

HTML/XHTML – extensible hypertext markup language



Dieses Dokument wird unter folgender creative commons veröffentlicht:
<http://creativecommons.org/licenses/by-nc-sa/2.0/at/>

Inhaltsverzeichnis

1. DIE WICHTIGSTEN TAGS	2
1.1. GRUNDLAGEN	2
1.2. GRUNDAUSSTATTUNG AN TAGS.....	3
2. XHTML - AB JETZT BITTE NUN NUR NOCH SO!.....	6
2.1. DIE WICHTIGSTEN UNTERSCHIEDE.....	6
3. VERWEISE UND SPRUNGZIELE	8
3.1. SPRUNGZIELE ZU ANDEREN HTML-SEITEN	8
3.2. SPRUNGZIELE INNERHALB DER SELBEN SEITE	8
4. ELEMENTE DER SEITENGESTALTUNG	9
4.1. TABELLEN	9
4.2. FRAMES UND GETEILTE SEITEN	11
5. GRAPHIKEN UND BILDER.....	13
5.1. GRAPHIKGRUNDLAGEN	13
5.2. AKTIVE BEREICHE EINES BILDES FESTLEGEN	13
5.3. POSITIONIERUNG VON GRAPHIKEN	14
6. LISTEN	14
6.1. UNSORTIERTE LIST 	14
6.2. NUMMERIERTE LISTE 	15
7. CASCADING STYLE SHEETS.....	15
7.1. WAS IST CSS?.....	15
7.2. WIE DEFINIERT MAN STYLES?	15
7.3. WO DEFINIERT MAN STYLES?	17
7.4. BEISPIEL HINTERGRUNDGRAPHIK	18

Die *hypertext markup language* wurde für den Informationsaustausch im Internet entwickelt. Sie ermöglicht es, neben reinen Textinformationen auch Steuerbefehle zu übertragen, sogenannte **tags**. Diese **tags** dienen im Wesentlichen dazu, den Text zu formatieren. Sie spielen eine ähnliche Rolle wie die Steuerzeichen einer Textverarbeitung. Während aber die Steuerzeichen der Textverarbeitung spezifisch für die verwendete Software sind, sollen *HTML-Dokumente* von jedem Browser sofort richtig interpretiert werden. Die Browser der neuesten Generationen erlauben bereits die Erstellung von Dokumenten mit vergleichbarer Bequemlichkeit wie eine Textverarbeitung, Sie fügen aber sehr viele unnötige **tags** in den Text ein, sodass man die von ihnen erzeugten Dokumente zumeist nachbearbeiten muss.

Ein sehr gebräuchliches und extrem ausführliches **Nachschlagewerk**, ist **SelfHtml** von Stefan Münz.

Dieses Kompendium ist gratis vom Internet zu beziehen! → <http://www.selfhtml.teamone.de/>

1. Die wichtigsten Tags

1.1. Grundlagen

1.1.1. Tag-Konvention

Die meisten **tags** treten paarweise auf: ein **Starttag** und ein **Endtag** (ist durch einen / gekennzeichnet). Die tags sind in spitze Klammern eingeschlossen

```
<Starttag> ... </Endtag>  
<b> ... </b>  
<head> ... </head>
```

1.1.2. Der Grundaufbau eines HTML-Dokuments

Der **head** soll einen Titeltext haben, der vom Browser in der Kopfzeile dargestellt wird. Der **body** nimmt die Information auf, die auf der HTML-Seite tatsächlich angezeigt werden. Somit sieht das Grundgerüst einer Html Datei folgendermaßen aus:

```
<html >  
  
<head>  
<ti tle> ... </ti tle>  
</head>  
  
<body>  
...  
</body>  
  
</html >
```

1.1.3. Informationen für Suchmaschinen

Über Suchmaschinen können Internetbenutzer solche Seiten finden, die für sie von Interesse sind. Jede Suchmaschine greift dabei auf einen speziellen Block im HTML Code zu. Dieser sieht folgendermaßen aus, und kann bedenkenlos auch weggelassen werden!

```
<html >
<head>

<title> ... </title>

<META NAME="Author" CONTENT="Stefan Hagmann">
<META NAME="Description" CONTENT="HTML Einführung">
<META NAME="KeyWords" CONTENT="Tags, Fun, Graphik, Code,
  Workshop">
<META NAME="SiteInfo" CONTENT="Http://www.html.at">
<META NAME="Copyright" CONTENT="©2002 Stefan Hagmann">
<Meta http-equiv="Reply-To" content="sh@utv.at">

</head>
```

1.2. Grundausrüstung an Tags

Body Tag <body>

Innerhalb des Body tags können eine Vielzahl von Attributen angegeben werden. Im nachfolgenden Kasten sind die wichtigsten aufgelistet.

Farben	<pre><body bgColor=#RRGGBB text=#RRGGBB link=#RRGGBB vlink=#RRGGBB ></pre> <p>#RRGGBB stellt eine Farbe im Format Rot-Grün-Blau dar (Hexadezimalzahl von 00-FF)</p> <p>bgcolor ist die Hintergrundfarbe text ist die Textfarbe für das gesamte Dok. link ist die Farbe für noch nicht besuchten Verweisen vlink ist die Farbe bereits besuchter Verweise</p>
Hintergrundbild	<pre><body background = "Dateiname"></pre> <p>diese Graphik wird gekachelt dargestellt</p>

Überschriften <hx>

Die <hx> Befehle definieren bis zu sechs unterschiedlich große Überschriften. Die **Ausrichtung** kann man im Anfangstag angeben, mögliche Ausrichtungen sind **left**, **right**, **center** und **justify**.

```
<h1 align=center> ... </h1>
<h2 align=left> ... </h2>
<h3 align=right> ... </h3>
```

Horizontale Trennstriche <hr>

Der <hr> Befehl definiert einen horizontalen Trennstrich. Ein Endtag ist nicht nötig! Mit dem Attribut Size wird die Dicke der Linie angegeben. Mit Width die Breite.

```
<hr>
<hr size="10">
<hr width="80%">
```

Zeilenbruch

Mit Hilfe des Kommandos
 kann man einen festen Zeilenbruch einfügen.

Absätze <p>

Der <p> Befehl definiert einen abgeschlossenen Absatz. Nach dem Absatz wird eine Leerzeile erzeugt.

```
<p> ... </p>
<p> Das ist ein <br> Zeilenende</p>
```

Textauszeichnungen

Mit Hilfe von Textauszeichnungen kann man Textteile hervorheben oder einfach anders gestalten. Die Textauszeichnung betrifft immer nur den Text zwischen dem Anfangs- und dem Endtag.

Mit einem einfachen Editor muss man diese tags leider noch selbst eingeben. Modernere Editoren nehmen einem diese Arbeit ab.

```
<b> ... </b>      fett
<i> ...</i>      italic=kursiv
<u> ...</u>      underline
<big> ...</big>  groß
<small> ...</small> klein
<sup> ...</sup>  hochgestellt
<sub> ...</sub>  tiefgestellt
```

Schriftgröße

```
<font size="x"> ... </font>
x kann einen Wert von 1 bis 7 haben
```

Schriftfarbe	<code> ... </code>
Schriftart	<code> ... </code> hier sollte man nur gängige Schriftarten verwenden, da die Schriftart auf dem Zielcomputer installiert sein muss

Pseudotag `<div>`

Dieser tag ist ein Pseudotag, d.h. der tag erfüllt keine spezielle Aufgabe. Er dient dazu Attribute an Daten weiterzuleiten.

Als Beispiel soll eine Graphik (siehe dazu später) und eine Unterschrift zentriert dargestellt werden. Dazu umklammert man beide Befehle mit dem `<div>` tag und setzt das Attribut align.

```
<div align=center>
   <br />
  Das ist ein Bild
</div>
```

Pseudotag ``

Im Gegensatz zum `<div>` tag, wird im Anschluss kein Zeilenumbruch eingefügt

Einbinden eines Bildes ``

Ein Bild muss in einem, für das Internet vorgesehenem, Dateiformate (*.gif, *.jpg oder *.png) vorliegen. Es wird dann mit einem tag an der gewünschten Position des Dokuments eingefügt.

```


```

Kommentare

Möchte man einen Kommentar in den Html Code setzen, muss dieser in diesen Sonderzeichen eingeklammert werden.

```
<!--
  Das ist ein
  Kommentar
-->
```

Deutsche Sonderzeichen, Html Entities

Die Umlaute und das ß sind Zeichen, die nur im Deutschen vorkommen. Sie müssen daher im Internet speziell gekennzeichnet werden

Ein Html-Editor nimmt einem diese Arbeit aber ab! Aber Achtung: Wenn die Website in z.B. UTF-8 Kodierung ausgeliefert wird, entfallen diese Sonderzeichen!

ä = ä ;	Ä = Ä ;
ü = ü ;	Ü = Ü ;
ö = ö ;	Ö = Ö ;
ß = &szl i g;	

Weitere Sonderzeichen

Es gibt eine ganze Anzahl an Sonderzeichen die man einfügen kann. Hier soll nur ein kleiner Teil davon, und vor allem die Schreibweise, präsentiert werden. Eine ausführliche Anleitung findet man im Inrnet. In der Tabelle erkennt man das Sonderzeichen und den entsprechenden Html Code. Der Code beginnt immer mit & und wird beendet mit ;.

<	<	®	®	½	½
>	>	±	±	α	α
©	©	¼	¼	€	€

Harte Leerzeichen

Ein „normales“ Leerzeichen hat auf die Ausgabe keine Wirkung. Möchte man trotzdem ein Leerzeichen auf den Schirm bringen, muss man das entsprechende Sonderzeichen einfügen.

2. XHTML - ab jetzt bitte nun nur noch so!

2.1. Die wichtigsten Unterschiede

»Die XHTML-Familie ist der nächste Schritt in der Evolution des Internet. Wenn die Entwickler von Inhalten heute auf XHTML umsteigen, können sie in die XML-Welt mit allen dort gebotenen Vorteilen vordringen und dennoch darauf vertrauen, dass ihr Inhalt vorwärts- und rückwärtskompatibel ist.«

Es gibt mehrere, wesentliche Unterschiede zwischen HTML 4 und XHTML. Die wichtigsten Änderungen sind

Wohlgeformtheit

Das bedeutet, dass Elemente sich nicht mehr überkreuzen dürfen. In HTML 4.0 wurde Quelltext wie dieser noch interpretiert:

```
<p>Das letzte Wort ist <b>fett</p></b>
```

In XHTML führt diese falsche und auch nicht logische Schreibweise zu einem Fehler.

Kleinschreibung

Alle bisher bekannten Elemente und Attribute müssen **klein** geschrieben werden. Dies ist notwendig, da in XHTML zwischen Groß- und Kleinschreibung unterschieden wird.

Erforderliche Schluss-Tags

In HTML 4 war es noch möglich, für Elemente wie zum Beispiel p oder i kein Schluss-Tag anzugeben. Man sprach davon, dass diese Elemente implizit geschlossen wurden. Dies ist in XHTML nicht mehr erlaubt.

Attributwerte müssen immer in Anführungszeichen gesetzt werden. Durfte man früher zum Beispiel border=1 schreiben, muss in XHTML jedes Attribut in Anführungszeichen gesetzt werden, man muss also border="1" schreiben.

Keine Attributminimierung

In XHTML muss jedes **Attribut** einen Wert haben. Somit sind Angaben wie
`<input type="checkbox" checked />`
nicht mehr möglich. Es muss in XHTML
`<input type="checkbox" checked="checked" />`
heißen, auch wenn das
vielleicht ein wenig merkwürdig aussieht.

Leere Elemente

Leere Elemente müssen nun entweder ein End-Tag erhalten, also `<hr></hr>`, oder sofort geschlossen werden: `<hr />`. Das Leerzeichen ist nicht vorgeschrieben, sollte aber aufgrund besserer Kompatibilität zu älteren Browser gesetzt werden.

Andere Einbindung von Script- und Style-Elementen in den Dateikopf
Der HTML-Parser versucht, Script- und Style-Angaben in XHTML wie Markup zu lesen. Daher muss man derartige Angaben auf folgende Art schützen:

```
<script>  
<![CDATA[
```

```
...  
...  
]]>  
</scri pt>
```

3. Verweise und Sprungziele

3.1. Sprungziele zu anderen Html-Seiten

Fast schon ein Standardverfahren ist das laden/anzeigen von Html-Seiten mit einem Mausklick. Um das zu erreichen, muss man einen sogenannten Verweisanker festlegen.

Nehmen wir an, man möchte von der Datei Hautseite.html zur Seite Zweite_Seite.html springen. Dazu schreibt man folgenden Quelltext

```
<a href="Zwei te_Sei te.html " > Textpassage </a>
```

href weist dabei die zu ladende Seite aus, während Textpassage jener Text ist, den man am Bildschirm sieht (üblicherweise unterstrichen!).

Aber nicht nur ein Text kann als Sprungmarke dienen, auch ein Graphik kann das tun

```
<a href="Zwei te_Sei te.html ">  
    <i mg src="Bi ld.png" al t="" >  
</a>
```

3.2. Sprungziele innerhalb der selben Seite

Oft ist es wünschenswert innerhalb einer Seite zu springen, z.B. möchte man alle Kapitelüberschriften anspringen. Bevor man allerdings auf eine bestimmte Stelle referenzieren kann, muss diese markiert werden.

```
<a name="Marken_name" ></a>
```

Somit ist genau diese Stelle mit dem Marken_namen versehen, und kann angesprungen werden. Der Sprungbefehl sieht wie folgt aus

```
<a href="#Marken_name" > Textpassage </a>
```

Damit wird innerhalb der Seite auf die markierte Stelle gesprungen. Befindet sich die markierte Stelle innerhalb einer anderen Datei, kann auch diese sofort angesprungen werden

```
<a href="Zweite_Seite.html #Marken_name" > Textpassage </a>
```

Hier wird zuerst die Datei Zweite_Seite.html geöffnet, und im Anschluß an die Stelle mit Namen Marken_namen gesprungen.

4. Elemente der Seitengestaltung

4.1. Tabellen

4.1.1. Tabellen auf Seiten

Tabellen dienen dazu, Textteile an bestimmten Positionen in einer Zeile auszurichten: Man kann mit Tabellen Aufzählungen untereinander gleich einrücken, man kann aber Tabellen auch mit Rahmen ausstatten, damit sie ähnlich aussehen wie Tabellen eines Kalkulationsprogramms.

<pre><table>...</table></pre>	Anfang und Ende der Tabelle
<pre><tr>...</tr></pre>	eine Tabellenzeile
<pre><td>...</td></pre>	eine Zelle

Beispiel:

```
<table border="2">
<tr>
<td> Zelle A1 </td>
<td> Zelle B1 </td>
<td> Zelle C1 </td>
</tr>
<tr>
<td> Zelle A2 </td>
<td> Zelle B2 </td>
<td> Zelle C2 </td>
</tr>
</table>
```

Im Informationskasten sind alle notwendigen tags zusammengefasst, die man braucht, um eine Tabelle für den Browser zu definieren. Da man bei den doch sehr vielen notwendigen tags bald den Überblick verliert, empfiehlt es sich, den Tabellenquelltext sehr sorgfältig und strukturiert zu schreiben!

Das Beispiel in obigem Kasten stellt eine Tabelle dar. Beachte die eingerückten Zellen-Tags und die restliche Formatierung. So eine übersichtliche Schreibweise erleichtert im Nachhinein das lesen und überarbeiten einer Tabelle.

Gibt man im Anfangstag der Tabelle den Zusatz border an, so werden die Zellen umrahmt und die Tabelle bekommt folgendes Aussehen:

Zelle A1	Zelle B1	Zelle C1
----------	----------	----------

Zelle A2	Zelle B2	Zelle C2
----------	----------	----------

Wenn man eine Tabelle definiert, so ist es ratsam, in jeder Zeile die gleiche Anzahl von Zellen zu haben, da man sich nicht darauf verlassen kann, dass jeder Browser mit der fehlenden Zelle gleich umgeht. Möchte man eine Zelle weiter unterteilen, so kann die Zelle ja wieder eine weitere Tabelle aufnehmen.

Um eine Tabelle genauer zu gestalten, kann man sie noch in Kopf-, Daten- und Fußbereich unterteilen, die man jeweils unterschiedlich formatieren kann.

Weitere Tabellenattribute

Rahmen	<code><table border frame = box void above below hsides vsides lhs rhs></code>
Gitternetzlinien	<code><table rules = none rows cols groups all></code>
Breite	<code><th width = x> ... </th></code> legt die Breite für die ganze Spalte fest - steht daher sinnvollerweise im Kopf
Höhe	<code><tr height = x> ... </tr></code> legt die Höhe für die Zeile fest
cellpadding	<code><table cellpadding=4></code> gibt den Abstand des Textes vom inneren Tabellenrand an!

4.1.2. Zusammenfassen von Zellen

Möchte man mehrere Zellen zusammenfassen, muss man direkt im Quellcode den entsprechenden Befehl dazu geben.

`col span = x` fasst `x` Spalten in einer Zeile zusammen
`row span = x` fasst `x` Zeilen in einer Spalte zusammen

Beispiel:

```
<table>
<tr>
  <td colspan="2">
</tr>
<tr>
  <td>
  <td>
</tr>
</table>
```

In der ersten Zeile werden 2 Spalten zusammengefasst. Daher ist das zweite `<td>` `</td>` **unnötig**, und kann gelöscht werden. Dieser Code erzeugt folgende Tabelle

Beispiel:

```

<table>
<tr>
    <td rowspan="2"> </td>
    <td> </td>
</tr>
<tr>
    <td> </td>
</tr>
</table>

```

Hier werden jetzt zwei Zeilen verbunden. Wieder kann eine `<td>` `</td>` Definition gelöscht werden!

4.2. Frames und geteilte Seiten

Anmerkung: Frames sollten nicht mehr verwendet werden (W3C). Eine Neuerung werden die sogenannten XFrames werden!

Will man einen längeren Text auf einer Seite darstellen, so ist es zweckmäßig, die Seite in zwei Bereiche zu unterteilen. In einem Teil bringt man ein **Inhaltsverzeichnis** unter, das eigentlich nur **Sprungziele** in den Text der anderen Seite enthält, und im anderen Teil den Text selbst. Solche Unterteilungen auf der Seite nennt man **Frames**, wobei jeder Frame in einer eigenen Datei abgespeichert wird!

```

<frameset cols oder rows="xx,xx">
    <frame src="url" name="Fenstername">
    <frame src="url" name="Fenstername">
</frameset>

```

Das frameset wird im head der Verwaltungsdatei definiert, der body dieser Verwaltungsdatei wird nur angezeigt, wenn der Betrachter einen Browser verwendet, der keine frames darstellen kann - daher kann er üblicherweise leer bleiben! Im Anfangstag wird angegeben, ob das Fenster senkrecht oder waagrecht geteilt wird

(cols = Spalten oder rows = Zeilen). Dabei können die Angaben in Pixel oder Prozent gemacht werden.

Beispiele:

```
<frameset cols = "20%, 80%" >
```

teilt senkrecht in zwei Fenster

```
<frameset cols = "500, * " >
```

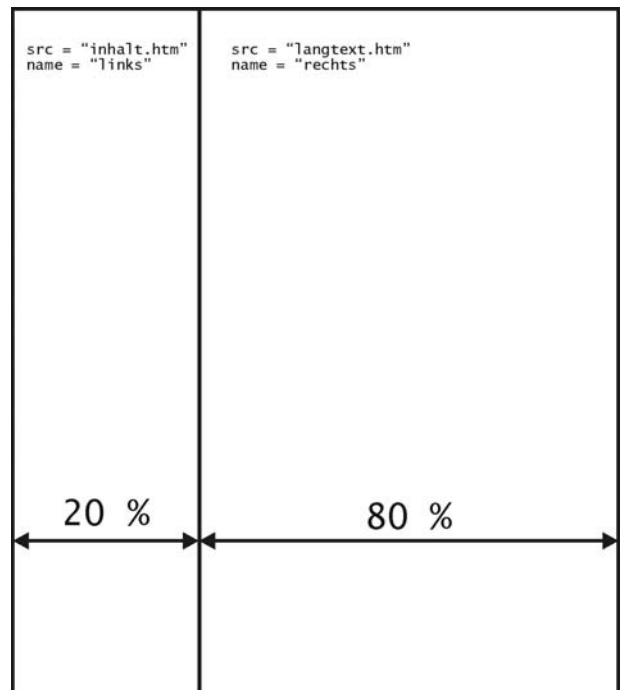
teilt auch senkrecht in zwei Fenster, das linke ist 500 Pixel breit, das rechte bekommt den Rest

```
<frameset rows = "200, * " >
```

teilt waagrecht in zwei Fenster

Für jedes definierte Fenster aus dem Anfangstag muss anschließend die **Quelldatei** angegeben werden (src = "url"). Der **Name** eines frames (name = "xxx") kann frei gewählt werden und wird bei Sprüngen als target angegeben. Eine mögliche frame-Definition würde etwa so aussehen

```
<html >
<head>
<frameset cols = "20%, 80%" >
  <frame src="inhalt.htm"
name="links" />
  <frame src="langtext.htm"
name="rechts" />
</frameset>
</head>
<body>
</body>
</html >
```



Im Prinzip kann man in die Html Datei auch nur die Frame-definition schreiben. Alle <html>, <head> und <bod> tags sind unnötig.

Im frame – tag gibt man mit src den Namen der Datei an, die in diesem frame dargestellt werden soll, den Namen des frame gibt man mit name an. Dieser ist wichtig für die Sprungbefehle, die wir nun erweitern müssen.

```
<a href="langtext.htm#Focke" > Fw109D Datenblatt </a>
```

zeigt die Datei langtext.htm im **aktuellen** frame an und scrollt gleich zum **Verweisanker** Focke. Das ist in unserem Beispiel aber nicht erwünscht, weil der Verweis im rechten frame dargestellt werden soll! Die erweiterte Formulierung lautet daher

```
<a href="langtext.htm#Focke" target="rechts" > Fw109D Datenblatt </a>
```

bewirkt, dass die Datei langtext im rechten frame dargestellt wird, und im linken bleibt unser Inhaltsverzeichnis. Die Erweiterung target wird ignoriert, wenn kein frame mit dem angegebenen Namen vorhanden ist. Es gibt einige vordefinierte target – Bezeichnungen, nämlich

_self	Anzeige innerhalb des aktuellen Frames
_parent	Anzeige in dem übergeordneten Frame
_top	Anzeige in dem obersten Frame
_blank	Erzeugt ein neues unbezeichnetes Fenster

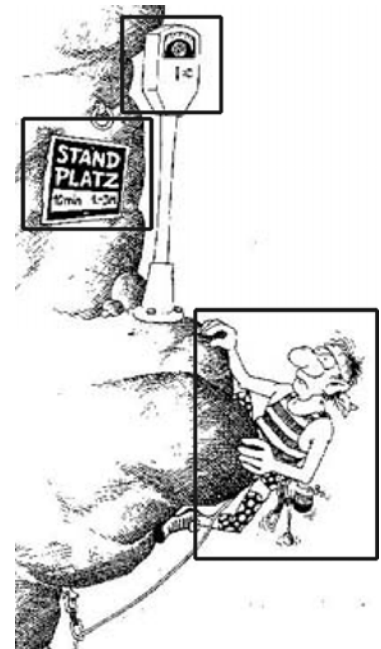
5. Graphiken und Bilder

5.1. Graphikgrundlagen

Die Graphikformat die bei Html-Seiten am besten Verwendung finden sind

- GIF – Format (*.gif)
- JPEG – Format (*.jpg)
- PNG – Format (*.png)

Jede Graphik kann man mit den gängigen Graphikprogrammen so verändern, dass sie den eigenen Wünschen entsprechen. Man sollte unbedingt darauf achten, wie viele Farben der Zielcomputer darstellen kann, ansonsten kann es zu Graphikfehlern bei der Darstellung kommen. Weiters ist zu bedenken: Je detailreicher eine Graphik ist, d.h. je größer die Graphikdatei ist, umso länger dauert es die Graphik über das Internet zu laden. Deshalb gilt: **Graphikdateien so klein wie möglich halten!**



5.2. Aktive Bereiche eines Bildes festlegen

Eine Graphik kann verwendet werden um zu verschiedenen Dateien zu verzweigen. Dazu verwendet man den üblichen Verweisbefehl und ersetzt den eingeschlossenen Text durch eine Graphik.

```
<a href="Kl ettern. htm" >  
  
</a>
```

Klickt nun ein Benutzer die Graphik an, wird zur entsprechenden Datei verzweigt! Aber es geht noch besser! Innerhalb einer Graphik ist es möglich **aktive Bereiche** festzulegen. Das sind solche Bereiche, die zu verschiedenen Dateien verzweigen. Im Beispielbild sind drei aktive Bereiche eingezeichnet.

Um einen **aktiven Bereich** zu kennzeichnen, muss man die Koordinaten (in Pixel) der linken oberen Ecke und der rechten unteren Ecke kennen (verwende ein Graphikprogramm).

Alle nötigen Informationen werden in einer so genannten **Map** festgelegt. Eine mögliche Definition könnte folgendermaßen aussehen

```
<map name="Climb" >
  <area shape=rect coords="120,180,180,250"
        href="Stand.htm" />
  <area shape=rect coords="140,100,180,160"
        href="Uhr.htm" />
  <area shape=rect coords="140,240,200,300"
        href="Climber.htm" />
</map>
```

Diverse Programme erledigen diese doch recht mühselige Arbeit einem ab!

Nun müssen wir nur noch die **Map** der entsprechenden Graphik zuordnen. Dies geschieht im `img`-Tag:

```

```

Beachte: Der Name der Map muss mit einem `#` eingeleitet werden!!

5.3. Positionierung von Graphiken

Möchte man Graphiken an einen festen Ort binden, sollte man auf Tabellen zurückgreifen. Das bedeutet zwar einen Mehraufwand an Arbeit, dafür erhält man die besten Resultate. Ein bisschen einfacher ist es den `align` Befehl zu verwenden.

```

```

Damit kann man erreichen, dass der Text um die Graphik herumfließt, bzw. dass die Graphik den Text nicht beeinflusst. Probiere es aus!

6. Listen

6.1. Unsortierte List ``

Eine unsortierte Liste hat ein bestimmtes Aufzählungszeichen, das durch das Attribut `type="disc|square|circle"` festgelegt wird. Der tag `` `` legt das entsprechende Listenelement fest.

```
<ul type="circle">
  <li>Probieren geht &uuml;ber Studieren</li>
  <li>Liebe geht über Triebe</li>
  <li>Tante f&auml;hrt &uuml;ber Kante</li>
</ul>
```

6.2. Nummerierte Liste

Eine solche Liste wird durchnummeriert. Die Nummerierung beginnt dabei bei start. Den Typ der Aufzählung legt man wiederum mit type="1|a|i|I" fest.

```
<ol start="1" type="i">
  <li>bei Anette vorbeischauen</li>
  <li>bei Bianca vorbeischauen</li>
  <li>bei Christine vorbeischauen</li>
</ol>
```

7. Cascading Style Sheets

7.1. Was ist CSS?

Mit **CSS** kann man einzelnen **Tags** ein bestimmtes Aussehen verleihen. Möchte man z.B: alle <h1> Überschriften in einer gewissen Schriftart mit 14pt und in Rot darstellen, müsste man das bei jedem Tag angeben!! → eine immense Arbeit.

So definiert man ein **Style Sheet**, in dem alle Anpassungen für <h1> vorgenommen sind, und braucht dann lediglich auf das Style Sheet verweisen.

Man erreicht daher mit CSS

- Einheitliches Aussehen der Überschriften und Absätze innerhalb einer HTML-Seite.
- Einheitliches Aussehen der Überschriften und Absätze innerhalb einer kompletten WebSite bestehend aus mehreren HTML-Dokumenten.
- Wesentlich mehr und flexiblere Gestaltungsmöglichkeiten als "pures" HTML.
- Genaues Positionieren von Bereichen (Text, Graphik), was pures HTML meist nur über den Umweg von Tabellen, aber in eingeschränktem Maße, zulässt.
- Schlankere HTML-Dateien und damit geringere Übertragungszeiten trotz besserer Gestaltungsmöglichkeiten.

7.2. Wie definiert man Styles?

7.2.1. Überschreiben von Tags

Man gibt jedem Tag eine bestimmte Formatierung. Möchte man z.B. den <a> Tag eine bestimmte Farbe geben, schreibt man

```
a{color: #AACC00}
```

Damit werden alle <a> Tag's in dieser Farbe dargestellt. Die Eigenschaften der Tags werden somit praktisch *überschrieben*.

Weitere Beispiele wären

```
H1 { FONT-FAMILY: Arial; color: #ffCC00}
H4 { FONT-FAMILY: Helvetica}
P { FONT-FAMILY: Arial, sans-serif; FONT-STYLE: italic}
```

Eine Eigenschaft ein und desselben Tags kann auch mehrmals festgelegt werden, wobei **cascading** bedeutet, dass eine Definition jede vorherige überschreibt – kurz: **die letzte gilt!**

Kommentare werden mit /* eingeleitet und mit */ geschlossen.

7.2.2. Definieren von Klassen

Grundsätzlich kann man nur die in HTML zur Verfügung stehenden Tags mit Hilfe von CSS umdefinieren. Günstig wäre es, wenn man sich für spezielle Zwecke seine eigenen Tags einführen könnte. Wenn also das vorhandene Repertoire von HTML-Tags zu klein ist, so können Varianten davon, sogenannte **Klassen**, bestimmt werden.

Dazu wird in der zentralen Definition zuerst das entsprechende Tag notiert und daran durch einen Punkt getrennt der **Klassennamen**.

```
<style>
  body,html { /* Tag ohne Klasse */
    font-family: Arial; font-size: 12pt;
    text-align: justify;
  }
  p.klein {
    font-size: 10pt;
  }
  .rot { /* bezieht sich auf alle Tags */
    color: red;
  }
  all.blau {
    color: blue;
  }
  .gross {
    font-size: 130%;
  }
</style>
```

BODY, HEAD, P,
H1 ...

BEZIEHT SICH AUF DEN ENTSPRECHENDEN TAG.

SOLL DAS AUSSEHEN FÜR MEHRERE TAGS GELTEN, TRENNT MAN DIESE DURCH EINEN BEISTRICH.

P. KLEIN

GILT FÜR <P> TAGS DIE MIT DER KLASSE CLASS="KLEIN" GEKENNZEICHNET SIND.

```

BSP:
<P CLASS="KLEIN" >
  HALLO
</P>
. ROT      GILT FÜR ALLE TAGS DIE MIT CLASS="ROT" GEKENNZEICHNET
          SIND.
BSP:
<DIV CLASS="ROT" >
  HALLO
</DIV>
#GROSS    GILT FÜR ALLE TAGS DIE MIT ID="GROSS" GEKENNZEICHNET SIND.
BSP:
<DIV ID="GROSS" >
  HALLO
</DIV>

```

Im Dokument muss dann beim entsprechenden Tag als Attribut die zugrundeliegende Klasse angegeben werden.

```

<p class="klein">Kleiner geschriebener Text</p>
<h1 class="rot">Rote Überschrift</h1>
<h3 class="rot">Kleine rote Überschrift</h3>

```

Wenn in der zentralen Definition vor der Klassebezeichnung **kein Tag** angegeben ist oder das **Schlüsselwort** all steht, so spricht man von einer leeren Klasse und die Angaben beziehen sich auf alle Tags.

7.3. Wo definiert man Styles?

7.3.1. Im HTML – Dateikopf

Die häufigste Variante Style-Angaben für verschiedene Tags zu definieren ist im Head-Bereich der HTML-Datei, dazu werden die Angaben innerhalb eines eigenen **Style-Tags** getätigt.

```

<html >
<head>
<title> ... </title>

<style type="text/css">
  body{background-color: yellow; }
  p, h1, h2, h3{
    font-family: Arial ;
    color: #00CC00
  }
  h1{font-face: bold ;}
</style>

</head>
<body>

...

```

```
</body>
<html >
```

Das Tag, dessen Eigenschaften (für das gesamte Dokument) verändert werden soll, wird angegeben, **mehrere Tags**, die dieselben Eigenschaften haben sollen, werden durch Beistriche getrennt.

In geschwungenen Klammern werden dann die entsprechenden Angaben gemacht. Die entsprechenden Eigenschaften werden nach einem Doppelpunkt auf einen passenden Wert gesetzt, mehrere verschiedene mögliche Werte werden durch **Beistrich** getrennt, mehrere zu ändernde Eigenschaften werden durch **Strichpunkte** getrennt.

Besteht ein Wert einer Eigenschaft (zB Schriftart) aus mehreren durch Leerzeichen getrennten Worten, so muss diese unter Anführungszeichen gesetzt werden.

7.3.2. Styles in einer separaten Datei definieren

Dieselben zentralen Angaben können auch in einer eigenen Datei abgelegt werden, wobei die Syntax dieselbe ist, wie im Head-Bereich einer HTML-Datei innerhalb des Style-Tags.

```
body{background-color: yellow}
p, h1, h2, h3{font-family: Arial; color: #00CC00}
h1{font-face: bold}
```

Die Datei wird mit folgendem Befehl im Head-Bereich der HTML-Datei **eingebunden**

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

Übrigens kann ein und dieselbe CSS-Datei für **mehrere HTML-Dateien** verwendet werden, und so für ein einheitliches Aussehen einer ganzen Site gesorgt werden.

7.4. Beispiel Hintergrundgraphik

Oft möchte man eine Graphik hinter den Text legen. In manchen Fällen soll sich die Graphik immer in der Mitte des Schirmes befinden, und soll sich nicht mit dem Text mitbewegen. Mit CSS kein Problem!

```
<style type="text/css">
body{
  background-image: url (image.gif);
  background-repeat: no-repeat;
  background-attachment: fixed;
  background-position: 30px 20px;
}
</style>
```

Mit `background-image:url(???)`; wird die Graphik festgelegt. Das Attribut `background-repeat:no-repeat`; bewirkt, dass sich das Bild nur ein mal darstellt, anderenfalls wird es gekachelt angezeigt. Mit `background-attachment:fixed`; bleibt die Graphik fixiert im Hintergrund stehen.