JavaScript



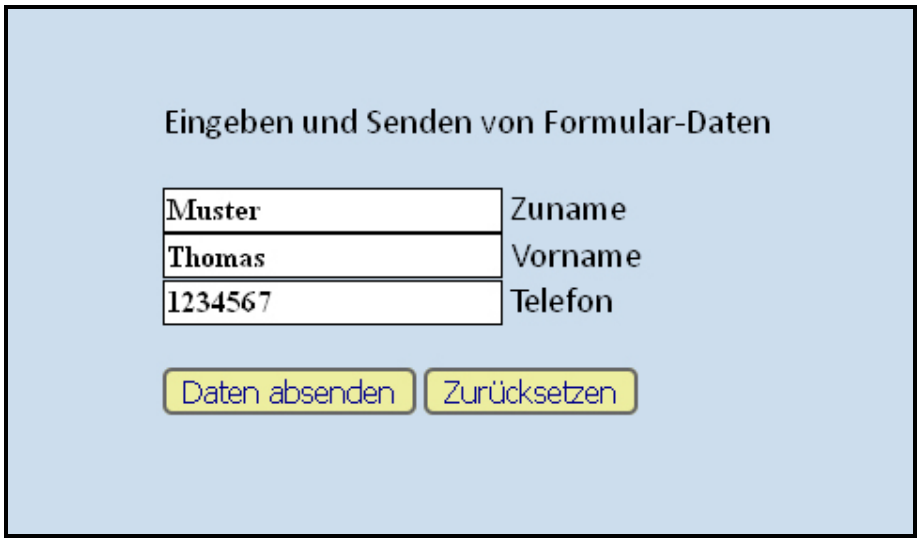

```
<!DOCTYPE html>
<html>
<head>
<title> Dritte Demoseite  </title>
<meta charset="ISO-8859-1">
<meta name="author" content="Paukert Herbert">
<meta name="description" content=" Dritte Demoseite ">

<style>
    body {background-color: #E8F0D8;}
    p {color: red; font-size: 120%;}
</style>

</head>

<body>
   <p>JavaScript-Programm zur Formel-Evaluierung. Neustart mit "Reset".</p>

   <script>
      formel = prompt('Bitte eine Formel eingeben');
      ergebnis = eval(formel);
      alert(formel + ' = ' + ergebnis);
   </script>

</body>
</html>
```

demo4.html – Schriftauszeichnung mit Blocksatz



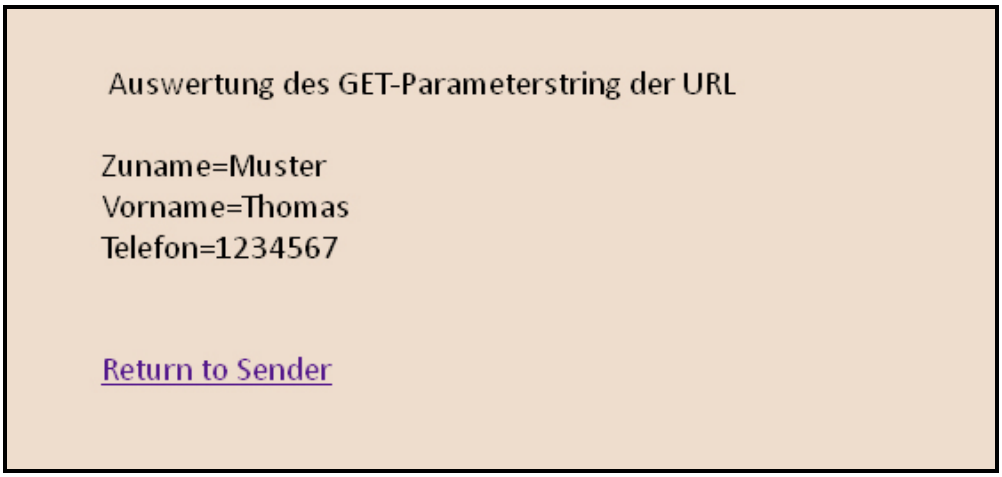

```
<!DOCTYPE html>
<html>
<head>
<title> CSS-Blocksatz </title>
<meta charset="ISO-8859-1">
<meta name="author" content="Herbert Paukert">
<meta name="description" content=" CSS-Blocksatz ">

<style>
   body { background-color: #AACCBB;
   }
   #block { background-color:#FFFFEE; color:darkblue; width:600px; padding:5%;
            border-width:5px; border-style:solid; border-color:#808080;
            font-family:"Times Roman"; font-size:14pt; font-style:normal;
            text-align:justify;
   }
</style>
</head>

<body>
<br><br><br><br><br>
<table style="margin:auto;">
   <tr>
      <td id="block">
        Haben Sie auch von Zeit zu Zeit ungebetene Gäste in Ihrer Mailbox?
        Nette, freigiebige Leute, die Ihnen in blumiger Sprache kostenlose Handys
        versprechen oder Ihnen Jobs anbieten, bei denen Sie unermesslich reich
        werden können, indem Sie einfach nichts tun? Oder gar Post von diversen
        Tanjas oder Nataschas, die sich mit Ihnen verabreden wollen?
        <br><br>
        Es wird immer schlimmer mit dem Mail-Müll: Bereits im Januar 2001 rechnete
        die Studie "Unerbetene kommerzielle Kommunikation und Datenschutz" im Auftrag
        der Europäischen Kommission mit täglich 20 Milliarden Werbe-E-Mails
        weltweit.
      </td>
   </tr>
   <th> <h3>CSS-Demo für Blocksatz</h3> </th>
</table>
</body>
</html>
```

demo5.html – Formulare senden



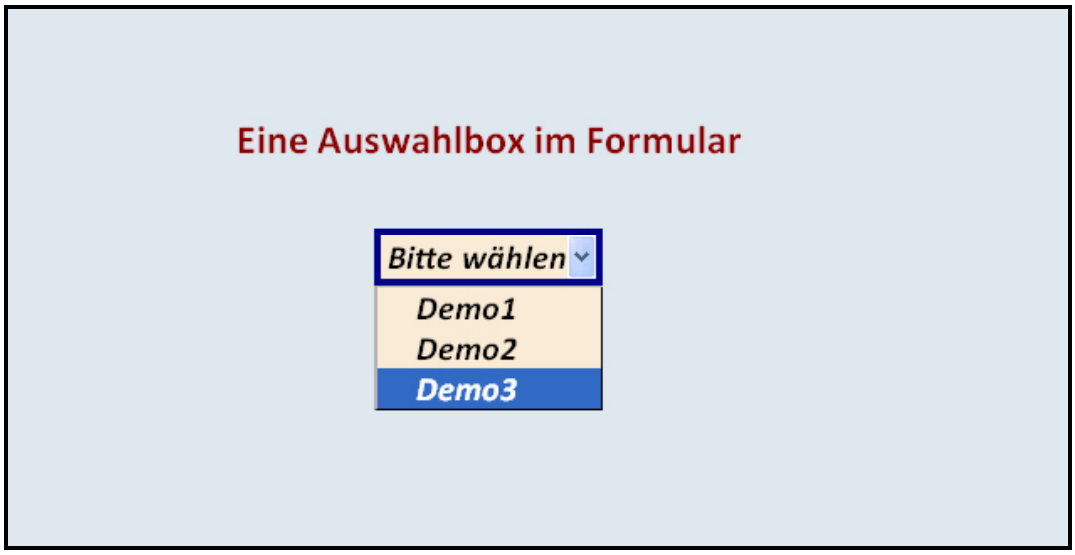

```
<!DOCTYPE html>
<html>
<head>
   <title> Formular senden </title>
   <meta charset="ISO-8859-1">
   <meta name="author" content="Herbert Paukert">
   <meta name="description" content=" Formular senden ">
   <style>
      body {
        background-color:#CCDDEE; color:black;
        font-family:Calibri; font-size:19px; font-weight:normal;
        margin:50px; padding-left:50px;
      }
     .ein {font-family:"Times Roman"; font-size:16px; font-weight:bold;
           border: 1px solid #000000;
      }
     .btn {background-color:#EEEEA0; color:darkblue; font-size:16px;
           border:2px solid #666666; border-radius:5px;
      }
   </style>
</head>

<body>
  <form action="demo6.html" method="get" enctype="text/plain">
    <br>Eingeben und Senden von Formular-Daten<br><br>
    <input class="ein" type="text" name="Zuname" value="Muster" size="20"> Zuname<br>
    <input class="ein" type="text" name="Vorname" value="Thomas" size="20"> Vorname<br>
    <input class="ein" type="text" name="Telefon" value="1234567" size="20"> Telefon<br>
    <br>
    <input class="btn" type="submit">
    <input class="btn" type="reset">
  </form>
</body>
</html>
```

demo6.html – Formulare empfangen



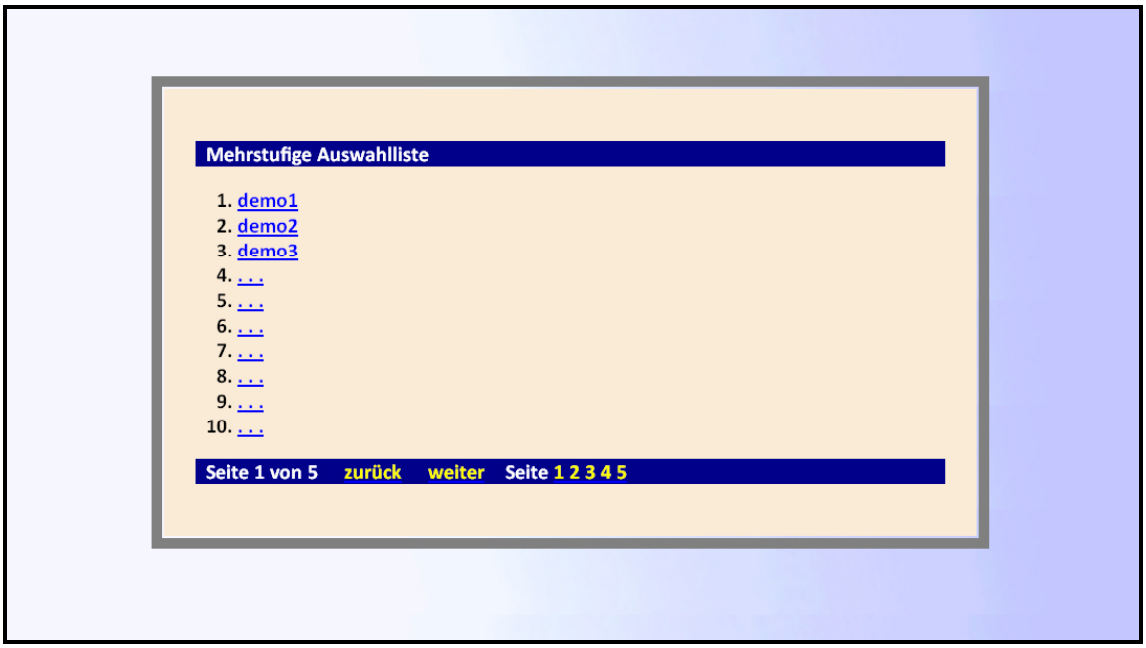

```
<!DOCTYPE html>
<html>
<head>
   <title> Formular empfangen </title>
   <meta charset="ISO-8859-1">
   <meta name="author" content="Herbert Paukert">
   <meta name="description" content=" Formular empfangen ">

   <style>
      body {
        background-color:#EEDDCC; color:black;
        font-family:Calibri; font-size:19px; font-weight:normal;
        margin:50px; padding-left:50px;
      }
    </style>

</head>

<body>
<br>
   Auswertung des GET-Parameterstring der URL
<br><br>

<script>
  with (document)
  {
    x = location.search.substring(1);
    a = x.split("&");
    for (var i = 0; i < a.length; i++)
       { write("  " + a[i] + "<br>"); }
  }
</script>

<br><br>
  <a href="demo5.html">Return to Sender</a>
<br>
</body>
</html>
```

demo7.html – Beispiel einer Auswahlbox




```
<!DOCTYPE html>
<html>
<head>
   <title>Auswahlboxen</title>
   <meta charset="ISO-8859-1">
   <meta name="viewport" content="width=device-width, initial-scale=1.0">
   <meta name="author" content="Herbert Paukert">
   <meta name="description" content="Auswahlboxen">

   <style type="text/css">
       body  {background-color:#E0E8EF; color:black;
              font-family:Calibri; font-size:16px; font-weight:normal;
              text-align:center; margin:5%;}
       h2    {color:darkred;}
       #sbox {background-color:antiquewhite; color:black;
              font-family:Calibri; font-size:20px; font-weight:bold; font-style: italic;
              border: 4px solid darkblue;}
   </style>

   <script>
      function func() {
        var el = document.getElementById("sbox");
        var n  = el.selectedIndex;
        var s  = el.options[n].text.trim().toLowerCase();
        location = s + '.html';
        el.value = "Bitte wählen";
      }
   </script>

</head>

<body>
<form>
   <h2> Eine Auswahlbox im Formular </h2>
   <br>
    <select id="sbox" onchange="func()">
      <option selected disabled hidden>Bitte wählen</option>
      <option>     Demo1     </option >
      <option>     Demo2     </option>
      <option>     Demo3     </option>
    </select>
   <br>
</form>
</body>
</html>
```

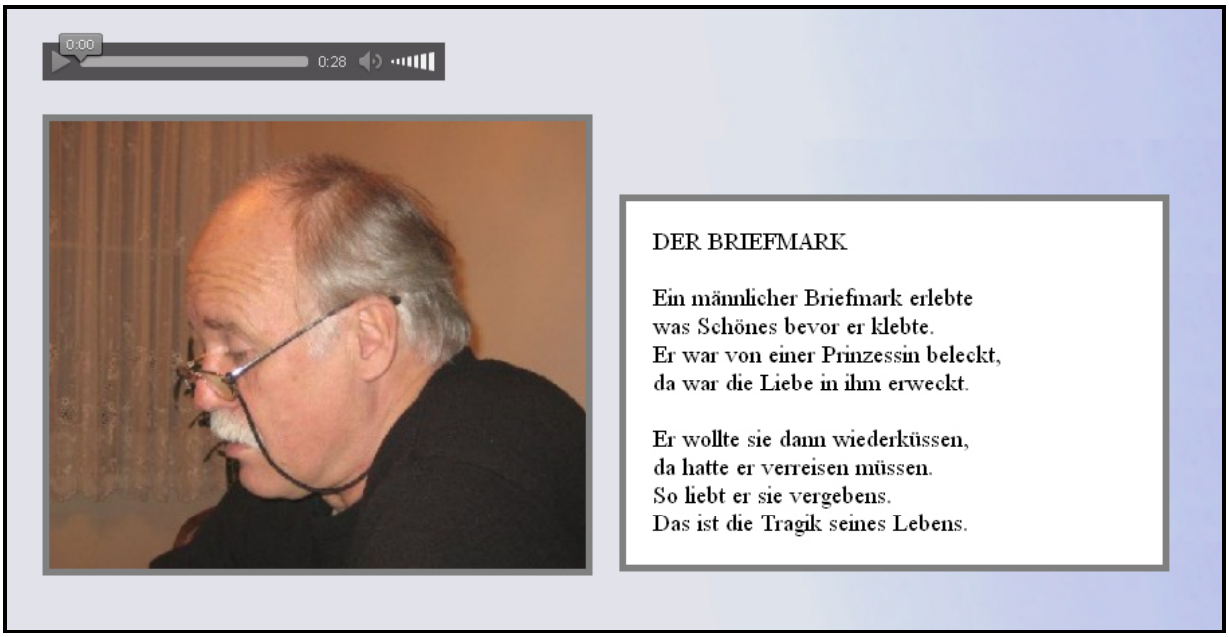
demo8.html – Beispiel einer Auswahlliste




```
<!DOCTYPE html>
<html>
<head>
<title> Auswahllisten  </title>
<meta charset="ISO-8859-1">
<meta name="author" content="Paukert Herbert">
<meta name="description" content=" Auswahllisten ">

<style>
body {
  background-image: url("back1.jpg"); background-color:#CCDDEE; color:black;
  font-family:Calibri,Arial; font-size:20px; font-weight:normal;
}

a:link {color: blue;}
a:visited {color: blue;}
a:hover {color: red;}

span {color: yellow;}

#zentral {
  position:absolute;
  top:45%;
  left:45%;
  width:40em;
  height:20em;
  margin-left:-15em;
  margin-top:-10em;
  border: 10px solid gray;
}

#cell {
  background-color:antiquewhite;
  padding:30px;
  text-align:left;
}

.menue {
  background-color:darkblue;
  color:white;
  font-style:italic;
}

</style>
</head>
```

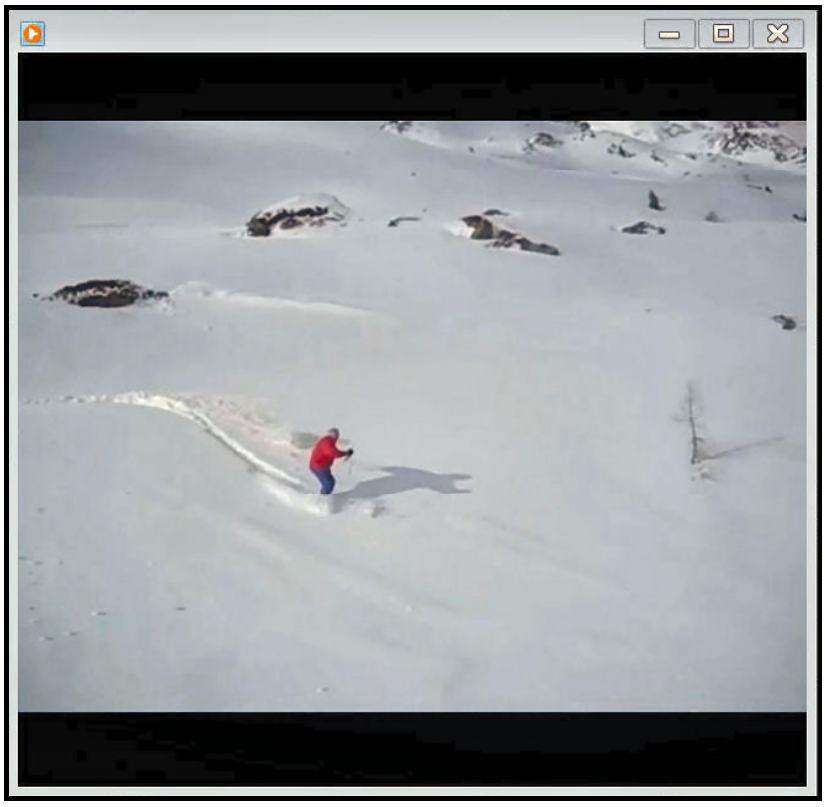
```
<body>
<table id = "zentral">
<tr>
<td id = "cell">
<b>

<p class = "menue">
  Mehrstufige Auswahlliste  
</p>

<ol start="1">
<li><a href="demo1.html">demo1</a></li>
<li><a href="demo2.html">demo2</a></li>
<li><a href="demo3.html">demo3</a></li>
<li><a href="..."> . . . </a></li>
<li><a href="..."> . . . </a></li>
<li><a href="..."> . . . </a></li>
<li><a href="..."> . . . </a></li>
<li><a href="..."> . . . </a></li>
<li><a href="..."> . . . </a></li>
<li><a href="..."> . . . </a></li>
</ol>

<p class = "menue">
  Seite 1 von 5
   
<a href="..."><span> zurück </span></a>
   
<a href="..."><span> weiter </span></a>
   Seite
<a href="demo8.html"><span> 1 </span></a>
<a href="..."><span> 2 </span></a>
<a href="..."><span> 3 </span></a>
<a href="..."><span> 4 </span></a>
<a href="..."><span> 5 </span></a>
</p>

</b>
</td>
</tr>
</table>

</body>
</html>
```

demo9.html – Multimedia mit Text, Bild und Ton




```
<!DOCTYPE html>
<html>
<head>
<title> Multimedia </title>
<meta charset="ISO-8859-1">
<meta name="author" content="Paukert Herbert">
<meta name="description" content=" Multimedia ">

<style>
body   { background-image: url(back1.jpg); background-attachment: fixed;
         background-color: #CCDDEE; color: black; margin-left: 5%; }
.bild  { width: 400px; border: 5px solid gray; float: left;}
.block { background-color: #FFFFFF; color: black; width: 360px;
         font-family: Times Roman; font-size:18px; font-weight: normal;
         border: 5px solid gray; padding: 20px; margin: 20px; float: left; }
</style>
</head>

<body>
<br>
<audio width="320" height="240" controls>
    <source src="briefmark.mp3" type="audio/mp3">
    Your browser does not support the audio tag.
</audio>
<br><br>
<img src = "mann.jpg" class="bild">
<br><br>
<div class="block">
    DER BRIEFMARK<br>
    <br>
    Ein männlicher Briefmark erlebte<br>
    was Schönes bevor er klebte.<br>
    Er war von einer Prinzessin beleckt,<br>
    da war die Liebe in ihm erweckt.<br>
    <br>
    Er wollte sie dann wiederküssen,<br>
    da hatte er verreisen müssen.<br>
    So liebt er sie vergebens.<br>
    Das ist die Tragik seines Lebens.<br>
</div>
<br>
</body>
</html>
```

demo10.html – Multimedia mit Video



```
<!DOCTYPE html>
<html>
<head>
<title> Video  </title>
<meta charset="ISO-8859-1">
<meta name="author" content="Paukert Herbert">
<meta name="description" content=" Video ">

<style>
    body { background-color: #667788; margin: auto; }
</style>
</head>

<body>
<br><br>
<video width="640" controls>
    <source src="schi.mp4" type="video/mp4">
    Your browser does not support the video tag.
</video>
<br><br>
</body>
</html>
```

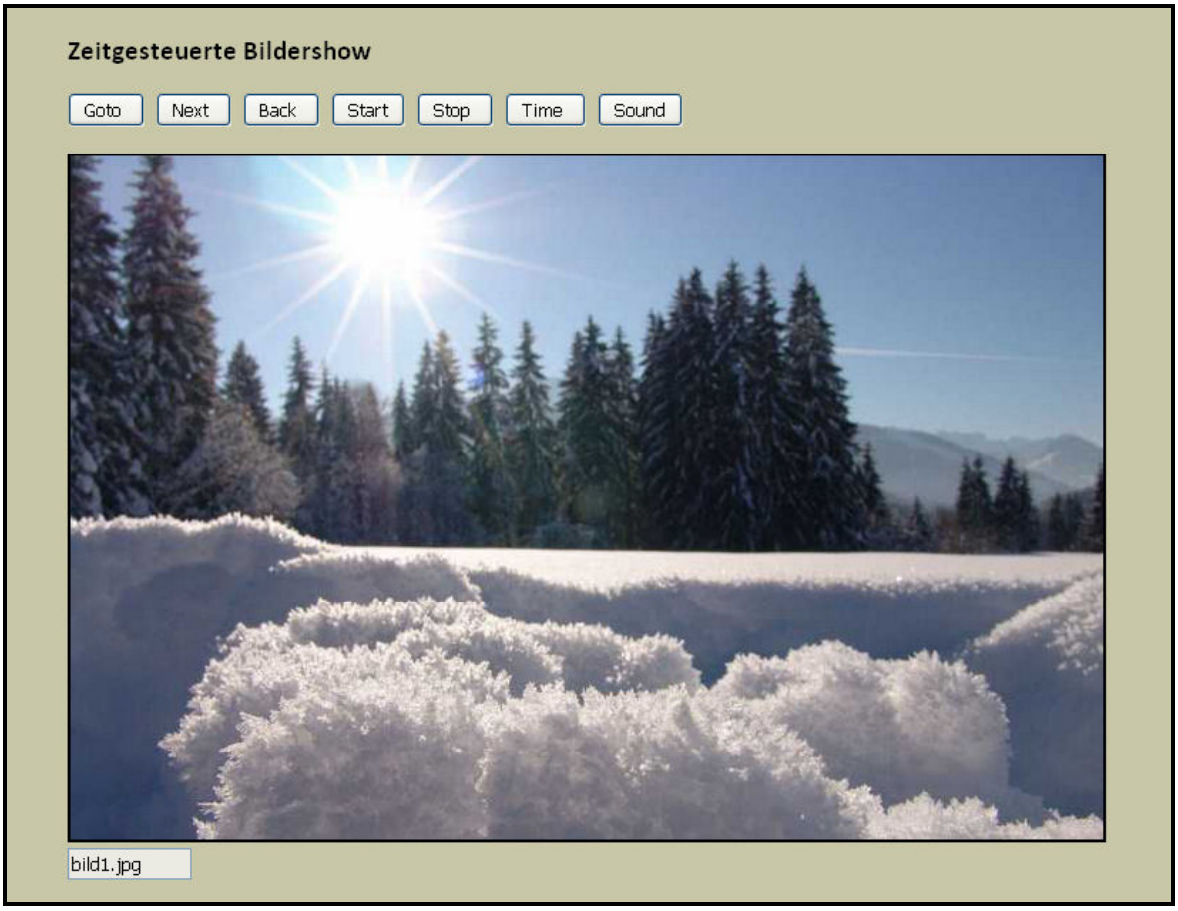
villon.html – Francoise Villon: „Ich bin so wild …“


```
<!DOCTYPE html>
<html>
<head>
<title>François Villon</title>
<meta charset="ISO-8859-1">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<meta name="author" content="Paukert Herbert">
<meta name="description" content="François Villon">
</head>

<body style = "background-image:url(farb02.gif); background-color:antiquewhite; color:#000080;
               font-family: Arial Narrow; font-size:18px; font-weight:normal; text-align: center;
               padding-left:64; padding-right:64; padding-top:8;padding-bottom:8; >
<div style = "font-size:24px; margin: auto;">
<br>
<img src="villon.gif" style ="border: 2px solid darkred">
<br>
<audio width="320" height="240" controls>
  <source src="villon.mp3" type="audio/mp3">
  Your browser does not support the audio tag.
</audio>
</div>
<br>
<p style="color:darkred; font-style:italic">
Hier ist eines der schönsten Liebesgedichte, das jemals geschrieben wurde.<br>
Es stammt von François Villon. Fast unglaublich, dass es schon über 550 Jahre alt ist.
</p>
<h2 style="color:red; font-style:italic">
Ich bin so wild nach deinem Erdbeermund
</h2>
<p>Du, ich bin so wild nach deinem Erdbeermund,<br>ich schrie mir schon die Lungen wund<br>nach
deinem weißen Leib, du Weib.<br>Im Klee, da hat der Mai ein Bett gemacht,<br>da blüht ein süßer
Zeitvertreib<br>mit deinem Leib die lange Nacht.<br>Da will ich sein im tiefen Tal<br>dein
Nachtgebet und auch dein Sterngemahl.</p>
<p>Im tiefen Erdbeertal, im schwarzen Haar,<br>da schlief ich manchen Sommer lang<br>bei dir und
schlief doch nie zuviel.<br>Komm her, ich weiß ein schönes Spiel<br>im dunklen Tal, im
Muschelgrund ...<br>Du, ich bin so wild nach deinem Erdbeermund.</p>
<p>Die graue Welt macht keine Freude mehr,<br>ich gab den schönsten Sommer her,<br>und dir hats
auch kein Glück gebracht;<br>hast nur den roten Mund noch aufgespart,<br>für mich, für mich so
tief im Haar verwahrt.<br>Ich such ihn schon die lange Nacht<br>im Wintertal, im Aschengrund
...<br>Du, ich bin so wild nach deinem Erdbeermund.</p>
<p>Im Wintertal, im schwarzen Erdbeerkraut,<br>da hat der Schnee ein Nest gebaut<br>und fragt
nicht, wo die Liebe sei.<br>Ich habe doch das rote Tier so tief<br>erfahren, als ich bei dir
schlief.<br>Ach, wär nur der Winter schon vorbei<br>und wieder grün der Wiesengrund.<br>Du, ich
bin so wild nach deinem Erdbeermund.</p>
<img src="villon.gif" style ="border: 2px solid darkred">
<br><br>
</div>
</body>

</html>
```

donjuan.html – Die Geschichte von Don Juan


```
<!DOCTYPE html>
<html>
<head>
<title>Don Juan</title>
<meta charset="ISO-8859-1">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<meta name="author" content="Paukert Herbert">
<meta name="description" content="Don Juan">

<style>
body  {color:black; background-color:#AAAADD;  font-family: sans serif;
       font-size: 18px; text-align: left; margin: 2%; }
.eins {font-size:24px; }
.zwei {background-color:antiquewhite; color: darkblue; width: 400px;
       font-family:sans serif; font-size:16px; font-weight: normal; text-align: left;
       padding-left: 2%; padding-right: 2%; padding-top: 2%; padding-bottom: 2%;
       border: 2px solid gray; border-radius: 10px; }
</style>
</head>

<body>
<br>
<img src = "fraufrog.jpg" width = "400" border="2">
<br>
<div class="eins">
<audio width="320" height="240" controls>
  <source src="donjuan.mp3" type="audio/mp3">
  Your browser does not support the audio tag.
</audio>
</div>
<br>
<div class = "zwei">
<h3>DON JUAN oder die wahre Geschichte vom falschen Frosch</h3>
Als Don Juan in die Jahre gekommen war, wo eine Krankenschwester wichtiger als eine Hure ist,
hatte er noch einmal einen Anfall von jugendlichem Irrsinn. Also putzte er sich heraus und
ging auf die Pirsch. Da stolperte er in seinem schwankenden Parkinson-Gang über eine wunderschöne
Prinzessin und fühlte sich noch jugendlicher - ganz so wie früher, vor vielen hundert Jahren.
<br><br>
Er lud sie zum Essen, Trinken und Tanzen ein - ganz so wie früher - merkte aber wohl,
dass Alles viel schwerer war, besonders das Tanzen (das Trinken weniger). Außerdem spielte ihm
sein dement seniles Gedächtnis einen Streich: Irgendetwas wollte er doch noch tun - ganz so wie
früher. Da war doch noch etwas, oder? Aber er hatte es vergessen und so sehr er sich auch abmühte,
es fiel ihm nicht ein. Zu seinem Entsetzen musste er feststellen, dass offensichtlich auch die
wunderschöne Prinzessin, immer ungedulgiger werdend, auf dasjenige wartete, woran er sich partout
nicht mehr erinnern konnte.
<br><br>
Was sollte er tun?  Don Juan, nicht gewohnt sich den Konflikten seines Lebens offen zu stellen,
löste das Problem auf bequeme Art und Weise: Er wurde wieder müde, alt und krank!
Daraufhin verwandelte sich die wunderschöne Prinzessin aus Wut und Enttäuschung
über die verlorene Zeit in einen Frosch und sprang davon.
</div>
<br>
&copy Herbert Paukert
<br>
</body>
</html>
```

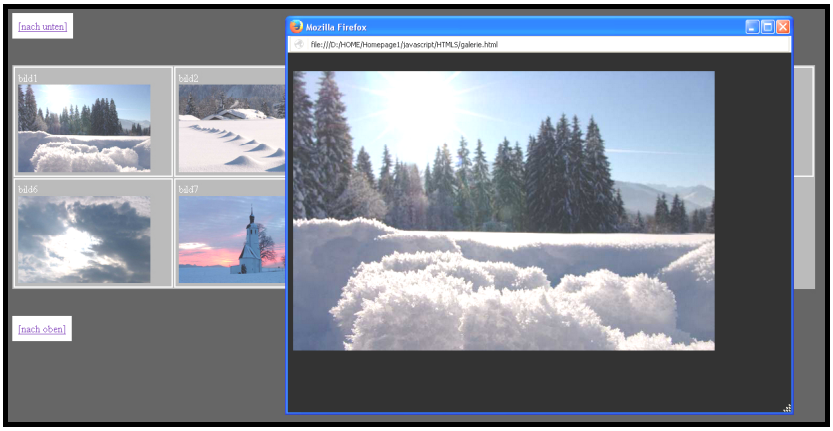
show.html – Eine zeitgesteuerte Bildershow




```
<!DOCTYPE html>
<html>
<head>
<title> Bildershow </title>
<meta charset="ISO-8859-1">
<meta name="author" content="Herbert Paukert">

<style type="text/css">
   body { background-color:#000; color:yellow;
          font-family:'Calibri,sans-serif'; font-size:16px; font-weight:normal;
          text-align:center; margin-left:5%;}
   .btn { background-color: #DDD; border: 2px solid yellow; border-radius:10px; font-size: 18px;}
</style>

<script>

   var tSound = true;
   var pictA = new Image();
   var zeit = 3000;
   var zeit0 = 3000;
   var active;
   var ShowRun = false;
   var num = 1;

   var max = 8;
   var name = 'bild';
   bild = new Array(max);
   for (var i = 1; i <= max; i++) { bild[i] = name + i + '.jpg'; }

   window.onload = function() {
     document.formu.ausgabe.value = bild[1];
     pictA.src = bild[1];
     document.pic1.src = pictA.src;
     window.scrollTo(0,0);
    }
```

```
   function GotoImage()  {
     if (ShowRun ) { return; }
     zahl = prompt('Nummer der Bildseite zwischen 1 und '+(max),num);;
     if (!isNaN(zahl)) {
       num = zahl - 1;
       if (num <= 0) { num = 0; }
       if (num >= max) { num = max-1 ; }
     }
     else { num = 0; }
     GotoNext();
   }
   function GotoNext() {
     num = num + 1;
     if (num > max) { num = 1; }
     document.formu.ausgabe.value = num + ' / ' + bild[num];
     pictA.src = bild[num];
     document.pic1.src = pictA.src;
     if (ShowRun) { active = window.setTimeout("GotoNext()",zeit);  }
   }
   function GotoLast() {
     if (ShowRun ) { return; }
     num = num - 1;
     if (num < 1) { num = max; }
     document.formu.ausgabe.value = num + ' / ' + bild[num];
     pictA.src = bild[num];
     document.pic1.src = pictA.src;
   }
   function ShowStart() {
     if (ShowRun) { return; }
     ShowRun = true;
     active = window.setTimeout("GotoNext()",zeit);
    }

   function ShowStop() {
     ShowRun = false;
     window.clearTimeOut(active);
   }
   function GetTime() {
     if (ShowRun) { return; }
     zahl = prompt('Darbietungszeit zwischen 100 und 9000 MSec',zeit);
     if (!isNaN(zahl)) {
       if ((zahl < 100) || (zahl > 9000)) { zahl = zeit0; }
     }
     else { zahl = zeit0; }
     zeit = zahl;
   }
  function SoundToggle() {
     tSound = !tSound;
     if (tSound == true)  { backplay.play();  }
     if (tSound == false) { backplay.pause(); }
  }
</script>
</head>

<body>
<form name = "formu">
<h2> Zeitgesteuerte Bildershow</h2>
<input type = "button" class = "btn" name = "btn1" value = " Goto  " onclick = "GotoImage()"> 
<input type = "button" class = "btn" name = "btn2" value = " Next  " onclick = "GotoNext()"> 
<input type = "button" class = "btn" name = "btn3" value = " Back  " onclick = "GotoLast()"> 
<input type = "button" class = "btn" name = "btn4" value = " Start " onclick = "ShowStart()"> 
<input type = "button" class = "btn" name = "btn5" value = " Stop  " onclick = "ShowStop()"> 
<input type = "button" class = "btn" name = "btn6" value = " Time  " onclick = "GetTime()"> 
<input type = "button" class = "btn" name = "btn7" value = " Sound " onclick = "SoundToggle()"> 
<br><br>
<img src="" id="pic1" name="pic1" style="height:480px; border:2px solid silver; border-radius:10px;">
<br>
<input type = "text" name="ausgabe" value="" size="10" readonly>
<audio id="backplay" class = "btn" loop autoplay> <source src="backsound.mp3" type="audio/mp3">
</audio>
</form>
</body>
</html>
```

galerie.html – Eine einfache Bildergalerie




```
<!DOCTYPE html">
<html>

<head>
<title>galerie.html</title>
<meta charset="ISO-8859-1">
<meta name="author" content="Herbert Paukert">
<meta name="description" content="Bilder">

<style>
   body { background-color: #333333; color: white; text-align: center; }
   .tab { width:90%; background-color: #888888; }
   td   { border:2px solid white; padding: 5px; margin: 5px; }
</style>

<script>
   var Fenster;
   window.addEventListener("unload", function() {Fenster.close();} );

   function PopUp(bild)
   {
     if (Fenster) { Fenster.close(); }
     Fenster = window.open("","","top=0,left=0,width=920,height=690,resizable=yes,
                          scrollbars=yes");
     var b = "<IMG SRC = " + bild + " height=80%>";
     with (Fenster.document)
     {
        writeln("<HTML><HEAD><TITLE></TITLE></HEAD>");
        writeln("<BODY style='background-color: black;'>");
        writeln("<br>");
        writeln(b);
        writeln("<br>");
        writeln("</BODY></HTML>");
      }
    }
</script>

</head>
```

```
<body>

<table style="background-color:antiquewhite; color:black; ">
  <tr><td> <a href="#bottom"> [nach unten] </a> </td></tr>
</table>

Bildergalerie - Bild mit Mausklick auswählen
<br><br>

<table class="tab">
  <tr>
     <td> bild1 <br><img src="bild1.jpg" width="200" onclick="PopUp('bild1.jpg')"> </td>
     <td> bild2 <br><img src="bild2.jpg" width="200" onclick="PopUp('bild2.jpg')"> </td>
     <td> bild3 <br><img src="bild3.jpg" width="200" onclick="PopUp('bild3.jpg')"> </td>
     <td> bild4 <br><img src="bild4.jpg" width="200" onclick="PopUp('bild4.jpg')"> </td>
     <td> bild5 <br><img src="bild5.jpg" width="200" onclick="PopUp('bild5.jpg')"> </td>
  </tr>
  <tr>
     <td> bild6 <br><img src="bild6.jpg" width="200" onclick="PopUp('bild6.jpg')"> </td>
     <td> bild7 <br><img src="bild7.jpg" width="200" onclick="PopUp('bild7.jpg')"> </td>
     <td> bild8 <br><img src="bild8.jpg" width="200" onclick="PopUp('bild8.jpg')"> </td>
  </tr>
</table>
<br><br>

<table style="background-color:antiquewhite; color:black;">
  <tr><td> <a href="#top"> [nach oben] </a> </td></tr>
</table>

<br><br>
<a name = "bottom"></a>

</body>
</html>
```

home.html – Entwurf einer einfachen Homepage


```
<!DOCTYPE html">
<html>
<head>
<title>Bilder, Bilderrahmen, Wien, Mustermann</title>
<meta charset="ISO-8859-1">
<meta name="author" content="Mustermann">
<meta name="description" content="Bilder, Bilderrahmen, Wien, Mustermann">

<style type="text/css">
body {background-color:#C0C4F4; color:black;
      font-family:Calibri,'Times New Roman'; font-size:13pt; font-weight:normal;
      padding:2%; margin:1%;
}
div.eins {background-color:#E8D8E8; color: black; padding: 20px; margin: 50px;
          font-family:Calibri,'Times New Roman'; font-size:13pt; font-weight:normal;
          text-align:center;
}
div.zwei {margin-left:50px; }

h1 {color: darkblue; }
h2 {color: darkred; }

a:link  {
    color: blue;
    background-color: white;
    font-style: italic;
    font-weight: bold;
    text-decoration: underline;
}
a:visited  {
    color: blue;
    background-color: white;
    font-style: italic;
    font-weight: bold;
    text-decoration: underline;
}
a:active  {
    color: blue;
    background-color: white;
    font-style: italic;
    font-weight: bold;
    text-decoration: underline;
}
a:hover  {
    color: yellow;
    background-color: #8888CC;
    font-style: italic;
    font-weight: bold;
    text-decoration: underline;
}
</style>
</head>
```

```
<body>
<table style = "width: 900px; padding: 20px; margin: auto;">
<tr>
<td style = "background-color: #F4F4FF;">

<table style="background-color:antiquewhite; color:black; border: 2px solid gray; padding: 2px">
  <tr><td>
       <a href="#bottom"> nach unten </a>
  </td></tr>
</table>

<div class="eins">

  <hr size="6" noshade>
  <b>
  <h1> BILDER und BILDERRAHMEN </h1>
  <h2>
       Verkauf und Reparatur<br>
       im Wiener Fachgesch&auml;ft und Meisterbetrieb<br>
       der Familie Mustermann
  </h2>
  </b>
  <hr size="6" noshade>
      <a href="galerie.html" target="_blank"> Bildergalerie </a>
        
      <a href="#verkauf"> Produkte </a>
        
      <a href="#reparatur"> Reparatur </a>
        
      <a href="#angebote"> Aktuelles </a>
        
      <a href="#standort"> Standort </a>
  <hr size="6" noshade>

</div>

<div class="zwei">

<a name="verkauf"></a>
<h2><u>Verkaufsprodukte</u></h2>
Hier steht der Text zu den Verkaufsprodukten  . . . . .

<a name="reparatur"></a>
<br><br>
<h2><u>Reparaturservice</u></h2>
Hier steht der Text zum Reparaturservice  . . . . .

<a name="angebote"></a>
<br><br>
<h2><u>Aktuelle Angebote</u></h2>
Hier steht der Text zu den aktuellen Angeboten  . . . . .

<a name="standort"></a>
<br><br>
<h2><u>Standort</u></h2>
Hier steht der Text zum Standort  . . . . .

<br><br>
<a href="mailto:manfred.mustermann@aon.at">
 eMail: manfred.mustermann@aon.at </a>
<br><br>

</div>

<a name="bottom"></a>

<br>
<table style="background-color:antiquewhite; color:black; border: 2px solid gray; padding: 2px">
  <tr><td>
       <a href="#top"> nach oben </a>
  </td></tr>
</table>

</td>
</tr>
</table>
</body>
</html>
```

Teil C: Responsives Webdesign (RWD)

Unter **Responsivem Webdesign** (RWD) versteht man die automatische Anpassung der HTML-Objekte an den jeweiligen Geräte-Bildschirm (device-screen) von desktops, laptops, tablets und phones.
Die erste, wichtigste Anweisung dazu ist der HTML-Meta-Tag für den Darstellungsbereich (viewport):

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

Diese Anweisung ermöglicht die Breite des Bildschirms in geräteunabhängigen Pixeln zu nutzen. Dadurch können die Inhalte einer Seite neu angeordnet und so an verschiedene Bildschirmgrößen angepasst werden, egal ob an das kleine Display eines Mobiltelefons oder den großen Bildschirm eines Desktopcomputers.

Die zweite Anweisung ist ein CSS-Befehl, welcher die Breite des jeweiligen Geräte-Bildschirms abfragt und dann CSS-Regeln für die Benutzung angibt: **@media (query) { CSS-Rules … }**
Drei Beispiele dazu:

```
@media (max-width: 640px) {
    div { background-color: red; }
}

@media (min-width: 641px) and (max-width: 960px) {
    div { background-color: green; }
}

@media (min-width: 961px) {
    div { background-color: blue; }
}
```

Im ersten Beispiel wird die Hintergrundfarbe eines div-Bereiches auf rot gesetzt, wenn der Browser weniger als oder gleich 640 Pixeln breit ist.
Im zweiten Beispiel wird die Hintergrundfarbe eines div-Bereiches auf grün gesetzt, wenn der Browser zwischen 640 und 961 Pixeln breit ist.
Im dritten Beispiel wird die Hintergrundfarbe eines div-Bereiches auf blau gesetzt, wenn der Browser mehr als oder gleich 961 Pixeln breit ist.

Anmerkung: Auch können Anweisungen wie **@media only screen and (max-width: 640px) { . . . }** verwendet werden.

Das Grundkonzept des responsiven Webdesign ist die flexible Anpassung an die unterschiedlichen Bildschirmgrößen (screens).

Grundsätzlich werden in allen Browsern die verschiedenen Seiteninhalte ohne Verwendung von CSS immer untereinander dargestellt. Bei großen Bildschirmen werden die Inhalte mithilfe von CSS häufig nebeneinander angeordnet. Bei kleinen Bildschirmen müssen die Inhalte mithilfe von CSS hingegen untereinander angeordnet werden, damit nicht oftmalige Verschiebungen und Größenänderungen der Objekte notwendig sind. Daher sollte – wenn möglich – das Layout zuerst für kleine Bildschirme erstellt werden („mobile first“) und dann die Erweiterungen auf große Bildschirme. Der umgekehrte Weg ist meistens mühevoll und kompliziert. Aus diesen Grundsätzen ergeben sich nachfolgende Richtlinien.

- Verwendung relativer Maßeinheiten (anstelle von fixen):
  Angabe der Objektgrößen relativ zur Screenbreite (z.B. width: 100%).
  Angabe der Schriftgrößen relativ zur Standardschrift (z.B. font-size: 1.25em).
  Optimale Lesebarkeit ist mit ca. 80 Zeichen pro Textzeile gegeben,
  wobei {font-family: sans-serif; font-size: 16px} der Schriftstandard ist.

- Möglichst wenige absolute bzw. fixe Positionierungen:
  z.B. display: block; position: relative; float: left.

- Setzen von erforderlichen, inhaltsbezogenen Breakpoints:
  Wenn es der Inhalt einer Seite (page content) erfordert, dann können mit geeigneten „Media Queries“ neue Layout-Richlinien erzeugt werden (so genannte Breakpoints).

Im Folgenden sollen fünf einfache HTML-Programme das responsive Webdesign demonstrieren.

RWD-Demo 1 (respons.html)

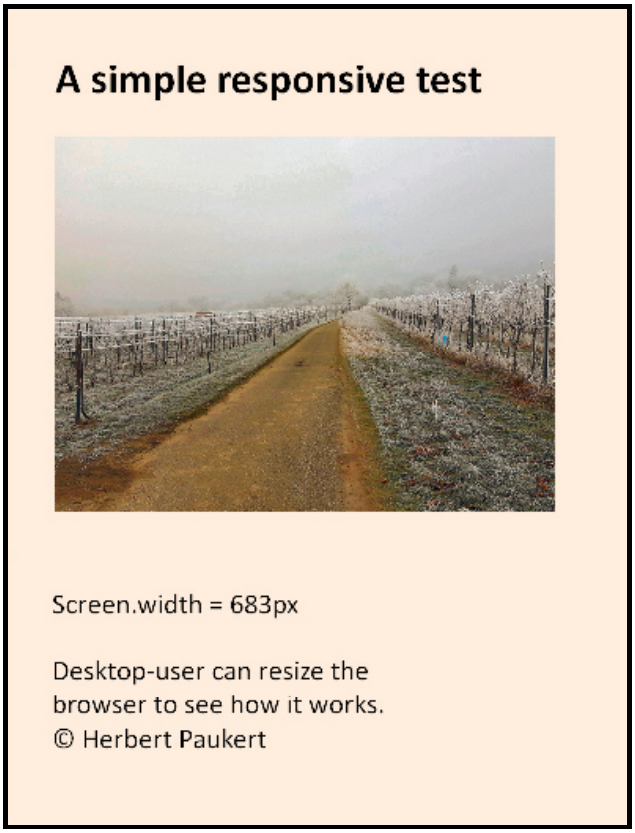
Die Abbildung zeigt die HTML-Seite auf einem großen Bildschirm.

Mit der Hilfe von drei „Media Queries“ werden für phones, tablets und desktops die Hintergrundfarben des Dokumentes, die Imagegrößen eines Bildes und auch die Schriftgröße verändert. Desktop-User können durch „Resizing“ des Screens die Wirkungen direkt beobachten.

```
<!DOCTYPE html>
<html>
<head>
<title>Respons</title>
<meta charset="ISO-8859-1">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<meta name="description" content="Respons">
<meta name="author" content="Paukert herbert">

<style>

* { margin: 0; padding: 10px; }

html { background-color: #DDEEFF; font-family: Calibri,sans-serif; font-size: 16px }
```

```
@media only screen and (max-width: 640px) {
  /* phones */
  html { background-color: #EEDDFF; font-size: 20px; }
  .myImg { width: 70%; }
}

@media only screen and (min-width: 641px) and (max-width: 960px) {
  /* tablets */
  html { background-color: #FFEEDD; font-size: 18px; }
  .myImg { width: 50% }
}

@media only screen and (min-width: 961px) {
  /* desktops */
  html { background-color: #DDEEFF; font-size: 16px; }
  .myImg { width: 30% }
}
}
</style>
</head>

<body onresize = "showSize()";>
  <br>
  <h2>A simple responsive test</h2>
  <image src="foto.jpg" class="myImg">
  <br>
  <p id="info"> </p>
  <p>
  Desktop-user can resize the<br>
  browser to see how it works.<br>
  &copy Herbert Paukert
  </p>
  <script>
     function showSize() {
       document.getElementById("info").innerHTML = "Screen.width = " + screen.width + "px";
     }
  </script>
</body>
</html>
```

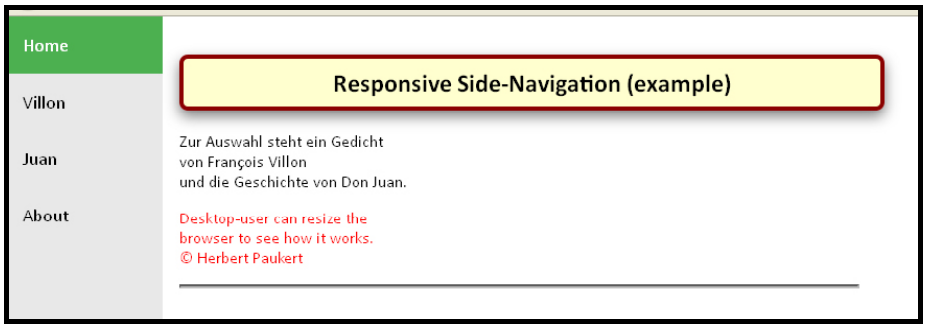
Die Abbildung zeigt die HTML-Seite auf einem kleinen Bildschirm.



RWD-Demo 2 (sidenavi1.html)

Der „**body**“ der HTML-Seite enthält einen dekorierten Titel und darunter normalen Text. Im linken Seitenteil ist ein vertikales Navigationsmenü (**sidebar**) fixiert. Dieses wird durch entsprechende CSS-Attribute erzeugt. Das Menü besteht aus vier Links, wobei der erste und der vierte deaktiviert sind.

Wenn der Bildschirm weniger als 640 Pixel breit ist (**@media screen and (max-width: 640px)**), dann werden alle Links auf die ganze Breite des Bildschirms ausgedehnt. Dabei sind alle Links untereinander angeordnet.



Die vorangehende Abbildung zeigt die Seite für große Bildschirme.
Die nachfolgende Abbildung zeigt die umgewandelte Seite für kleine Bildschirme.


```
<!DOCTYPE html>
<html>

<head>
<title>Side-Navigation 1</title>
<meta charset="ISO-8859-1">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<meta name="description" content="Side-Navigation 1">
<meta name="author" content="Paukert Herbert">

<style>
body   { margin: 0; font-family: Calibri, sans-serif; font-size: 16px; }
```

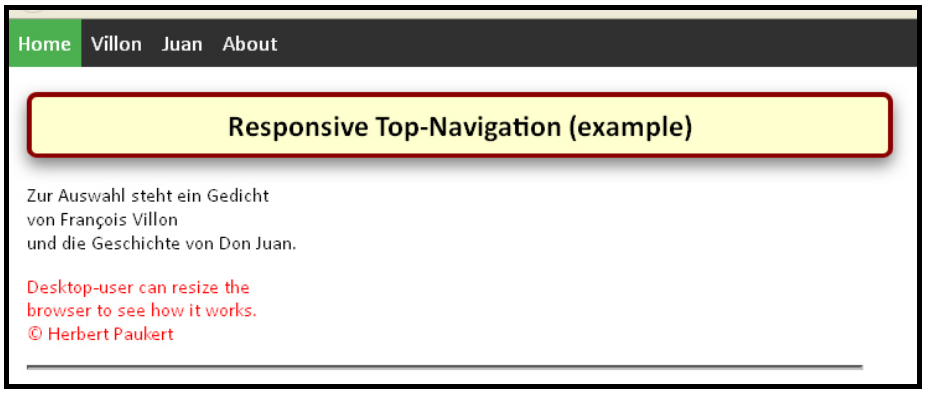
```
#titel {
  width: 55%;
  background-color: #FFFFD0;
  padding: 8px;
  border: 4px solid darkred; border-radius: 8px;
  box-shadow: 0 6px 12px 0 rgba(0,0,0,0.5);
  text-align: center;
}
#satz { color: red; }
hr {
  display: block;
  width: 55%;
  margin-left: 0;
  margin-right:0;
  border-width: 2px;
}
div.content {
  margin-left: 150px;
  padding: 16px;
  height: 100%;
}
.sidebar {
  font-size: 18px;
  margin: 0;
  padding: 0;
  width: 150px;
  background-color: #E8E8E8;
  position: fixed;
  height: 100%;
  overflow: auto;
}
.sidebar a {
  display: block;
  color: black;
  padding: 16px;
  text-decoration: none;
}
.sidebar a.active { background-color: #4CAF50; color: white; }
.sidebar a:hover:not(.active) { background-color: #505050; color: white; }

@media screen and (max-width: 640px) {
  div.content {margin-left: 0;}
  .sidebar {
    width:  100%;
    height: auto;
    position: relative;
  }
  .sidebar a {
    text-align: center;
    float: none;
  }
}
</style>
</head>

<body>
<div class="sidebar">
  <a href="#home" class="active">Home</a>
  <a href="villon.html">Villon</a>
  <a href="donjuan.html">Juan</a>
  <a href="#about">About</a>
</div>
<div class="content">
  <h2 id="titel">Responsive Side-Navigation (example)</h2>
  <p>
  Zur Auswahl steht ein Gedicht<br>
  von François Villon<br>
  und die Geschichte von Don Juan.
  </p>
  <p id="satz">
     Desktop-user can resize the<br>
     browser to see how it works.<br>
     &copy Herbert Paukert
  </p>
  <hr>
</div>
</body>
</html>
```

RWD-Demo 3 (topnavi.html)

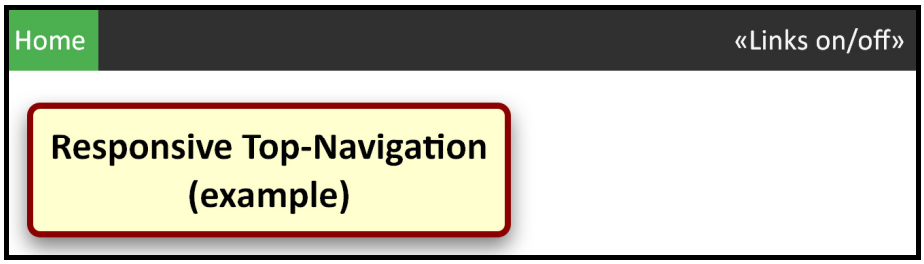
Der „**body**“ der HTML-Seite enthält einen dekorierten Titel und darunter normalen Text. Im Kopfteil der Seite ist ein horizontales Navigationsmenü (**topnav**) fixiert. Dieses wird durch entsprechende CSS-Attribute erzeugt. Das Menü besteht aus vier Links, wobei der erste und der vierte deaktiviert sind.

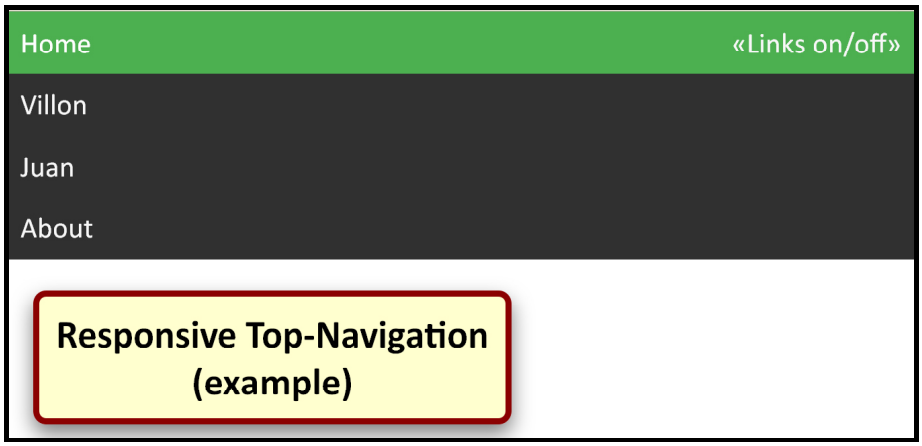
Wenn der Bildschirm weniger als 640 Pixel breit ist (**@media screen and (max-width: 640px)**), dann werden alle vier Links auf die ganze Breite des Bildschirms ausgedehnt und untereinander angeordnet. Dabei werden, ausgenommen den ersten Link, die restlichen Links nicht angezeigt. Stattdessen wird ein Wechselschalter (**Links on/off**) sichtbar. Mit seiner Hilfe können alle Links abwechselnd ein- oder ausgeblendet werden. Diese Klickfunktion wird durch einen JavaScript-Code erzeugt (**myFunction**).

Im vorliegenden Programm wird bei Unterschreiten einer bestimmten Screen-Breite das feste, horizontale Auswahlmenü in ein vertikales PopUp-Menü umgewandelt.



Die vorangehende Abbildung zeigt die Seite für große Bildschirme.
Die nachfolgenden Abbildungen zeigen die umgewandelte Seite für kleine Bildschirme.





```
<!DOCTYPE html>
<html>
<head>
<title>Top-Navigation</title>
<meta charset="ISO-8859-1">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<meta name="description" content="Top-Navigation">
<meta name="author" content="Paukert Herbert">

<style>
body   { margin: 0; font-family: Calibri, sans-serif; font-size: 16px; }
#titel {
  width: 50%;
  background-color: #FFFFD0;
  padding: 8px;
  border: 4px solid darkred; border-radius: 8px;
  box-shadow: 0 6px 12px 0 rgba(0,0,0,0.5);
  text-align: center;
}
#satz { color: red; }
hr {
  display: block;
  width: 50%;
  margin-left: 0;
  margin-right:0;
  border-width: 2px;
}

.topnav { overflow: hidden; background-color: #303030;  }
.topnav a {
  float: left;
  display: block;
  color: #F8F8F8;
  text-align: center;
  padding: 8px;
  text-decoration: none;
  font-size: 18px;
}
.topnav a:hover  { background-color: #D8D8D8; color: black; }
.topnav a.active { background-color: #4CAF50; color: white; }
.topnav .info    { display: none; }

@media screen and (max-width: 640px) {
  .topnav a:not(:first-child) { display: none; }
  .topnav a.info {
    float: right;
    display: block;
  }
}

@media screen and (max-width: 640px) {
  .topnav.responsive { position: relative; }
  .topnav.responsive .info {
    position: absolute;
    right: 0;
    top: 0;
  }
  .topnav.responsive a {
    float: none;
    display: block;
    text-align: left;
  }
}
</style>
</head>

<body>

<div class="topnav" id="myTopnav">
  <a href="#home" class="active">Home</a>
  <a href="villon.html">Villon</a>
  <a href="donjuan.html">Juan</a>
  <a href="#about">About</a>
  <a href="javascript:void(0);" class="info" onclick="myFunction()">
  «Links on/off»
  </a>
</div>
```

```
<div style="padding-left:16px">
  <h2 id="titel">Responsive Top-Navigation (example)</h2>
  <p>
  Zur Auswahl steht ein Gedicht<br>
  von François Villon<br>
  und die Geschichte von Don Juan.
  </p>
  <p id="satz">
  Desktop-user can resize the<br>
  browser to see how it works.<br>
  &copy Herbert Paukert
  </p>
  <hr>
</div>

<script>
function myFunction() {
  var x = document.getElementById("myTopnav");
  if (x.className === "topnav") { x.className += " responsive"; }
  else {  x.className = "topnav"; }
}
</script>

</body>
</html>
```

RWD-Demo 4 (homeres1.html)

Im nachfolgenden Programm wird eine einfache Homepage erzeugt, deren „**body**“ als Hauptcontainer aus verschiedenen Abschnitten (**divs**) besteht. Am Seitenanfang befindet sich ein „**header**“ und am Seitenende ein „**footer**“. Der Abschnitt dazwischen wird in **drei Teile** (**divs**) eingeteilt. In dem linken Teil befindet sich ein Menü (**menu**), welches aus 4 Links besteht. Der mittlere und der rechte Teil enthalten verschiedene Texte. Diese drei Elemente sind von einem Hüllelement (wrapper) umschlossen, dessen CCS-Attribut { **overflow: auto;** } bewirkt, dass auch über die Elemente hinaus ragende Inhalte dargestellt werden. Mit { **overflow: hidden;** } werden sie hingegen abgeschnitten.

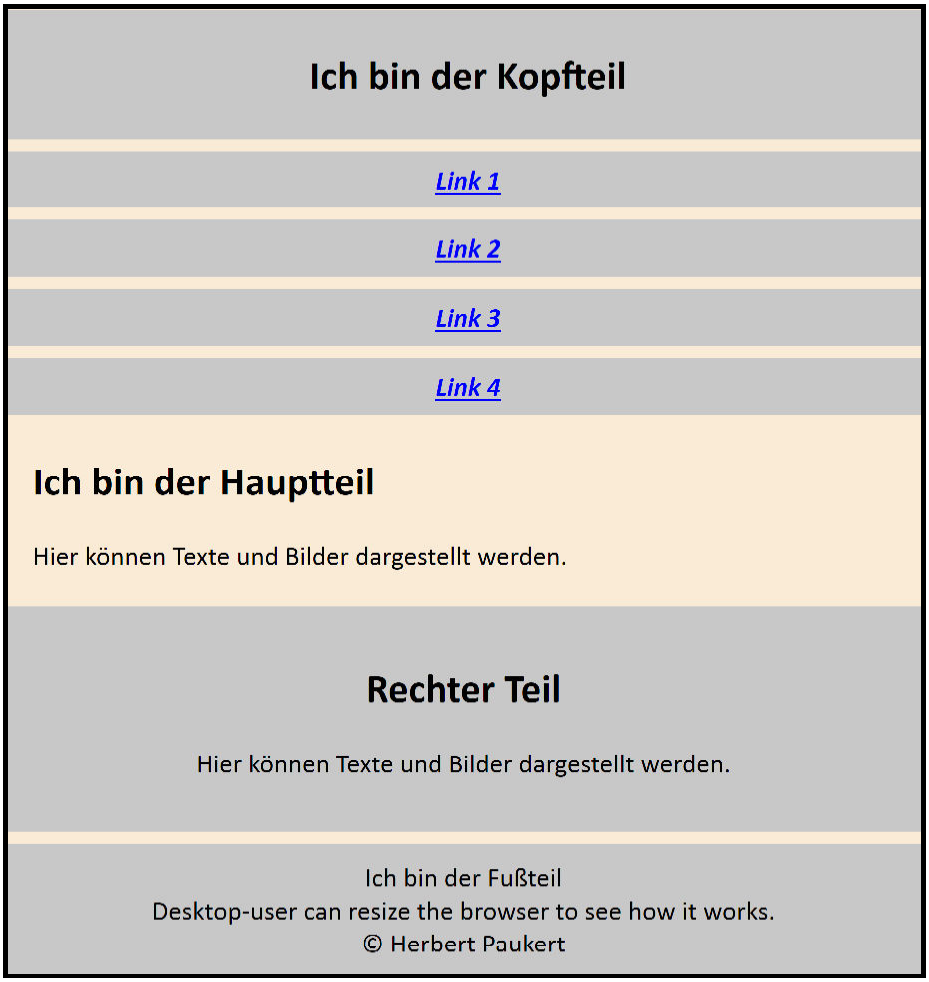
Der Universalselektor * wählt alle Elemente der vorliegenden Seite aus, und * **{ box-sizing: border-box; }** bewirkt, dass "padding" und "border" in die Breite (width) und Höhe (height) der Elemente eingerechnet werden.

Die vorangehende Abbildung zeigt die HTML-Seite für große Bildschirme.

Die beschriebene Einteilung der Seite gilt nur für Bildschirme mit einer Mindestbreite von 640 Pixel. Bei kleineren Screens wird jeder der drei Teile in untereinander liegende Blöcke umgewandelt, die alle genau so breit wie der Screen sind:

```
@media only screen and (max-width: 640px) {
  .menu, .main, .right { width: 100%; }
}
```

Die nachfolgende Abbildung zeigt die umgewandelte Seite für kleine Bildschirme.




```
<!DOCTYPE html>
<html>

<head>
<title>Homeres 1</title>
<meta charset="ISO-8859-1">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<meta name="description" content="Responsive Homepage 1">
<meta name="author" content="Paukert herbert">

<style>

* { box-sizing: border-box; }

body    { background-color: antiquewhite; color: black;
          font-family: Calibri,sens-serif; font-size: 18px; }

.header { background-color:#c8c8c8; padding:4px; text-align:center; }

.footer { background-color:#c8c8c8; padding:12px; text-align:center; margin-top: 8px; }
```

```
.wrapper {overflow: auto; }
.menu {
  float:left;
  width:25%;
  text-align:center;
}
.menu a {
  background-color:#c8c8c8;
  padding:8px;
  margin-top:8px;
  display:block;
  width:100%;
  color: blue;
  font-size: 18px;
  font-weight: bold;
  font-style: italic;
}
.main {
  float:left;
  width:50%;
  padding:0 20px;
}

.right {
  background-color:#c8c8c8;
  float:left;
  width:25%;
  padding:16px;
  margin-top:8px;
  text-align:center;
}

@media only screen and (max-width:640px) {
  /* For mobile phones: */
  .menu, .main, .right { width:100%; }
}
</style>
</head>

<body>

<div class="header">
  <h2>Ich bin der Kopfteil</h2>
</div>

<div class="wrapper">

  <div class="menu">
    <a href="#">Link 1</a>
    <a href="#">Link 2</a>
    <a href="#">Link 3</a>
    <a href="#">Link 4</a>
  </div>

  <div class="main">
    <h2>Ich bin der Hauptteil</h2>
    <p>Hier können Texte und Bilder dargestellt werden.</p>
  </div>

  <div class="right">
    <h2>Rechter Teil</h2>
    <p>Hier können Texte und Bilder dargestellt werden.</p>
  </div>

</div>

<div class="footer">
  Ich bin der Fußteil
  <br>
  Desktop-user can resize the browser to see how it works.
  <br>
  &copy Herbert Paukert
  <br>
</div>

</body>
</html>
```

RWD-Demo 5 (homeres2.html)

Zuerst einige Anmerkungen zu den unterschiedlichen CSS-Selektoren, die im Programm verwendet werden.

Grundsätzlich gibt es drei Arten von Selektoren für die Elemente: id-Selektoren (**#id**), Klassen-Selektoren (**.class**), Pseudo-Selektoren (**:**). Der Universalselektor * wählt alle Elemente der vorliegenden Seite aus, und *** { box-sizing: border-box; }** bewirkt, dass "padding" und "border" in die Breite (width) und Höhe (height) der Elemente eingerechnet werden.

Der Attribut-Selektor **[attr^="wert"]** wählt nur jene Elemente aus, deren Attribute "**attr**" mit dem Ausdruck "**wert**" beginnen.

Der Pseudo-Selektor **:not(.class)** wählt nur jene Elemente aus, welche das nachfolgende Klassenattribut nicht aufweisen.
Die Pseudo-Selektoren **.class:before** oder .**class:after** fügen einen Inhalt (content) vor oder nach das Element mit dem Klassenselektor ein.

Die Selektor-Attribute "**display: inline**", "**display: block**", "**display: none**" bestimmen, ob Elemente nebeneinander, untereinander oder überhaupt nicht angezeigt werden. Andere Attribute bestimmen die Stilmerkmale der Elemente (Abstände, Ränder, Größen, Farben, Dekorationen, usw.).

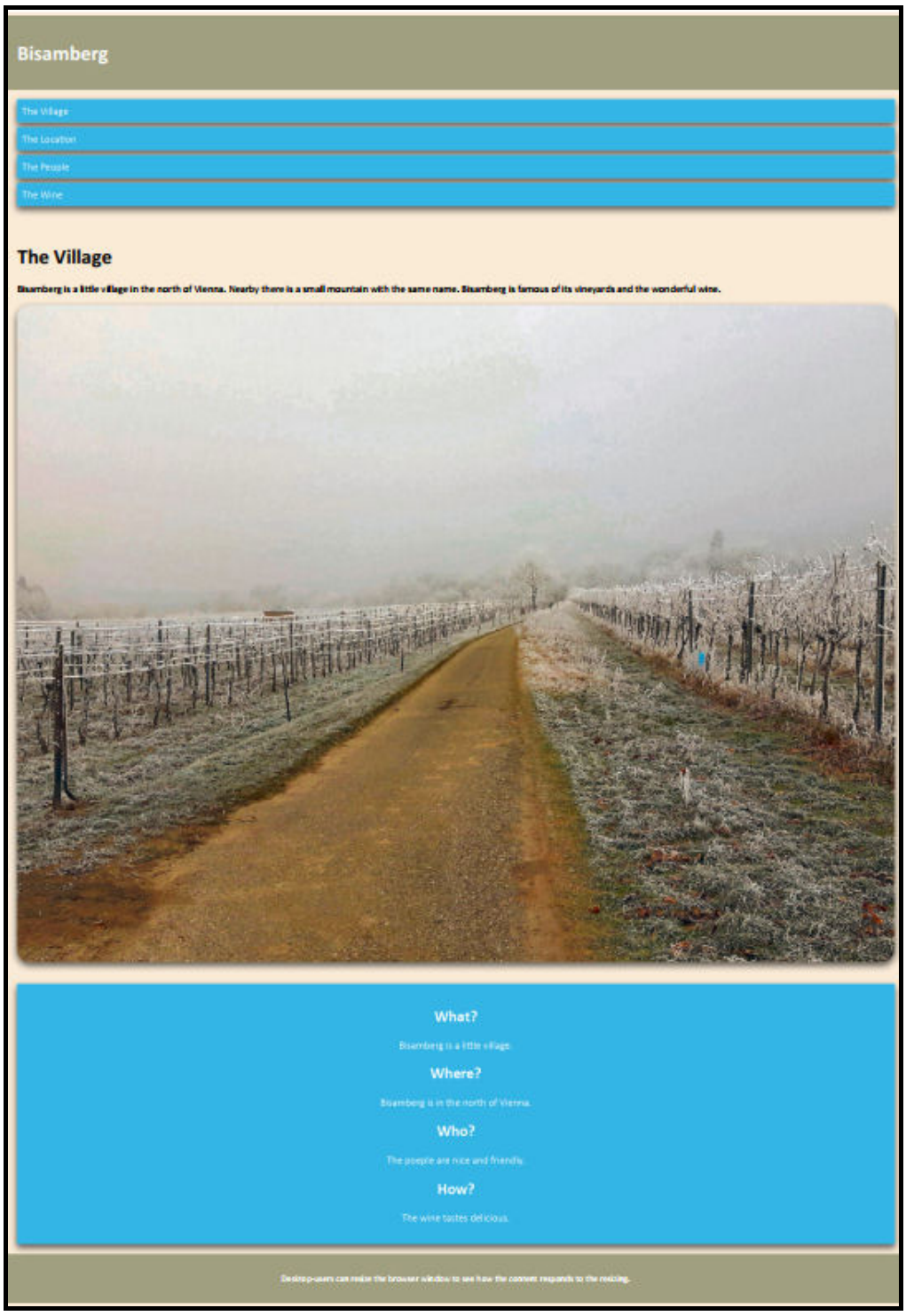
Die Selektoren können auch miteinander kombiniert werden.

Im nachfolgenden Programm wird eine einfache Homepage erzeugt, deren „**body**“ als Hauptcontainer aus verschiedenen Abschnitten besteht. Am Seitenanfang befindet sich ein „**header**“ und am Seitenende ein „**footer**“. Der Abschnitt dazwischen wird in **drei Spalten** (**cols**) eingeteilt. In der linken Spalte befindet sich eine Menü-Liste (**menu**), die aus 4 Links besteht. Die mittlere Spalte enthält einen Text und darunter ein Bild. In der rechten Spalte ist ein **aside**-Element eingebettet, wo ein Text steht.

Die vorangehende Abbildung zeigt die Seite für große Bildschirme.

Die beschriebene Spalteinteilung der Seite gilt nur für Bildschirme mit einer Mindestbreite von 640 Pixel (**@media only screen and (min-width: 640px)**). Bei kleineren Screens wird jede der drei Spalten in untereinander liegende Blöcke umgewandelt, die alle genau so breit wie der Screen sind (**[class*="col-"] { width: 100%; }**).

Die nachfolgende Abbildung zeigt die umgewandelte Seite für kleine Bildschirme.



```
<!DOCTYPE html>
<html>

<head>
<title>Homeres 2</title>
<meta charset="ISO-8859-1">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<meta name="description" content="Responsive Homepage 2">
<meta name="author" content="Paukert herbert">


<style>

* { box-sizing: border-box; }

html { font-family: Calibri, sans-serif; font-size: 16px; }

body { background-color: antiquewhite; }

.row::after {
  content: "";
  clear: both;
  display: table;
}

#myFoto {
   width: 100%;
   border-radius: 20px;
   box-shadow: 0 5px 10px rgba(0,0,0,0.25), 0 5px 10px rgba(0,0,0,0.50);
 }

[class*="col-"] {
  float: left;
  padding: 15px;
}

.header {
  background-color: #a0a080;
  color: #ffffff;
  padding: 15px;
}

.menu ul {
  list-style-type: none;
  margin: 0;
  padding: 0;
}

.menu li {
  padding: 8px;
  margin-bottom: 7px;
  background-color: #33b5e5;
  color: #ffffff;
  box-shadow: 0 5px 10px rgba(0,0,0,0.25), 0 5px 10px rgba(0,0,0,0.50);
}

.menu li:hover { background-color: #0088aa; }

.aside {
  background-color: #33b5e5;
  color: white;
  padding: 15px;
  text-align: center;
  font-size: 16px;
  box-shadow: 0 5px 10px rgba(0,0,0,0.25), 0 5px 10px rgba(0,0,0,0.50);
}

.footer {
  background-color: #a0a080;
  color: #ffffff;
  text-align: center;
  font-size: 14px;
  padding: 15px;
}

/* For mobile phones: */
[class*="col-"] { width: 100%; }
```

```
@media only screen and (min-width: 640px) {
  /* For tablets: */
  .col-s-1 {width: 8.33%;}
  .col-s-2 {width: 16.66%;}
  .col-s-3 {width: 25%;}
  .col-s-4 {width: 33.33%;}
  .col-s-5 {width: 41.66%;}
  .col-s-6 {width: 50%;}
  .col-s-7 {width: 58.33%;}
  .col-s-8 {width: 66.66%;}
  .col-s-9 {width: 75%;}
  .col-s-10 {width: 83.33%;}
  .col-s-11 {width: 91.66%;}
  .col-s-12 {width: 100%;}
   #myFoto {width: 65%}
}

@media only screen and (min-width: 640px) {
  /* For desktop: */
  .col-1 {width: 8.33%;}
  .col-2 {width: 16.66%;}
  .col-3 {width: 25%;}
  .col-4 {width: 33.33%;}
  .col-5 {width: 41.66%;}
  .col-6 {width: 50%;}
  .col-7 {width: 58.33%;}
  .col-8 {width: 66.66%;}
  .col-9 {width: 75%;}
  .col-10 {width: 83.33%;}
  .col-11 {width: 91.66%;}
  .col-12 {width: 100%;}
   #myFoto {width: 65%}
}
</style>
</head>

<body>
<div class="header">
  <h1>Bisamberg</h1>
</div>

<div class="row">
  <div class="col-3 col-s-3 menu">
    <ul>
      <li>The Village</li>
      <li>The Location</li>
      <li>The People</li>
      <li>The Wine</li>
    </ul>
  </div>

 <div class="col-6 col-s-9">
    <h1>The Village</h1>
    <p> Bisamberg is a little village in the north of Vienna.
        Nearby there is a small mountain with the same name.
        Bisamberg is famous of its vineyards and the wonderful wine. </p>
    <p style="text-align: center;”>
        <img src="foto.jpg" id="myFoto">
    </p>
  </div>

  <div class="col-3 col-s-12">
    <div class="aside">
      <h2>What?</h2>  <p>Bisamberg is a little village.</p>
      <h2>Where?</h2> <p>Bisamberg is in the north of Vienna.</p>
      <h2>Who?</h2>   <p>The poeple are nice and friendly.</p>
      <h2>How?</h2>   <p>The wine tastes delicious.</p>
    </div>
  </div>
</div>
<div class="footer">
  <p> Desktop-users can resize the browser window
      to see how the content responds to the resizing. </p>
</div>

</body>
</html>
```

E N D E