HTML-Grundlagen

Wenn in den Medien das Wort Internet auftaucht, dann ist meistens das *World Wide Web* (WWW) gemeint. Der große Erfolg dieses Dienstes, den es erst seit dem Ende der achtziger Jahre gibt, beruht auf seiner einfachen Bedienbarkeit.

Das Zauberwort heißt *Hypertext*. Im Prinzip stellen sich alle Angebote innerhalb des WWW als eine riesigen Ansammlung von Seiten dar, die untereinander verbunden sind. Über die Verknüpfungen, die *Hyperlinks* kann man von einer Seite zur anderen *surfen.* Die Verknüpfung geschieht über hervorgehobene Schlüsselwörter, die auf andere Seiten verweisen. Klicke ich ein derartiges Schlüsselwort an, so wird die Seite geladen, auf die verwiesen wurde.

Für die Nutzung des WWW benötigt man eine entsprechende Client-Software, die normalerweise als Browser (engl. to browse: stöbern, schmökern) bezeichnet wird. Ein weit verbreiteter Brwoser ist der *Firefox*, der unter der Adresse http://www.mozilla.com/ kostenlos für viele Betriebssysteme zur Verfügung steht.

1. Grundlagen

Das was der Benutzer aus dem Internet lädt ist eine Art Programm in einer relativ primitiven Seitenbeschreibungssprache, der *Hypertext Mark-Up Language* (HTML). Der Browser setzt dann dieses Programm in die entsprechende graphische Darstellung um. Da es derartige Browser für sehr verschiedene Betriebssystem und von unterschiedlichen Herstellen gibt, sind die Ergebnisse nicht immer gleich. Die Seitenbeschreibungssprache wird nämlich so schnell erweitert, dass nicht alle Browser alle Konstrukte der Sprache beherrschen. Am korrektesten arbeitet zur Zeit wohl Opera gefolgt vom Mozilla Firefox.

Seit einiger Zeit befindet sich das WWW und damit HTML in einem deutlichen Wandel. Die *Cascading Style Sheets* (CSS) werden immer wichtiger. Grob gesagt erlaubt CSS die Trennung zwischen Inhalt und Layout. Der Inhalt steckt in der HTML-Seite, das Layout im Style Sheet. In der HTML-Seite erfolgen also nur logische Textauszeichungen der Art „dies ist eine Überschrift“, oder „dies ist ein Absatz“. Wie groß eine Überschrift ist, und auf welcher Schriftart sie basiert, legt das Style Sheet fest.

Für das Erstellen von HTML-Seiten gibt es sehr viele Programme, sowohl kommerzielle, als auch kostenlose. Da bleibt die Frage, warum man sich überhaupt mit HTML beschäftigen muss. Ein sehr wichtiger Grund ist, dass heute kaum noch jemand einfache HTML-Seiten erstellt. Sehr oft werden *Content Management Systeme* (CMS) verwendet oder HTML-Code mit Programmiersprachen wie PHP gemischt. Selbst für den Umgang mit einem sehr einfache bedienbarem CMS wie dem MediaWiki kann es sinnvoll sein Grundlagen von HTML und CSS zu beherrschen.

1.1. HTML

Die folgende Seite hat noch einen sehr einfachen Aufbau, ist aber schon eine vollwertige HTML-Seite:

(Beispielseite im Mozilla-Firefox auf einem Linux-Rechner)

Zu dieser Seite führt das folgende HTML-Programm:

```
<html>
<head>
  <title>Dies ist der Titel</title>
</head>
<body>

<h1>Dies ist eine &Uuml;berschrift</h1>
Alle weiteren Zeichen geh&ouml;ren zum normalen Text.
</body>
</html>
```

Das HTML-Programm bzw. den Quelltext einer Seite kann man sich ansehen, indem man im Firefox im Menü *Ansicht* den Punkt *Seitenquelltext anzeigen* auswählt. Es öffnet sich ein Editor, der den Quelltext recht übersichtlich darstellt.

In dem Listing fallen mehrere Schlüsselwörter auf, die in spitze Klammern gesetzt wurden. Es sind die so genannten *Tags*, die Befehle von HTML. Viele Tags bestehen aus zwei Teilen, einem *Anfangs-Tag* und einem *End-Tag*. Der End-Tag wird durch einen Schrägstrich nach der ersten spitzen Klammer gekennzeichnet.

<title> ist ein zweiteiliger Tag. Alle Zeichen zwischen *<title>* und *</title>* gehören zum Titel der Seite und werden in der Titelzeile des Browserfensters angezeigt.

Auch *<h1>* ist ein solcher Tag. Alle Zeichen zwischen *<h1>* und *</h1>* werden als Überschrift dargestellt. Insgesamt gibt es sechs vordefinierte Größen für Überschriften, *<h1>....<h6>*. Dabei steht h1 für eine große Überschrift, h6 für eine kleine. Der Quelltext ist in mehrere Teile gegliedert. Die große Klammer besteht aus *<html>*, dies kennzeichnet den Beginn des Textes. Als letzter Tag muss immer *</html>* auftauchen.

Der erste Teil eines Textes ist dann der Kopf, der durch *<head>* eingeleitet und *</head>* beendet wird. Im Kopf wird der Titel festgelegt und auch Dinge wie die später anzusprechenden Styles. Auf den Kopf folgt der Rumpf, der durch *<body>* eingeleitet und durch *</body>* beendet wird. Im Rumpf-Bereich stehen die eigentlichen Inhalte der Seite.

Im Beispiel taucht noch der Tag * * auf, der ein Absatzende markiert. Genau genommen sollten Absätze durch * * und * * eingerahmt werden, was aber noch selten geschieht.

Was man noch an diesem Beispiel sieht, ist die Tatsache, dass auch Umlaute umschrieben werden müssen. Der Text ist in 7Bit-ASCII gehalten, da gibt es keine Umlaute. Es werden die folgenden Umschreibungen benutzt:

```
ä = &auml;
Ä = &Auml;
ö = &ouml;
Ö = &Ouml;
ü = &uuml;
Ü = &Uuml;
ß = &szlig;
```

In einem ordentlichen HTML-Editor würde die hier beschriebene einfache HTML-Seite automatisch um weitere Elemente ergänzt. Das hängt damit zusammen, dass es inzwischen sehr verschiedene Versionen von HTML bzw. XHTML gibt. Man gibt daher dem Browser mit der Zeile

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
```

an, mit welcher Version man arbeitet.

1.2. Zeichensätze

Seit einiger Zeit verändert sich auch der Umgang mit den Zeichensätzen am Computer. Lange Zeit war der *American Standard Code for Information Interchange* (ASCII) üblich, ein Zeichensatz mit maximal 128 Zeichen.

Schon unsere Umlaute machen da ein Problem, sie sind einfach nicht vorgesehen. Daher gibt es von diesem Zeichensatz Erweiterungen mit maximal 256 Zeichen. Da auch das nicht für die Sonderzeichen aller Sprachräume ausreicht, gibt es mehrere Variationen bei denen die spezifische Zeichen eines Sprachraumes vorhanden sind.

Für diese Zeichensätze gibt es eine Normierung durch die *International Organization for Standardization* (ISO), die ISO-8859. Für den Westeuropäischen Sprachraum gültig ist der Zeichensatz ISO-8859-1 oder besser ISO-8859-15, dann ist auch das €-Zeichen berücksichtigt.

Je nach Zeichensatz gehört also zum gleichen Code ein anderes Zeichen. Daher ist es sinnvoll in einer HTML-Seite anzugeben, mit welchem Zeichensatz man bei der Erstellung gearbeitet hat, falls nicht alle Sonderzeichen korrekt in HTML-Umschreibungen umgewandelt wurden.

```
 <meta content="text/html; charset=ISO-8859-1"  http-equiv="content-type">
```

Durch diese Zeile weiss der Browser, wie er Sonderzeichen zu interpretieren hat. Seit einiger Zeit gibt es Bemühungen die verschiedenen Zeichensätze in einem Unicode genannten Standard zusammen zu fassen. Für die Speicherung von Unicode gibt es das *Unicode Transformation Format* (UTF). Üblich ist heutzutage UTF-8, wobei ein einzelner Buchstabe in mehreren Byte gespeichert werden muss, damit UTF-8 alle Zeichen umfassen kann.

Bei modernen Betriebssystemen ist generell eine Umstellung des Zeichensatzes auf Unicode er-

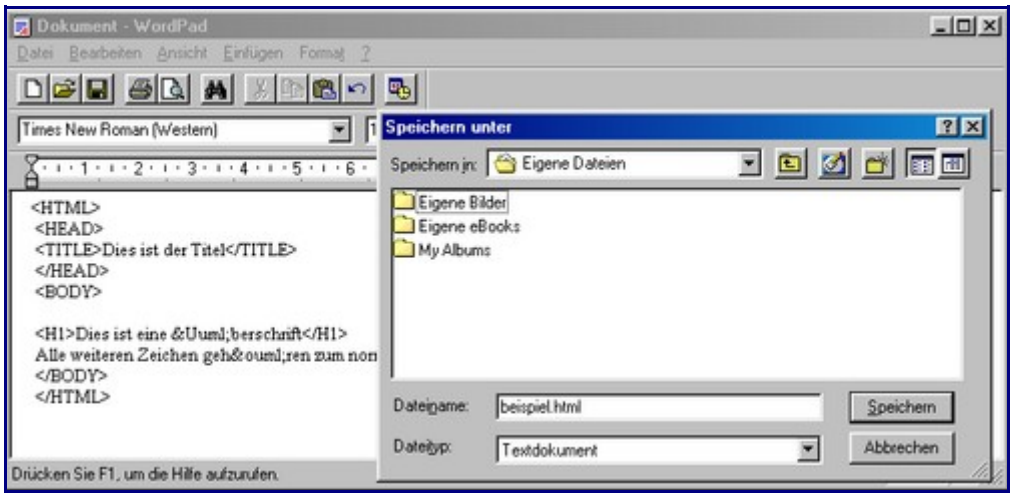
folgt, wenn auch die zukünftigen Windows-Arten mit Unicode arbeiten, dann sollten Zeichensatzprobleme der Vergangenheit angehören. Solange diese Umstellung nicht abgeschlossen ist sollte man Umlaute und andere Sonderzeichen in die Ersatzdarstellung bringen, also statt ä besser ä schreiben.

Die komplette Beispielseite vom Anfang hätte aktuell also folgenden Aufbau.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
<head>
  <meta content="text/html; charset=ISO-8859-1"  http-equiv="content-type">
  <title>Dies ist der Titel</title>
</head>
<body>
<h1>Dies ist eine &Uuml;berschrift</h1>
Alle weiteren Zeichen geh&ouml;ren zum normalen Text.
</body>
</html>
```

Das ist eigentlich schon alles, was man über HTML wissen muss.

1.3. Editoren



Für das Erstellen von HTML-Seiten langt im Prinzip ein einfacher Texteditor, wie WordPad unter Windows bzw. KWrite unter Linux. Wichtig ist immer nur, dass der Quelltext als unformatierter Text (Dateityp Textdokument) gespeichert wird. WordPad hat leider die unangenehme Eigenschaft zusätzlich noch die Erweiterung ".txt" an den Dateinamen anzuhängen. Da wird dann aus "Seite1.html" beim Speichern plötzlich "Seite1.html.txt". Verhindern kann man das, indem man den Dateinamen in Anführungsstriche setzt.

Da WordPad kein besonders mächtiger Editor ist greifen viele HTML-Autoren zu spezielleren Werkzeugen. Dabei gibt es eine breite Palette von kommerziellen Werkzeugen, die oft aber eine enorme Einarbeitung erfordern und relativ teuer sind.

Ein weit verbreiter HTML-Editor ist Dreamweaver von Adobe bzw. Macromedia. In der Regel kommt man auch mit einem kostenfreien Produkt wie dem NVU Composer gut zurecht.

2. Einige HTML Elemente

HTML ist eine ganz einfache Sprache. Alle Elemente dieser Sprache, die Tags, sind an den spitzen Klammern leicht zu erkennen.

2.1 Grundgerüst

Das Grundgerüst eines HTML-Programms ist zwar an einigen Stellen erweitert worden, ist aber immer nach folgendem Schema aufgebaut:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
   <head>
       <title>Text für Titelleiste</title>
   </head>
   <body>
               ... Datei-Inhalt ...
   </body>
</html>
```

In der Anfangszeit des WWW hat man kaum mehr benutzt, inzwischen sind nahezu alle Elemente deutlich erweitert worden. Erweitert wurde u.a. der *<body>*-Tag. Hier kann man jetzt eine Hintergunddatei mit angeben:

```
<body background=”Grafikdatei”>
```

Auch die Möglichkeiten zur farblichen Gestaltung des Hintergrundes sind hier angesiedelt:

```
<body bgcolor=#RGB text=#RGB link=#RGB vlink=#RGB alink=#RGB>

bgcolor = Bildschirmhintergrund
text = Textfarbe
link = Farbe von Verweisen
vlink = Farbe von Verweisen zu besuchten Zielen
alink = Farbe von Verweisen beim Anklicken
R = Rotwert, hexadezimal 00 bis FF
G = Grünwert, hexadezimal 00 bis FF
B = Blauwert, hexadezimal 00 bis FF
```

Eine mögliche Kombination zeigt die folgende Zeile:

```
<body link="#220099" vlink="#800000">
```

Für Verweise wird eine Farbe benutzt, die aus etwas Rot und viel Blau zusammengesetzt ist. Für Verweise zu besuchten Zielen eine, die nur aus viel Rot besteht. Alle anderen Einstellungen verbleiben auf den Standardwerten.

Man sollte die Farbeinstellungen beim Erstellen von Seiten nicht mehr im BODY-Tag direkt vornehmen, sondern besser per CSS (s.u.).

2.2. Farbangaben in HTML

Bei allen Farbangaben in HTML muss man beachten, dass es sich hier um eine *additive Farbmischung* handelt und nicht um eine *subtraktive Farbmischung* wie beim Tuschkasten. Beim Mischen aller Farben des Tuschkastens sollte sich Schwarz als Mischfarbe ergeben. Bei der additiven Farbmischung, wie sie bei Monitoren und Fernshgeräten üblich ist, ergibt sich Weiß als Mischfarbe.

Monitore und Fernseher besitzen Leuchtpunkte in den Grundfarben Rot, Grün und Blau. Für jeden dieser Leuchtpunkte lässt sich die Intensität zwischen den Werten 0 und 255 (bzw. hexadezimal #00 bis #ff) einstellen. Da die Leuchtpunkte sehr dicht beieinander liegen nimmt das Auge sie nicht ge-

trennt wahr, sondern meldet die entsprechnde Mischfarbe an das menschliche Gehirn. Da zugrundeliegende Farbmodell wird aufgrund der Grundfarben als RGB-Modell bezeichnet.

Ungewohnt für Menschen mit Erfahrungen im Print-Bereich ist auch die Tatsache, dass *Gelb* im RGB-Modell eine Mischfarbe aus den Grundfarben *Rot* und *Grün* ist.

Der RGB-Farbcode für Gelb wäre also hexadezimal #ffff00 und dezimal (255,255,0). Für jeden Bildpunkt stehen drei Byte an Informationen zur Verfügung und damit können 256*256*256 ≈ 16,8 Millionen unterschiedliche Farben beschrieben werden. Das ist viel mehr, als das menschliche Auge überhaupt unterscheiden kann.

2.3. Kommentare

```
<!-- Kommentartext -->
```

Kommentartags dürfen an beliebigen Stellen innerhalb des Textes stehen, sie werden nicht mit ausgewertet. Der Kommentartext darf dabei auch über mehrere Zeilen gehen.

2.4. Texthervorhebungen

Die verschiedenen HTML-Versionen verfügen über eine Vielzahl von Möglichkeiten zur Hervorhebung von Texten. Die folgenden Tags sollten allgemein funktionieren:

```
<b> ... </b> = fett
<i> ... </i> = kursiv
<u> ... </u> = unterstrichen
<s> ... </s> = durchgestrichen
<em> ... </em> = betont
<strong> ... </strong> = stark betont
```

Da es sich hier um logische Auszeichnungen handelt behalten diese auch in den Zeiten von Style-Sheets ihre Bedeutung.

2.5. Verweise und Referenzen

Die wohl wichtigste Funktion innerhalb von WWW-Seiten sind die Verweise oder *Hyperlinks.* Verweise können auf Elemente der gleichen Seite zeigen, oder auf beliebige andere Elemente innerhalb des WWW. Durch die (Hyper-)Links sind die Seiten des WWW untereinander verwoben.

Ein Verweis auf ein beliebiges Element innerhalb des WWW sieht folgendermaßen aus:

```
<a href=”Protokoll://Server/Verzeichnis/Datei”> Verweistext</a>
```

Dabei können u.a. die folgenden Protokolle auftauchen:

```
http://        HTML-Dateien im WWW
ftp://         Dateien auf FTP-Servern
news://        Newsgroup im Usenet
mailto:        persönlichen Email-Adresse
```

Ein Verweis auf mein Wiki würde also folgendermaßen aussehen:

```
<a href="http://www.debacher.de/wiki/">Wiki von Uwe Debacher</a>
```

Der Name der Seite kann weggelassen werden, wenn er „welcome.html” oder „index.html” lautet.

Liegt die Seite im gleichen Verzeichnis, wie die aufrufendes Seite, so kann man den gesamten Pfad weglassen:

```
<a href="filmmuseum1.html">Das Filmmuseum</a>
```

Wann immer möglich sollte man derartige relative Adressen benutzen, dann lassen sich die Webseiten einfach auf andere Webserver kopieren, ohne dass man an den Links etwas ändern müsste.

Vor allem wenn man mit HTML-Werkzeugen arbeitet muss man drauf achten, dass diese nicht absolute Links aus dem lokalen Dateisystem eintragen. Ein Link wie

```
<a href="file://a:\projekt\filmmuseum\filmmuseum1.html">Das Filmmuseum</a>
```

wird nicht funktionieren.

Der Text „Das Filmmuseum” würde auf der Seite blau und unterstrichen dargestellt werden. Wenn man auf diesen Text klickt, dann wird die Seite mit der angegebenen URL (Adresse) geladen.

Gelegentlich benötigt man Verweise innerhalb einer Seite. Das könnte z.B. innerhalb einer Literaturliste geschehen, wenn man direkt auf andere Literaturangaben verweisen möchte, oder in einem Gesetzestext, wenn man von einem Paragraphen auf einen anderen Paragraphen verlinken möchte.

Hierzu muss man an der jeweiligen Zielstelle einen „Anker” setzen:

```
<a name=”Seitenanfang”>Inhaltsverzeichnis</a>
```

definiert einen Anker, auf den dann innerhalb des Dokumentes mittels:

```
<a href=”#Seitenanfang”>Zum Anfang der Seite</a>
```

verwiesen werden kann. Von einer anderen Seite aus kann man auch direkt auf diesen Anker verweisen, mittels

```
<a href=”seite.html#Seitenanfang”>Zum Anfang der anderen Seite</a>
```

Will man einen Verweis auf einen Anker innerhalb der gleichen Seite richten, so muss im Text der entsprechende Anker gesetzt werden:

```
<a name="Bezeichner">Text</a>
```

Anschließend kann man auf diesen Anker verweisen:

```
<a href="#Bezeichner">Verweistext</a>
```

Man kann sogar auf einen lokalen Anker innerhalb einer anderen Seite verweisen:

```
<a href="filmmuseuml.html#Anker">Hier zum Anker</a>
```

oder

```
<a href="http://www.debacher.de/wiki/HTML-Grundlagen#2.5._Verweise_und_Refe-
renzen">Abschnitt 2.5 von HTML-Grundlagen</a>
```

2.6. Grafiken

Eine moderne Gestaltung bekommt eine HTML-Seite durch eingebundene Grafiken:

```
<img src=”dateiname” align="left" alt=”Beschreibung” width=”Zahl” heigth=”Zahl”>
```

Mit *width* bzw. *height* kann die Größe des Bildes explizit angegeben werden. *alt* liefert einen Ersatztext, falls die Abbildung nicht, oder noch nicht geladen wurde. Wichtig ist diese Angabe auch für behinderte Menschen, die sich die Seite von speziellen Programmen vorlesen lassen.

Mit *align* wird die Ausrichtung des Bildes zum Text festgelegt. Hier gibt es die Möglichkeiten *top*, *middle* und *bottom*.

Der Dateiname hinter *src* gibt wieder an, wo das Bild zu suchen ist. Hier kann es sich um eine Adresse auf einem vollkommen anderen Rechner handeln, oder im einfachsten Fall um eine Datei im gleichen Verzeichnis. Dann verkürzt sich die Adresse wieder:

```
<img src="logo.gif" alt="Das Logo" width="366" height="151">
```

Die Größenangaben dienen nur zur Information des Browsers, sie sollten auf keinen Fall zur Skalierung benutzt werden. Die Skalierung nimmt man immer vorher mit einem Grafikprogramm vor und bindet die Grafik dann in der passenden Größe ein. Damit vermeidet man unnötigen Datenverkehr und die damit verbundenen Wartezeiten.

Fehlen die Größenangaben, so kommt es zu einem unruhigen Aufbau der Seite. Der Browser lädt zuerst den Quelltext der Seite und reserviert dabei den Platz für die Abbildung aus Standardwerten. Wenn die Seite aufgebaut ist, dann lädt er nacheiander die Grafiken. Jedesmal wenn der Browser feststellt, dass die geladene Grafik nicht in den reservierten Platz passt muss er die Seite insgesamt neu aufbauen.

3. Grundlagen zum Thema Grafik

Ein wichtiges Gestaltungsmerkmal für Web-Seiten sind die Grafiken. heute kommt keine Web-Seite mehr ohne nette Bilder oder zumindest ein Logo aus. Hier gibt es aber ein paar Punkte zu beachten.

3.1. Bildschirmauflösung und Farbtiefe

Bei der Darstellung auf dem Bildschirm gibt es nur ein paar Dinge zu beachten. Wichtig ist die Bildschirmauflösung, es macht nämlich selten Sinn, wenn eine Grafik größer ist, als der Anzeigebereich. Außerdem ist die Dateigröße einer Grafik natürlich auch von der Bildgröße abhängig. Üblich sind heute die Bildschirm-Auflösungen von:

```
640x480 Punkten,
800x600 Punkten,
1024x786 Punkten,
1280x800 Punkten oder gar
1280x1024 Punkten.
```

Jeder dieser Punkte hat nun eine Farbe. Die Zahl der gleichzeitig zur Verfügung stehenden Farben bezeichnet man als Farbtiefe. Je höher die Farbtiefe, desto größer wieder der Speicherverbrauch des Bildes.

Im Web teilweise noch üblich sind Farbtiefen von 256, dann verbraucht jeder Punkt genau 1 Byte. Moderne Computer können aber auch 16,7 Millionen Farben darstellen. Dann werden für jeden Punkt 3 Byte verbraucht, ein Byte für jede der drei Grundfarben.

Ein Bildschirminhalt von 1024x768 Punkten und voller Farbtiefe belegt dann 1024x768x3 Bytes = 2.359.296 Bytes ≈ 2,3 Mbyte.

3.2. Grafikformate

Es gibt sehr viele Möglichkeiten, wie man eine Bildschirm-Grafik aus Diskette oder Festplatte speichern kann. Die Unterschiede liegen z.B. darin, mit welchem Punkt man beim Speichern anfängt. Viele Formate beginnen Oben Links, manche in einer anderen Ecke. Unterscheiden kann man die Formate an den unterschiedlichen Erweiterungen (Extensionen) unter denen sie abgespeichert werden. Ein sehr verbreitetes Format ist das BMP-Format, das zu Windows gehört.
Im Internet sind hauptsächlich drei Grafikformate üblich, das GIF, das PNG und das JPG Format. Andere Formate können vom den meisten Browser nicht dargestellt werden. Das PNG-Format wurde als Nachfolger für das GIF-Format entwickelt, um Urheberechtsprobleme zu vermeiden. Neu hinzu kommt momentan das SVG-Format für frei skalierbare Zeichnungen.

Diese genannte Formate zeichnen sich dadurch aus, dass sie mit dem Speicherplatz sehr schonend umgehen, da die Daten komprimiert werden.

Ein Foto wie das dargestellte belegt

- als BMP-Datei 2.300 kByte,
- als GIF-Datei etwa 465 kByte und
- als JPG-Datei nur 80 kByte.

Zwischen den drei Darstellungen ist ein Unterschied kaum wahrnehmbar. Die einzige Einschränkung besteht darin, dass das GIF-Format nur 256 Farben erlaubt. Photos sollten generell im JPG-Format gespeichert werden, da hier beliebig viele (bzw. 16,7 Millionen) Farben zur Verfügung stehen. Das PNG bzw. das GIF-Format finden immer dann Anwendung, wenn es sich um selbst erstellte Zeichnungen handelt, bei denen große gleichfarbige Flächen auftauchen. Zwei weitere Vorteile hat das GIF-Format noch, es erlaubt transparente und animierte Grafiken.

3.3. Transparente Grafiken

Transparenz bei Grafiken kennen das PNG und das GIF-Format. Dabei wird eine beliebige der im Bild benutzten Farben als durchsichtig erklärt. Im Browser scheint dann an den entsprechenden Stellen der Seiten-Hintergrund durch.

Damit lässt sich der Eindruck erwecken, dass die Abbildung rund geschnitten wäre.

3.4. Animierte Grafiken

Mit dem GIF-Format kann auch eine Art Daumenkino realisiert werden. Im Prinzip handelt es sich also jeweils um eine Datei, in der mehrere Bilder vorhanden sind, die nacheinander abgespielt werden. Es entsteht hierdurch der Eindruck von Bewegung.

Die Animation besteht aus einer Folge von Einzelbildern.

(Grundlage für die Animation ist ein Streifen zum Lebensrad (Phenakistiskop), Quelle: http://www.filmmuseum-hamburg.de)

4. Anordnen von Text und Grafik

Beim Mischen von Text und Grafik fällt auf, dass man diese Elemente nicht so anordnen kann, wie man es aus der Textverarbeitung gewohnt ist.

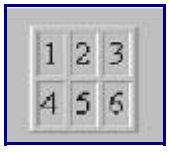
Eine Grafik innerhalb einer Textzeile wird behandelt wie ein großer Buchstabe. Man kann also nur festlegen, ob der Text oben, unten oder mittig zur Grafik erscheint. Eine Fortsetzungszeile beginnt dann auch immer am linken Rand der Seite. Will man Text und Grafik genauer platzieren, so muss man spezielle Elemente einsetzen.

4.1. Tabellen

Mit den folgenden Tags wird eine einfache Tabelle erzeugt:

```
<table border="1">
<tr>
   <td>1</td>

   <td>2</td>
   <td>3</td>
 </tr>
 <tr>
   <td>4</td>
   <td>5</td>
   <td>6</td>
 </tr>
</table>
```



Eine Tabelle mit einer Zeile und zwei Spalten kann man dann sehr gut benutzen, um in der ersten Spalte das Bild unterzubringen und in der zweiten Spalte den Text.

Wenn man dann noch durch die Angabe

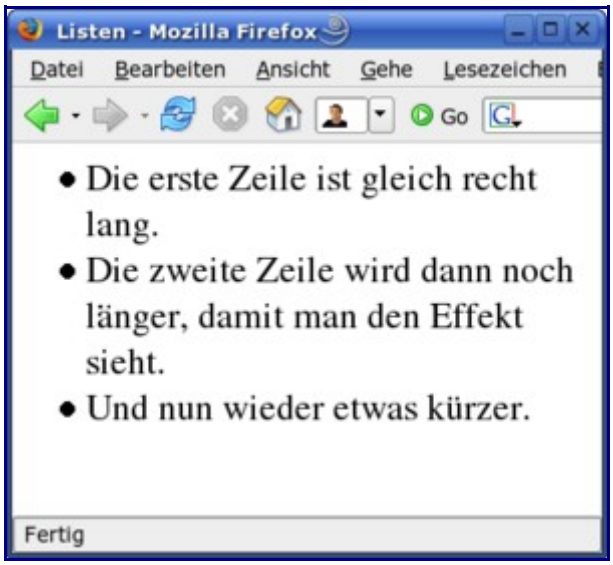
```
<table border="0">
```

die Tabelle ohne sichtbaren Rahmen erzeugt, dann ist eine Anordnung möglich, wie man sie von den Textverarbeitungen her kennt.

Die Möglichkeiten von Tabellen sind sehr vielseitig. So kann man z.B. die Übersichtlichkeit von längeren Tabellen dadurch erhöhen, dass man z.B. jede zweite Zeile grau hinterlegt. Dadurch bekommt man einen Eindruck, wie man ihn von älteren Computerausdrucken her kennt. Die Hintergrund einer ganzen Tabelle, einer Zeile oder nur einer Zelle lässt sich mittels bgcolor="#RGB" einstellen. Der Hintergrund einer einzelnen Zelle lässt sich also über

```
<td bgcolor="#ff0000"> auf Rot stellen.
```

4.2. Listen



Zu den Möglichkeiten der Textanordnung gehören auch die Listen. Bei einer Liste möchte man erreichen, dass jeder Eintrag mit einem Gedankenstrich, einem Punkt oder einer Nummer beginnt. Sollte für den Text eine Fortsetzungszeile notwendig werden, so sollte diese eingerückt sein.

Aufgebaut wird eine derartige Liste mit folgenden Tags:

```
Nun Listen:
<ul>
 <li>
  Die erste Zeile ist gleich recht lang.</li>
<li>
  Die zweite Zeile wird dann noch länger, damit man den Effekt sieht.</li>
<li>
  Und nun wieder etwas kürzer.</li>
</ul>
```

Mit <ul> baut man eine „bulleted list” auf, eine Liste mit Punkten. Entsprechend mit <ol> eine nummerierte Liste.

5. Cascading Style Sheets

In vielen Anwendungsprogrammen kann man das Aussehen über Vorlagen vereinheitlichen. In den aktuellen Browser-Versionen kann man bei HTML für diesen Zweck die Style Sheets nutzen. Da man diese Vorlagen kaskadieren kann werden sie meist als *Cascading Style Sheets* (CSS) bezeichnet.

Mit diesen CSS kann man fast jedem Tag neue Eigenschaften zuordnen bzw. auch so etwas wie eigene Tags kreieren. Dazu ein einfaches Beispiel (einige Zeilen sind entfernt um die Übersichtlichkeit zur erhöhen):

```
<html><head>

 <style type="text/css">
 body   { background-color: red; }
 h1     { color: blue; }
 </style>

 </head>
 <body>
 <h1>Hier die &Uuml;berschrift</h1>
 Hier der Body-Text
 </body></html>
```

Neu ist hier der <style> Tag, der im Head untergebracht und auch beendet wird. Innerhalb des Style-Tags kann ich andere Tags erwähnen, hier body und h1, die dann im Body-Bereich der Seite neue Eigenschaften bekommen, die Eigenschaften werden in geschweiften Klammern angegeben. Im vorliegenden Fall wird eingestellt, dass im body-Bereich die Hintergrundfarbe Rot sein soll und die Schriftfarbe bei h1 Blau.

Diese Einstellungen wirken auf alle entsprechenden Tags der vorliegenden Seite! Richtig hilfreich werden diese Style-Sheets, wenn man sie in einer extra Datei sammelt und dann in alle eigenen Seiten einbindet:

```
<html><head>
 <link rel="stylesheet" href="style.css"  type="text/css">
 </head>

 <body>
 <h1>Hier die &Uuml;berschrift</h1>
 Hier der Body-Text
 </body></html>
```

Zum Einbinden einer externen Style-Sheet-Datei dient der Link-Tag, der auch im Head-Bereich steht, aber nicht beendet werden muß.

In der Style-Sheet-Datei stehen dann die reinen Definitionen, im vorliegenden Fall also:

```
body    { background-color: red; }
h1      { color: blue; }
```

Der Vorteil dieser Konstruktion besteht darin, dass ich das Layout mehrerer Seiten ganz einfach dadurch verändern kann, dass ich die Datei mit den Style-Sheets ändere.

Will man bei einem Tag mehrere Eigenschaften gleichzeitig ändern, so trennt man diese durch ein Semikolon:

```
body    { background-color: red; }
h1      { color: blue; font-style: italic;}
```

Die folgenden Eigenschaften können innerhalb von Style-Sheets benutzt werden (die Liste ist auf keinen Fall vollständig:

Einfache Textauszeichnungen mit einzelnen Strichen (über dem #)	
Eigenschaft	**Beschreibung**
font-family	Legt die Schriftart fest, hier können z.B. die Windows-Schriften wie Arial angegeben werden, oder die Bezeichner serif, sans-serif, cursive, fantasy oder monospace. Diese Bezeichner sind auf jedem System mit einer passenden Schrift verknüpft.
font-size	Legt die Schriftgröße fest. Hier können xx-small, x-small, small, medium, large, x-large und xx-large benutzt werden. Möglich sind aber, wie bei den Randeinstellungen, numerische Angaben in den Einheiten: • in: Inch bzw. Zoll, etwa 2,54cm • mm, cm: unsere gewohnten Einheiten • pt: Punkt, entspricht 1/72 inches • pc: Pica, entspricht 12 Punkt • px: Pixel, stark auf das Ausgabegerät bezogen • em: Relativ bezogen auf die Schriftgröße des Elementes • ex: relativ bezogen auf die Größe des x in der Schrift • %: relativ zur Größe des Elementes
font-style	Bei den Schriftstilen sind oblique, italic oder normal definiert.
font-weight	Gibt an, wie fett die Schrift sein soll. Hier sind Angaben wie normal, bold oder Zahlenwerte wie 100, 200, ...,900 möglich.
color	Legt die Farbe bzw. Schriftfarbe fest. Möglich sind Bezeichner wie: aqua, black, blue, fuchsia, grey, green, lime, maroon, olive, purple, red, silver, teal, white, yellow oder RGB-Tripel wie #ff00ff, rgb(255, 0, 255) oder rgb(100%, 0, 100%).
text-decoration	Legt die Ausgestaltung der Schrift fest, mögliche Werte sind underline, blink oder none.
background-color	Legt die Hintergrundfarbe eines Elementes fest.
text-align	Legt die Ausrichtung eines Textbereiches fest. Mögliche Werte sind left, center, right oder justify.
text-indent	Legt einen Texteinzug der ersten Textzeile eines Elementes fest. Angegeben werden kann eine absolute Zahl (s.o.), oder eine Prozentangabe bezogen auf die Gesamtbreite sein.
vertical-align	Legt die vertikale Textausrichtung fest. Mögliche Werte sind baseline, middle, sub, super, text-top und text-bottom
line-height	Zeilenhöhe. Die angegeben Zahl ist der Multiplikator für die Zeilenhöhe. Bei 2 wird die Zeilenhöhe verdoppelt.
margin-left	Legt den Abstand zum linken Seitenrand für ein Element fest.
margin-top	Legt den oberen Rand eines Elementes fest.
margin-right	Legt den rechten Rand fest.

margin-bottom	Legt den unteren Rand fest.
margin	Legt alle Ränder eines Elementes fest.
padding	Padding legt für ein Element den Abstand nach Innen fest, während margin den Abstand nach Außen angibt.

5.1. Klassen

Im vorangegangenen Abschnitt hatten wir u.a. den h1-Tag umdefiniert. Damit steht der „ursprüngliche“ Tag nicht mehr zur Verfügung. Will man beide Möglichkeiten nutzen können, so gibt es mehrere Möglichkeiten.

```
h1.meineDokumente {color: blue; }
```

Hiermit definiert man eine eigene „Klasse“ von H1 Tags. Normalerweise bleiben die Tags unverändert, es sein den, man gibt die Klasse mit an:

```
<h1 class=“meineDokumente“>Dieser Text wird blau</h1>
```

In diesem Beispiel haben wir die Klasse meineDokumente an den Tag h1 gebunden. Das ist aber nicht unbedingt sinnvoll und notwendig. Wir hätten eine Klasse auch unabhängig definieren können (man beachte den Punkt vor dem Namen!):

```
.rot {color: red;}
```

Nun kann ich diese Klasse in verschiedenen Tags benutzen:

```
<h1 class=“rot“>Dieser Text wird rot</h1>
 <p class=“rot“>Dieser Absatz auch.
```

5.2. ID

Ähnlich wie die Definition einer Klasse funktioniert die Festlegung einer ID:

```
#rot {color: red; }
```

Definiert eine ID (man beachte das einleitende #), die dann in Tags benutzt werden kann:

```
<h1 id=“rot“>Dieser Text wird rot</h1>
 <p id=“rot“>Dieser Absatz auch
```

5.3. SPAN und DIV

Will man nur ein einzelnes Wort, oder einen Textbereich verändern, so geht das bisher nicht ganz einfach, da wir ja keinen Standard-Tag einsetzen können. Dafür gibt es die eigenschaftslosen Pseudo-Tags *span* und *div*.

```
In diesem Satz wird ein Wort <span id=“rot“>rot</span> geschrieben.
```

Die Wörter bleiben alle in der gleichen Zeile, lediglich die Textfarbe wird verändert. Beim div-Tag wird zusätzlich noch vor und nach dem Tag ein Zeilenumbruch eingefügt:

```
In diesem Satz wird ein Wort <div id=“rot“>rot</div> geschrieben.
```

Nun erfolgt die Ausgabe in drei Zeilen.

5.4. Positionieren von Elementen

Eine vollkommen neue Möglichkeit von CSS besteht darin, dass man Elemente auch absolut positionieren kann. Dazu kann man für ein Element über CSS die Position der linken oberen Ecke angeben. Wenn man dann noch weiß, dass das Bildschirm-Koordinatensystem links oben mit dem Punkt (0/0) beginnt, dann kann man Texte und Bilder sehr genau positionieren:

```
<html><head>

 <style type="text/css">
 #element1 {    position: absolute;  top: 50px;  left: 100px; }
 </style>

 </head>
 <body>
 <div id="element1">Toll nicht wahr?</div>
 </body>
 </html>
```

Damit wird der Text genau an dieser Position angezeigt.

Die Möglichkeiten der Positionierung gehen so weit, dass man sogar überlappenden Text erzeugen kann:

```
<html><head>

 <style type="text/css">
 #element1 {    position: absolute;  top: 50px;  left: 100px; }
 #element2 {    position: absolute;  top: 55px;  left: 120px; }
 </style>

 </head>
 <body>
 <div id="element1">Toll nicht wahr?</div>
 <div id="element2">Man kann auch ueberlappen!</div>
 </body>
 </html>
```

Neben Text kann man natürlich auch Bilder entsprechend positionieren:

```
<html><head>

 <style type="text/css">
 #element1 {    position: absolute;  top: 50px;  left: 100px; }
 #element2 {    position: absolute;  top: 55px;  left: 120px; }
 </style>

 </head>
 <body>
 <div id="element1">Toll nicht wahr?</div>
 <div id="element2"><img src="obj.gif" width="278" height="215"></div>
 </body>
 </html>
```

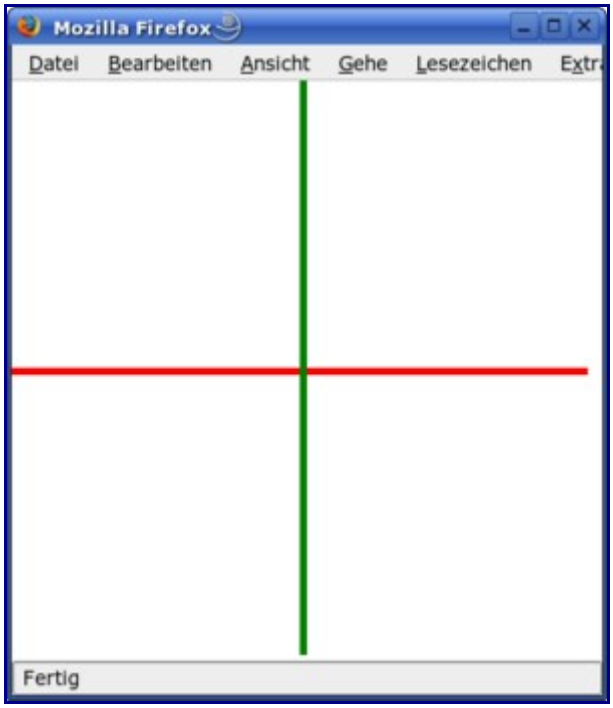
Dieses Beispiel wird nicht so ganz den Erwartungen entsprechen, da zuerst der Text dargestellt wird, danach die Abbildung. Wenn diese nicht transparent ist, dann ist vom Text nur noch wenig zu sehen. Man müsste also die Reihenfolge ändern, in der die Elemente dargestellt werden, oder angeben, welches Element vorne und welches hinten liegt.

Genau dafür gibt es noch den z-index, der die dritte Dimension beschreibt. Je höher dieser Wert ist, desto weiter vorne liegt das Element:

```
<html><head>

 <style type="text/css">
 #element1 {    position: absolute; top: 50px;  left: 100px; z-index: 2;}
 #element2 {    position: absolute; top: 55px;  left: 120px; z-index: 1;}
 </style>

 </head>
 <body>
 <div id="element1">Toll nicht wahr?</div>
 <div id="element2"><img src="obj.gif" width="278" height="215"></div>
 </body></html>
```



Was übrigens auch funktioniert, ist das nachträgliche Verschieben eines positionierten Elementes durch ein JavaScript (das ist dann DHTML). Leider gibt es hierbei feine Unterschiede zwischen den Browsern.

Im folgenden Listing werden zwei Abbildungen mit jeweils einer farbigen Linie geladen.

```
<html><head>
 <style type="text/css">
 #element1 { position: absolute; top: 50px; left: 10px; z-index: 1; }
 #element2 { position: absolute; top: 55px; left: 200px; z-index: 1; }
 </style></head>
 <body>
 <div id="element1"><img src="wlinie.gif" name="bild1" border="0"></div>
 <div id="element2"><img src="slinie.gif" name="bild2" border="0"></div>
 </body></html>
```

Aufgabe ist es die Linien so anzuordnen, dass sich ein Pluszeichen ergibt und die rote Linie über der grünen liegt (am Schnittpunkt).

6. Frames

Will man sich ernsthaft mit Frames auseinander setzen, so benötigt man sicherlich mehrere Tage dazu. Da aber viele der zu bearbeiteten Seiten mit Frames aufgebaut sind muss man ein paar Grundkenntnisse besitzen.

Grundsätzlich arbeitet man, wenn man Frames einsetzt immer mit mehreren Seiten, man spricht daher auch von einem *Frameset*. Die Hauptseite des Framesets besitzt normalerweise keinen eigentlichen Inhalt, sondern bindet die beteiligten Seiten ein.

Der Quelltext für die Hauptseite ist dadurch immer so kurz wie das folgende Beispiel.

```
<html>
 <head>
   <title>Unser Frameset</title>
 </head>
 <frameset cols="30%,*">
   <frame src="navigation.html">
   <frame src="inhalt.html">
 </frameset>
 <noframes>
 <body>Schade, Ihr Browser kennt keine Frames!<p>
 </body>
 </noframes>
 </html>
```

Mit dem Tag *frameset* wird die eigentliche Definition eingeleitet. Wir legen fest, dass zwei Spalten auftauchen sollen, die erste Spalte bekommt 30% des zur Verfügung stehenden Platzes, die zweite den Rest. Will man die Seite horizontal teilen, so benutzt man statt des Schlüsselwortes *cols* das Schlüsselwort *rows*. Danach kommen dann mit dem frame-Tage die Definitionen der beiden Spalten. Die eigentlichen Rahmen sind dann wieder ganz normale (vollständige) HTML-Seiten:

```
<html><head></head>
 <body>
 <p>
 Linker Rahmen (Frame)
 </body>
 </html>
```

Gelegentlich findet sich innerhalb des FRAME-Tags noch eine Namensangabe wie

```
<frame src="inhalt.html" name="FensterRechts">
```

Durch diese Namensangabe kann man später bei Links festlegen, in welchem Frame die neue Seite erscheint. Dazu würde man dann einen Link einbauen wie:

```
<a href="news.htm" target="FensterRechts">Neuigkeiten</a>
```

7. Image-Maps

Im Internet findet man häufiger Abbildungen, bei denen man einzelne Teile anklicken kann und so auf eine zugehörige Seite kommt.

Klickt man hier z.B. auf die Brille von Lars, so landet man auf der Seite www.brille.de.

Das zugehörige Listing sieht folgendermaßen aus.

```
<html><head><title>Klick auf Lars</title></head>
 <body><h1>Klicke auf Lars</h1>

 <map name="Lars">
  <area shape="circle" coords="227,194, 20" href="http://www.brille.de/"
alt="Brille">
  <area shape="circle" coords="258,209, 20" href="http://www.brille.de/"
alt="Brille">
  <area shape="circle" coords="250,200, 45" href="http://www.kopf.de/"
alt="Kopf">

  <area shape="rect" coords="96,24,281,139" href="http://www.rumpf.de/"
alt="Rumpf">
  <area shape="poly" coords="91,76, 18,252, 87,288, 62,246, 135,119"
href="http://www.beine.de/" alt="Beine">
  <area shape="rect" coords="0,0,451,295" href="http://www.nix.de/" alt="Nix">

 </map><p>
 <img src="Lars_auf_kopf.jpg" width="451" height="295" border="0" alt="Karte"
usemap="#Lars"></p>
 </body>
 </html>
```

Die verweis-sensitiven Flächen können Kreise, Rechtecke oder Polygone sein. Ein Kreis wird durch den Mittelpunkt und seinen Radius gekennzeichnet. Das Rechteck wird durch den Punkt links oben und den Punkt rechts unten festgelegt. Zur Definition eines Polygons dient eine Liste von Punkten.

Bei überlappenden Elementen spielt die Reihenfolge eine Rolle. Je weiter vorn ein Element in der Liste steht, desto eher wird es berücksichtigt.

8. Ein kleiner Ausflug in Javascript

Javascript ist eine 1995 von Netscape eingeführte Programmiersprache, die in HTML-Seiten eingebunden werden kann und vom lokalen Browser ausgewertet wird. In dieser Abhängigkeit vom Browser liegt ein großes Problem. Da sich manche Softwarehäuser nicht immer an die vorgegebenen Standards halten, muss man innerhalb von Javascript sehr oft unterschiedlichen Code für unterschiedliche Browser schreiben. Recht gut an die Webstandards hält sich der Browser Opera, der zudem auch noch sehr schnell ist.

Javascript läßt sich an vielen Stellen in den Code einer HTML-Seite integrieren. Einen sehr interessanten Effekt erhält man, wenn man Links um die Eigenschaften onmouseover und onmouseout erweitert:

```
<html><head></head>
 <body><h1>Hover Effekt</h1>
 Mit der Maus einfach über die rote Linie gehen, möglichst am linken Rand:<br>
 <a href="http://www.linie.de"
   onmouseover="document.bild1.src='slinie.gif'; "
   onmouseout="document.bild1.src='wlinie.gif';  ">
 <img src="wlinie.gif" name="bild1" border="0"></a>
 <br>Hier geht es weiter.
 </body></html>
```

Geht man jetzt mit der Maus über den Link, so verändert sich die Farbe der Linie, da die andere Abbildung geladen wird. Die Abbildungen sind sehr unterschiedlich (waagrecht bzw. senkrecht), daher ist der Effekt auch je nach Browser unterschiedlich. Im Netscape (4.07) wechselt nur die Farbe, die Linie bleibt waagrecht. Im Firefox verändert sich das gesamte Seitenlayout, da wirklich die senkrechte Linie dargestellt wird.

Wichtig ist, dass die Abbildung einen Namen bekommt, hier bild1. Über diesen Namen kann das kleine Javascript-Programm, das beim Überstreichen des Links mit der Maus aufgerufen wird, die Abbildungsquelle (src) ändern.

Beim Laden der zweiten Abbildung kann es eine kleine Verzögerung geben, daher lädt man das zweite Bild meist schon vorher:

```
<html><head>
 <script type="text/javascript">
   var b = new Array();
   b[0] = new Image(); b[0].src = "wlinie.gif";
   b[1] = new Image(); b[1].src = "slinie.gif";
 </script>
 </head>

 <body><h1>Hover Effekt2</h1>
 Mit der Maus einfach über die rote Linie gehen, möglichst am linken Rand:<br>
 <a href="http://www.linie.de"
     onmouseover="document.bild1.src=b[1].src;  "
     onmouseout= "document.bild1.src=b[0].src;  ">
 <img src="wlinie.gif" name="bild1" border="0"></a>
 <br>Hier geht es weiter.
 </body></html>
```

Hier werden die beiden Bilder schon im Header geladen und stehen dann innerhalb der Seite sofort zur Verfügung.

9. Formulare in HTML

Alle Eingaben im Web erfolgen in ein HTML-Formular. Ein Formular besteht mindestens aus den folgenden Tags: (Kursiv gesetzte Wörter sind Platzhalter für eigene Bezeichner)

```
<form action=”auswert.php” method=”get”>
 <input type=”text” name=”name” size=”Size” maxlength=”Länge”>
 <input type=”submit” value=”Absenden”>
 <input type=”reset”  value=”Verwerfen”>
 </form>
```

Mit dem Einleitungs-Tag muss ein Programm angegeben werden, das die Eingabedaten auswertet, in diesem Fall ein Programm namens auswert.php. Zusätzlich muss angegeben werden, wie dieses Programm die Daten erhält. Dazu gibt es die Möglichkeiten get und post. Bei der Methode get werden die Daten einfach an die URL des Auswertungs-Programmes angehängt:

```
auswert.php?name=Meier&vorname=Klaus
```

Bei der Methode *post* sieht man diese Daten nicht, da eine Art Dialog erfolgt. Bei der Programmentwicklung ist also die Methode *get* praktischer, im endgültigen Programm ist post vorteilhafter.

Die zweite Zeile definiert ein Eingabefeld mit dem Namen name und der Größe Size. Diese Angabe betrifft nur die Darstellung auf dem Bildschirm. Die Maximalzahl der Zeichen die eingegeben werden kann wird auf den Wert Länge festgelegt. Normalerweise hat ein Formular natürlich mehr als ein Eingabefeld. Die einzelnen Felder werden dabei durch ihre Namen unterschieden. Wichtig für jedes Formular ist auch ein Knopf zum Abschicken. Dazu dient der Typ *submit*. Die Daten im Formular werden ausgelesen und an das Auswertprogramm übergeben. Üblich in Formularen ist auch ein Resetknopf. Klickt man auf diesen Knopf, so werden alle Eingabefelder gelöscht.

Als action kann anstelle eines eigentlichen Auswertprogrammes auch eine Mail verschickt werden:

```
<form action=”mailto:debacher@gyloh.hh.schule.de” method=”post”>
```

Es stehen innerhalb eines Formulars nicht nur einfache Eingabezeilen zu Verfügung.

9.1 Text

Der Standardtyp für Eingabe felder ist *text.* Derartige Felder dienen zur Eingabe von mximal 256 Zeichen.

Wie die meisten Eingabefelder erlaubt auch das Text-Feld die Angabe eines Wertes, der vorab in das Feld eingetragen wird.

```
<input type=”text” name=”name” value=”Klausi”>
```

Diese Möglichkeit ist besonders dann wichtig, wenn man mittels Formular vorhandene Datensätze ändern möchte.

9.2. Hidden

Eine Angabe der folgenden Art:

```
<input type=”hidden” name=”name” value=”Klausi”>
```

bewirkt keinerlei Darstellung auf dem Bildschirm. Dieses Feld wird mit seinem Wert einfach nur an das Auswertprogramm übergeben, ohne dass der Benutzer Eingabe machen kann oder überhaupt etwas von diesem Feld bemerkt. Dieser Tag macht nur Sinn, wenn ein *value* mit angegeben wird.

9.3. Password

Auch dies ist nur eine Variation des Types Text.

```
<input type=”password” name=”name”>
```

Hierbei wird der Eingabetext nicht bzw. in Form von Sternchen auf dem Bildschirm dargestellt.

9.4. Textarea

Mit diesem Tag wird ein Eingabefeld definiert, dessen Höhe und Breite festgelegt werden müssen

```
<textarea name="name" rows="Höhe" cols="Breite">
 Vorgabetext
 </textarea>
```

Dem Eingabefeld kann man leider keine Maximalzahl von Zeichen übergeben. Höhe und Breite beziehen sich auf die Bildschirmdarstellung. Innerhalb des Bereiches kann gescrollt werden. Vorgaben werden hier nicht mit dem Value-Tag eingetragen, sondern zwischen Anfangs- und Endtag gesetzt.

9.5. Auswahlfelder

Will man keine freie Eingabe zulassen, sondern dem Benutzer nur die Auswahl zwischen vorgegebenen Werten ermöglichen, dann bietet sich die folgende Kombination an:

```
<select name=”name” size=Zeilen>
  <option value=”wert”>Beschreibungstext
  <option selected value=”wert”>Beschreibungstext
  ...
 </select>
```

Hiermit stellt man dem Benutzer ein Feld zur Verfügung, das mit einem Mausklick geöffnet wird und die angegebenen Einträge zur Auswahl stellt. Nachdem ein Eintrag angeklickt wurde schließt sich das Auswahlfenster wieder. Ein Wert darf durch Zusatz von *selected* als voreingestellt gekennzeichnet werden. Dieser Wert erscheint dann auch im geschlossenen Eingabefeld.

Der Select-Tag erlaubt zusätzlich das Schlüsselwort *multiple,* welches die Auswahl mehrerer Elemente gleichzeitig erlaubt. Für die Nutzung mittels PHP muss dann der Feldname auch ein Array kennzeichnen, also die eckigen Klammern beinhalten:

```
<select name=”name[]” multiple size=Zeilen>
  <option value=”wert”>Beschreibungstext
  ...
 </select>
```

9.6. Checkboxen

Die Verwendung von Auswahlfeldern ist sehr platzsparend. Manchmal möchte man aber alle Optionen immer auf dem Bildschirm sehen, dann arbeitet man besser mit Checkboxen

```
<input type=”checkbox” name=”name1” value=”wert1”>Text1<br>
 <input type=”checkbox” name=”name2” value=”wert2”>Text2<br>
```

Für jeden der Einträge taucht auf dem Bildschirm ein ankreuzbares Kästchen auf. Diese Kästchen sind voneinander unabhängig. Es können also z.B. alle Kästchen oder kein Kästchen angekreuzt sein. Lässt man den *value*-Eintrag weg, so wird im Zweifelsfall der Wert *on* genommen.

9.7. Radiobuttons

Will man erreichen, dass immer nur eine Möglichkeit aus einer Auswahl markiert werden kann, dann arbeitet man besser mit Radiobuttons:

```
<input type=”radio” name=”name” value=”eintrag1”>Text1<br>
 <input type=”radio” name=”name” value=”eintrag2”>Text2<br>
```

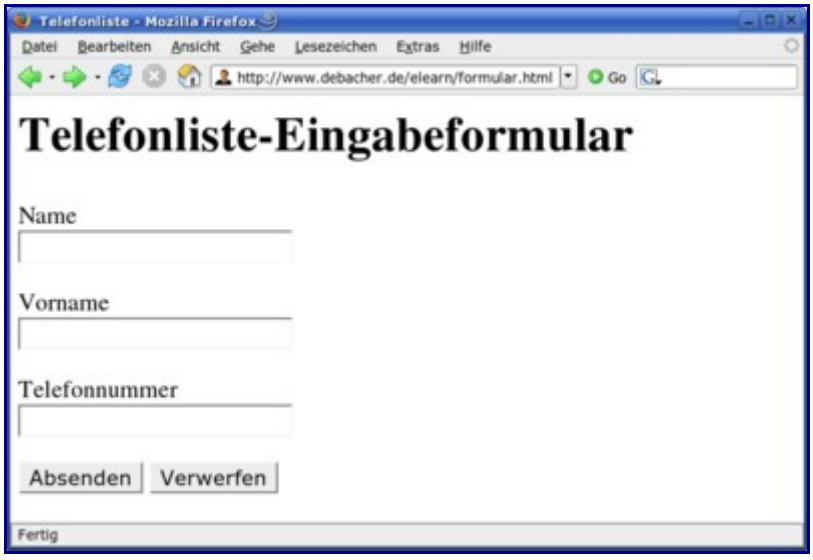
Hier haben alle Felder den gleichen Namen aber unterschiedliche Werte.

9.8. Ein kleines Beispiel

Das folgende Beispiel kann als Einstieg in ein eigenes Gästebuch dienen. Konkret handelt es sich hier um das Eingabeformular:

```
<!-- Formular für eine Telefonliste von Uwe Debacher 2006-->
<html>
 <head><title>Telefonliste</title></head>
 <body><h1>Telefonliste-Eingabeformular</h1>
   <form ACTION="http://www.debacher.de/elearn/auswert.php" method="get">
    <!-- Ein normales Eingabefeld -->
    Name<br>
    <input type="text" name="name" size="20" maxlength="50"><p>
    Vorname<br>
    <input type="text" name="vorname" size="20" maxlength="50"><p>
    Telefonnummer<br>
    <input type="text" name="telefon" size="20" maxlength="50"><p>
    <input type="submit" value="Absenden">
    <input type="reset"  value="Verwerfen">
   </form>
  </body>
 </html>
```

Lädt man die Seite in einen Browser, so ergibt sich folgendes Bild:



Von "http://www.debacher.de/wiki/HTML-Grundlagen"