Elemente von HTML

HTML steht für **H**yper**T**ext **M**arkup **L**anguage, d.h. ein Textdokument, das nur druckbare Zeichen enthält und Anweisungen, sogenannte *Tags*, welche z.B. die Bedeutung oder die Darstellung des Textes bzw. des Bereiches ausdrücken. HTML-Texte werden dabei erst durch einen sogenannten *Browser* interpretiert und nach dessen Fähigkeiten dargestellt. Insbesondere können solche Darstellungen sich von Browser zu Browser unterscheiden.

Es gibt folgende Arten von Elementen:

1. Elemente die durch *Anfangs-* und *End-Tags* markiert sind und dazwischen Text enthalten, wie z.B. <NAME>.... ein Text .... </NAME>. Dabei ist NAME ein definierter Tag, umschlossen von <..> - Zeichen und </NAME> die zugehörige Endemarkierung.
2. „Leere“ Elemente kennzeichnen keinen Text, sondern haben sofortige Wirkung und besitzen daher keine Endemarkierung.
3. Elemente können zusätzlich *Attribute* enthalten, welche der Anfangsmarkierung beigestellt sind. <NAME ATTRIBUTE=“zeichenfolge“>.

Hinweise:

1. Tags unterscheiden keine Groß-Kleinschreibung. Allerdings gilt das für Zeichenfolgen in Anführungszeichen!
2. Elemente dürfen geschachtelt werden, sich aber nicht überlappen!

Einfaches Beispiel:

```
<HTML>
<HEAD>
<TITLE> Das ist der Titel des Dokuments </TITLE>
</HEAD>
<BODY>
<H1> Erste Überschrift </H1>

<P>Hallo! Dies ist ein nicht gerade wichtiges Dokument.
Es
    enthält vor allem keine herausragende <EM>dichterische</EM> Leistung,

sondern soll die <STRONG>Art und <BR>Weise</STRONG> zeigen, wie
das Internet und World Wide Web funktioniert.</P>

<P>Dieser Text erscheint          ziemlich
durcheinander
enthält aber Beispiele für HTML-Markierungen. Hier ist z.B. eine
"unsortierte Liste":</P>
<UL>
   <LI>Ein Punkt der Liste,
   <LI>Ein zweiter Punkt <LI>Ein dritter Punkt, der über einige Zeilen
   geht und obwohl am rechten Rand ein Zeilenumbruch erfolgt, vom
   Browser in Listenart sauber formatiert wird.
   <LI>Letzter Punkt.
</UL>
<P>Listen sind wichtig. Es gibt auch geordnete Listen, welche
nummeriert sind und beschreibende Listen.</P>
<HR>
<P>Man kann auch horizontale Linien zeichnen, die nützlich sind,
um Bereiche voneinander zu trennen.</P>
</BODY>
</HTML>
```

Struktur eines HTML-Dokuments:

Das HTML-Dokument wird in den Tags <HTML> ... </HTML> eingeschlossen, ist also selbst ein Element.

Das **HEAD**-Element gehört nicht zum eigentlichen Text, der dargestellt wird, sondern enthält Informationen über das Dokument. So kann z.B. ein Titel mit **TITLE** gewählt werden. Dieser wird abseits angezeigt. Mit dem Netscape-Browser z.B. in der Titelzeile des Fensters.
Der Titel dient auch zum Auffinden von Seiten im WWW und sollte daher kurz und aussagekräftig sein. Der HEAD-Tag muss am Anfang, also direkt nach dem HTML-Tag stehen.

Das **BODY**-Element umfasst den Bereich, der schließlich im Anzeigebereich des Browsers dargestellt wird.

Mehrfache Leerzeichen, Tabulatoren und Zeilenumbrüche werden als einfacher Wortabstand verarbeitet. Für sofortigen Zeilenwechsel dient der <**BR**> - Tag.
Zusammengehörige Abschnitte werden durch den Paragraph-Tag <P> ... </P> eingefasst. Hierbei kann der Ende-Tag entfallen, wenn ein neuer Beginn erfolgt.
Zur Strukturierung in Texten dienen verschiedene Elemente, wie Überschriften (**H1**,..., **H6**) oder z.B. Listen (**UL**, **OL**, **DL**), die auch geschachtelt werden können. Außerdem können z.B. Bilder eingefügt werden mit **IMG**, dem Bild-Element, und Verweise auf andere Stellen und Seiten, sogenannte Hypertext-Links bzw. Anker-Elemente (**A**).

Das folgende Beispiel umfasst diese Elemente:

```
<HTML>
<HEAD>
    <TITLE> Beispiel 2a, Darstellung von IMG und Hypertext Links </TITLE>
</HEAD>
<BODY>
<H1> Beispiel 2a: Einfügen von Bildern und Hypertext Links </H1>

<P> Grüße aus der aufregenden Welt der HTML-Beispiel-Dokumente. OK, der Text
    ist nicht so aufregend. Aber wie wäre es mit einigen Bildern!</P>

<P> Es gibt viele Wege, Bilder einzubauen. Zum Beispiel kannst du es auf
    diese Weise erledigen:
    <IMG SRC="home.gif" ALIGN="top">, so
    <IMG SRC="home.gif" ALIGN="middle"> oder auch so
    <IMG SRC="home.gif" ALIGN="bottom">.</P>

<P> Ein weiterer wichtiger Punkt: Du kannst
    <A href="beispiel2b.html">hypertext links</A> zu anderen Dateien erstel-
    len. <BR>
    Dieser <A href="beispiel2b.html"> zweite link </A> hat Leerzeichen zwi-
    schen den Anfangs- und Ende-Tags, und dem eingeschlossenen Text.<BR>
    Du kannst auch Bilder in hypertext links einfügen, zum Beispiel:
    <A href="beispiel2b.html"><IMG SRC="pfeilr.gif" ALIGN="middle"></A>.<BR>
    Du kannst den Rahmen um das Bild entfernen durch die Einstellung  des
    Attributes BORDER="0" im IMG-Element. Zum Beispiel:
    <A href="beispiel2b.html"><IMG SRC="pfeilr.gif" BORDER="0"
    ALIGN="middle"></A>.</P>

<P> Schließlich ist hier noch eine Bilderreihe:<BR>
    <IMG src="home.gif" alt="[Home Icon]">
    <IMG src="home.gif" alt="[Home Icon]">
    <IMG src="home.gif" alt="[Home Icon]">
    <IMG src="home.gif" alt="[Home Icon]">
    <IMG src="home.gif" alt="[Home Icon]">
    <IMG src="home.gif" alt="[Home Icon]">
    <IMG src="home.gif" alt="[Home Icon]"></P>
</BODY></HTML>
```

Image-Elemente: `<IMG SRC="Bildquelle">`

Zur Angabe der Bildquelle dient das **SRC**-Attribut (source). Übliche Bildformate sind GIF und JPEG.
Bilder können zum umgebenden Text ausgerichtet werden mit dem **ALIGN**-Attribut. Die häufigsten Werte sind ”top“, ”middle“ und ”bottom“.
Das **ALT**-Attribut sorgt für einen alternativen Text, wenn das Bild nicht angezeigt werden kann. Dieser Text erscheint auch im Netscape-Browser oder Internet Explorer als Hinweis, wenn der Mauszeiger darauf zeigt.
Bei großen Bildern kann das Laden der Bilddaten einige Zeit dauern. Bis der Browser die Seite anzeigen kann benötigt er aber zumindest die Bildgröße. Daher kann diese zusätzlich mit den Attributen **WIDTH** und **HEIGHT** angegeben werden. Die Angabe erfolgt dabei in Pixeln.

Anchor-Elemente: `<A HREF="Ziel">Hinweistext</A>`

Der Verweis zur nächsten Seite erfolgt mit dem Attribut **HREF**. Der Text, der Zwischen Anfangs- und Ende-Tag steht wird hervorgehoben und dient als Bereich zum Anklicken mit der Maus. Werden Bilder verwendet, so können „Buttons“ gestaltet werden, wie das von anderen Programmen bekannt ist. Hier empfiehlt sich dann die Verwendung von ALT, für Browser, welche keine Grafiken anzeigen.

Hier noch die nachfolgende Seite:

```
<HTML>
<HEAD>
    <TITLE> Beispiel 2b: Ziel des Beispiels Hypertext Link </TITLE>
</HEAD>
<BODY>

<H2> Ziel des Hypertext-Links </H2>

<P>  OK! Nun bist du also hier, wie kommst wieder zurück? Dieses Dokument
     enthält keine Hypertext-Links. Du musst also den Zurück-Knopf in der
     Symbolleiste verwenden, um wieder zurückzukommen zum vorher angezeig-
     ten Dokument.

</BODY></HTML>
```

Grundsätze zum Erstellen einer *Home-Page*

- Home-Pages sollten klein gehalten werden ohne große Grafiken und Texte. Umfangreiche Bilder sollten anderswo abgelegt sein. Bilder und Icons sollten weniger als 20 kB umfassen. Mit einem 56 k-Modem dauert das Laden dafür bereits 5 Sekunden. Niemand will beim Besuch einer Home-Page lange warten!
- Da eine Home-Page als Einführung in eine Sammlung von Dokumenten und Informationen dient, sollte sie auch entsprechend erklärend sein und angeben wo welche Daten zu finden sind. Die Beschreibung zu den Verweisen sollte kurz gehalten werden und entsprechende Links enthalten. Ein „Help“ könnte zusätzliche Erläuterungen bieten. Bei mehrsprachigen Seiten sollten entsprechende Wahlmöglichkeiten angeboten werden.
- Eine Home-Page sollte nicht (allein) auf Grafiken basieren. Viele „Surfer“ schalten das automatischen Laden von Grafiken aus Zeitgründen ab. Es sollte daher geprüft werden, ob alle Informationen ohne Grafik erfassbar sind. Man verwende insbesondere IMG-Elemente mit dem ALT-Attribut.
- Es empfiehlt sich auch eine Kontaktadresse anzugeben, z.B. in Form einer E-Mail-Adresse. Üblicherweise erfolgt das am Fuß der Home-Page.

Um die Form und Gliederung von HTML-Seiten zu beeinflussen gibt es zahlreiche Elemente, wie etwa BLOCKQUOTE, OL (ordered list), FORM, TABLE usw. Die Beispiele 3 bis 5 behandeln textbezogene Dokumente. Ein üblicher Gebrauch von HTML dient für Online-Dokumentationen oder Online-Sammlungen für Materialien. Hier kann es sich natürlich um sehr umfangreiche Texte handeln. Somit ist eine Gliederung nötig.

Das PRE-Element: `<PRE> ... </PRE>`

Der Text bleibt entsprechend "vorformatiert" (preformatted), d.h. es bleiben zusätzliche Leerzeichen oder Zeilenumbrüche erhalten. Außerdem findet ein schreibmaschinenähnlicher Zeichensatz mit konstanten Zeichenabständen Verwendung. Innerhalb eines PRE-Elements sind nur STRONG-, EM- und Anchor-Elemente verwendbar. Anwendung ist z.B. zur Darstellung von Code innerhalb eines HTML-Dokuments.

Das Blockquote-Element: `<BLOCKQUOTE> ... </ BLOCKQUOTE >`

Hierdurch wird der Text vom übrigen Text abgesetzt. z.B. etwas eingerückt.

Allgemeines:

Die **einzelnen HTML-Dokumente sollten kurzgehalten** werden und sich auf maximal 2-3 Bildschirmseiten beschränken. Besser ist es **Links** zu verwenden.
Jedes Dokument sollte Navigationshilfen enthalten also z.B. Links zum nächsten und vorhergehenden Dokument.
Jedes Dokument sollte ein einheitliches Aussehen bieten.
Bei umfangreichen Dokumentationen sollte eine entsprechende Gliederung aufgebaut werden und ein **Inhaltsdokument** vorliegen, von welchem man über Links alle Dokumente rasch erreichen kann. Auf dieses Dokument sollte von jedem ein Link z.B. *Index* bestehen.
Zum **Download des Gesamttextes** kann ein Hinweis als entsprechender Link enthalten sein. Dies ist z.B. nützlich zum Offline-Lesen oder Ausdrucken.

Beispiele:

```
<HTML>
<HEAD>
<TITLE>Beispiel 3: HR-Element in HTML</TITLE>
</HEAD>
<BODY>

<H1> Horizontale Linien </H1>

<P>Das HR-Element dient zum Zeichnen einer horizontalen Linie, welche den
Bildschirm quer teilt.
Man benutzt sie, um logisch unterschiedliche Text-Bl&ouml;cke voneinander
zu trennen oder
eine Liste von ICONS vom Textk&ouml;rper.</P>

<P>Das HR-Element ist leer. (Man ben&ouml;tigt also auch kein &lt;/HR&gt;).
</P>
<H2> Beispiel </H2>
Nachfolgender Text demonstriert den Gebrauch von &lt;HR&gt;:
<BLOCKQUOTE>
<PRE>
&lt;P&gt; Folgendes Dokument ist der Seite einer Verpackung von S-Express
entnommen.&lt;/P&gt;
&lt;HR&gt;
&lt;H1&gt; S-Express - Der Drink aus der Schoko-Fabrik &lt;/H1&gt;
&lt;P&gt; S-Express. Der Drink mit sch&ouml;n viel Schokogeschmack.
Kommt ja auch aus der Schokofabrik. Und f&uuml;r die Power:
wertvolle Vitamine und Traubenzucker. Einfach hei&szlig;.
```

```
Auch in kalter Milch.&lt;/P&gt;
</PRE>
</BLOCKQUOTE>
<P> <B> Dies wird folgenderma&szlig;en dargestellt: </B></P>
<P> Folgendes Dokument ist der Seite einer Verpackung von S-Express entnom-
men.</P>
<HR>
<H1> S-Express - Der Drink aus der Schoko-Fabrik </H1>
<P> S-Express. Der Drink mit sch&ouml;n viel Schokogeschmack.
Kommt ja auch aus der Schokofabrik. Und f&uuml;r die Power:
wertvolle Vitamine und Traubenzucker. Einfach hei&szlig;.
Auch in kalter Milch.</P>
<HR>

</BODY>
</HTML>
```

Hinweis:

Wenn Tags <TAG> in einen Text aufgenommen werden sollen, muss das "<"-Zeichen umschrieben werden, da es ja sonst als Tag ausgeführt wird. Solche Umschreibungen werden mit dem &-Zeichen eingeleitet und einem ; abgeschlossen. Diese Umschreibung erfolgt auch für andere Sonderzeichen, so z.B. auch für Umlaute, denn nicht jeder Browser erkennt Umlaute in einem vorliegenden Text.

<:	<	>:	>	ß:	ß
Ä:	Ä	Ö:	Ö	Ü:	Ü
ä:	ä	ö:	ö	ü:	ü

```
<HTML>
<HEAD>
<TITLE> Beispiel 4: HTML Inhaltsliste </TITLE>
</HEAD>
<BODY>
<H1> HTML Dokumentation Inhalts&uuml;bersicht </H1>

Archiv-Dateien zum <A href="archives.html">[Download]</A>
<DL>
  <DT><A href="htmlindex.html">Inhalts&uuml;bersicht (diese Seite)</A>
  <DT><A href="about_the_autor.html">&Uuml;ber den Autor</A>
</DL>
<OL>
  <LI><A href="intro.html">Einf&uuml;hrung in dieses Dokument</A>
  <LI><A href="html_intro.html">Einf&uuml;hrung in HTML</A>
  <OL>
    <LI><A href="elements.html">HTML Elemente</A>
    <LI><A href="doc_struct.html">HTML Dokumentstruktur</A>
    <LI><A href="naming.html">HTML Namensgebung</A>
  </OL>
  <LI><A href="head.html">Kopf</A> eines HTML-Dokuments
  <OL>
    <LI><A href="title.html">TITLE</A>
    <LI><A href="isindex.html">ISINDEX</A>
    <LI><A href="nextid.html">NEXTID</A>
    <LI><A href="link.html">LINK</A>
    <LI><A href="base.html">BASE</A>
  </OL>
  <LI><A href="body.html">K&ouml;rper</A> eines HTML-Dokuments
  <OL>
    <LI><A href="headings.html">&Uuml;berschriften</A> (Hn)
    <LI><A href="paragraph.html">Abs&auml;tze</A> (Paragraphen) (P)
    <LI><A href="line_break.html">Zeilenumbr&uuml;che</A> (BR)
```

```
    <LI><A href="inl_img.html">Zeilenorientierte Bilder</A> (IMG)
    <OL>
      <LI><A href="ex_img.html">Beispiele</A> zu Bildern
    </OL>
    <LI><A href="anchor.html">Hypertext Anker</A> (A)
    <OL>
      <LI><A href="link_to.html">Link auf</A> ein Objekt (HREF)
      <LI><A href="link_from.html">Link von</A> einem Objekt (NAME)
      <LI><A href="relation.html">Beziehungen</A> zwischen Objekten (REL)
    </OL>
    <LI><A href="address.html">E-Mail-Adressen</A> (ADDRESS)
    <LI><A href="hr.html">Horizontale Linien</A> (HR)
    <LI>....
  </OL>
  <LI>....
</OL>
</BODY>
</HTML>
```

Hinweis:

Dieses Beispiel zeigt auch das Verschachteln von Listen.

Hier noch ein Beispiel für Archivlisten:

```
<HTML>
<HEAD>
<TITLE> Beispiel 5: HTML Archivliste </TITLE>
</HEAD>
<BODY>
<H2> HTML Dokumentation Archive </H2>

<P> Die Archiv-Dateien liegen in folgenden Formaten vor:
<OL>
  <LI><A href="alldocs.zip">alldocs.zip</A> (150 KBytes)
  -- <EM> DOS PKZIP </EM>
  <LI><A href="alldocs.tar">alldocs.tar</A> (150 KBytes)
  -- <EM> UNIX tar </EM>
  <LI><A href="alldocs.tar.Z">alldocs.tar.Z</A> (90 KBytes)
  -- <EM> UNIX tar (compressd) </EM>
  <LI><A href="alldocs.htnl">alldocs.html</A> (150 KBytes)
  -- <EM> Concatenated </EM>
  <UL>
    <LI><EM> Diese Datei ist eine gek&uuml;rzte Form des Dokuments,
             gedacht f&uuml;r den Druck vom Browser aus. Die Hypertext-
             Links wurden entfernt. </EM>
  </UL>
</OL>
<HR noshade>
<B> Letze Aktualisierung: </B> <EM> 1998-11-12 </EM> --
    &lt;<A href="mailto:Dehmer@gyneu.ka.bw.schule.de">
    Dehmer@gyneu.ka.bw.schule.de</A>&gt;
</BODY>
</HTML>
```

Einbinden von Dateien und Applets
Verwendung von Java-Script

Mit dem **EMBED**-Element können beliebige Dateien in das HTML-Formular eingebunden werden. Verfügt der Browser über ein entsprechendes Plugin zur Darstellung, so werden die Daten direkt in das HTML-Formular eingebaut. Ansonsten wird gegebenenfalls die zugehörige Anwendung aufgerufen. Dies setzt voraus, dass die Datei z.B. an Hand ihrer Endung erkannt wird.

Beispiel 6:

```
<HTML>
<HEAD>
<TITLE> Beispiel 6: EMBED-Element </TITLE>
</HEAD>
<BODY>
<H1>Beispiel zum EMBED-Element</H1>
<BLOCKQUOTE>
<P>Dieses einfache Beispiel demonstriert das <B>EMBED</B>-Element, wie
   es zur Zeit in Netscape und einigen anderen Browsern integriert ist.<BR>

<EMBED SRC="img/doghouse.psd" WIDTH="192" HEIGHT="170"></P>

<P>Hier ist ein weiteres <B>EMBED</B>, dieses Mal für eine Audio-Datei. Das
   eingebundene Plugin ist ein Audio-Steuerpult -

<EMBED SRC="img/sound.wav" WIDTH="145" HEIGHT="20" HSPACE="10" VSPACE="5">

   das Pult wird behandelt wie eine Grafik.</P>

<P>Das Plugin kann auch links oder rechts ausgerichtet werden wie Bilder.
<EMBED SRC=" img/sound.wav" WIDTH="145" HEIGHT="20" HSPACE="10" VSPACE="5"
ALIGN="Left">
<EMBED SRC=" img/sound.wav" WIDTH="145" HEIGHT="20" HSPACE="10" VSPACE="5"
ALIGN="Right">
Hier sind zwei Beispiele. Achte darauf, wie der Text die Steuerpulte um-
fließt, genau wie bei normalen Bildern.</P>
</BLOCKQUOTE>
</BODY>
</HTML>
```

Übung:

Suche nach anderen Dateien und versuche diese in ähnlicher Weise in ein HTML-Formular einzubauen.

Das folgende Beispiel 7 zeigt den Einbau eines **Java-Applets**. Dabei handelt es sich um Programme, die z.B. interaktiv mit dem Benutzer arbeiten.

Der Einbau eines Applets erfolgt mit dem **APPLET**-Element

```
< APPLET code="Appletdatei.class>...</APPLET >
```

Mit `<PARAM name="Variablenname" value="Inhalt">`
können Variablen des Applets mit Werten versorgt werden.
Nachfolgendes Beispiel zeigt dies.

```
<HTML>
<HEAD>
<TITLE> Beispiel 7: APPLET-Element </TITLE>
</HEAD>
<BODY>
<H1>Beispiel zu einem eingebetteten Applet</H1>
<BLOCKQUOTE>
<P>Dies ist ein einfaches Beispiel f&uuml;r ein eingebettets
<STRONG>APPLET</STRONG>-Element,
<APPLET code="SineText.class" codebase="java" width="500" height="100">
  <param name="text" value=" Guten Morgen !!  Wie geht's ?">
  <param name="traveling" value="no">
  <param name="rate" value="2">
  <param name="background" value="ff0101">
  <param name="foreground" value="003399">
      <BLOCKQUOTE>
        Wenn Ihr Browser Java-f&auml;hig ist, sehen sie eine Animation an
        Stelle dieses Textes!
      </BLOCKQUOTE>
</APPLET>
in diesem Fall ein Java-Applet, das eine sinusf&ouml;rmige Laufschrift de-
monstriert. Der Bereich reagiert auf Klick, wobei sich die Richtung
ver&auml;ndern l&auml;sst.</P>
<P>Beachte, dass der Text-Inhalt des Applets im HTML-Dokument nicht ange-
zeigt wird, au&szlig;er der Browser versteht die Applet-Anweisung nicht o-
der ist nicht f&auml;hig, Java-Applets ablaufen zu lassen.</P>
<P>Das Applet-Beispiel stammt von Katharina Baulig.</P>
</BLOCKQUOTE>
</BODY>
</HTML>
```

Java-Script ist eine von der Programmiersprache *Java* abgeleitete *Script*-Sprache mit eingeschränkten Programmieranweisungen. Es können damit z.B. keine Dateioperationen ausgelöst werden.
Mit **Java-Script** lassen sich in ein HTML-Dokument Funktionalitäten einbauen, die diese Dokumentbeschreibungssprache als solche nicht besitzt. Dadurch können z.B. Aktionen ausgelöst werden oder auf solche des Benutzers reagiert werden. Auch das Erscheinungsbild des aktiven Dokuments kann damit beeinflusst werden.
Der Programmcode kann beliebig positioniert sein, wird aber z.B. bei der Deklaration von Funktionen häufig im HEAD-Tag untergebracht.
Damit der Programmtext vom Browser nicht interpretiert wird, muss dieser auskommentiert werden mit `<!-- Java-Code -->`. Das Ende-Zeichen "`-->`" muss wiederum für den Java-Interpreter auskommentiert werden mit "`//`", dadurch entsteht `<!-- Java-Code //-->`.
Zum Einbau dient das **SCRIPT**-Element:

```
<SCRIPT language="JavaScript"> ... </SCRIPT>
```

Beispiel 8 erzeugt ein Nachrichtenfenster:

```
<HTML>
<HEAD>
<TITLE> Beispiel 8: Hallo Welt </TITLE>
</HEAD>
<BODY>
  <SCRIPT language="JavaScript">
  <!--
   alert("Hallo Welt!");
  //-->
  </SCRIPT>
</BODY>
</HTML>
```

Grafiken

Als nächstes noch einige Hinweise zu Grafiken. Häufig benötigen Grafiken wegen der großen Datenmenge einige Zeit für den Übertrag über das Netz. Es empfiehlt sich daher die Größe des Bildes anzugeben. Der Browser lässt dann zunächst eine entsprechende leere Fläche und stellt wenigstens den Rest der Seite (Textbereiche) dar. Um dem Betrachter vorab schon einmal eine Grobstruktur zu geben kann die Grafik auch "interlaced", d.h. die Zeilen werden in größeren Abständen übertragen und danach die Zwischenzeilen aufgefüllt, so dass die Feinheiten zusehends besser erkennbar sind. Hierzu ist ein Grafikprogramm nötig, das solche GIF-Dateien (GIF 89A) produziert. Es können auch animierte GIF-Dateien erzeugt werden. Diese bestehen aus einer Folge von Bildern, welche nacheinander im selben Rahmen angezeigt werden. Auch hierzu ist geeignete Grafiksoftware nötig.

Beispiele:

```
<HTML>
<HEAD>
<TITLE> Beispiel 9: Grafiken </TITLE>
</HEAD>
<BODY>
<H1>Beispiel zum IMG-Element</H1>
<BLOCKQUOTE>
<P>Dieses einfache Beispiel demonstriert einige M&ouml;glichkeiten der Darstellung von Grafiken. Die folgende Grafik wurde <B>interlaced</B>
abgespeichert. Beachte den Aufbau des Bildes, zun&auml;chst als Grobdarstellung welche dann verfeinert wird.<BR>

<IMG SRC="img/raft.gif" WIDTH="640" HEIGHT="427" VSPACE="20"><BR>

Hier eine kleine Animation:<BR>

<IMG SRC="img/earth_s.gif" WIDTH="64" HEIGHT="64" VSPACE="20">

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

Mapping

Sicher sind Ihnen im Internet schon Bilder begegnet, in welchen Bereiche angeklickt werden können und damit auf andere Seiten verzweigt wird. Dieses Verhalten wird erreicht, durch überziehen des Bildes mit einer "Bildkarte" (MAP). In dieser werden die aktiven Bereiche festgelegt einschließlich der zugehörigen *Links*. Am besten werden als Bilder solche vom GIF-Format verwendet, da ältere Browser aktive JPEGs nicht behandeln können. Auch muss das HEIGHT- und WIDTH-Attribut mit der tatsächlichen Bildgröße übereinstimmen.
Das IMG-Element bekommt zunächst eine Erweiterung, die auf die Bildkarte verweist:

`<IMG SRC="MeinBild.gif" **USEMAP="#MeinMap"**>`.

Der Zusatz **USEMAP** gibt also an, dass eine Bildkarte mit dem Namen *"MeinMap"* zu verwenden ist. Diese Bildkarte wird festgelegt mit dem **MAP**-Element:

```
<MAP NAME="MeinMap">
  <AREA SHAPE="circle" COORDS="116,40,30" HREF="Datei1.html">
  <AREA SHAPE="rect" COORDS="36,30,84,90" HREF="Datei2.html">
  <AREA SHAPE="poly" COORDS="86,154,64,170,76,188,104,184,112,166"
        HREF="Datei3.html">
</MAP>
```

Bestandteil des MAP-Elements als Attribut ist der Name, da eine HTML-Seite mehrere MAPs beinhalten kann. Das AREA-Element legt den "Klickbereich" im Bild fest, der den

entsprechenden Link erzeugt. Als mögliche Attribute können die Form (SHAPE) des Klickbereiches angegeben werden, die hierzu benötigten Koordinaten (COORDS) und natürlich die Adresse der aufzurufenden Seite (HREF), was ja vom ANKER-Element bereits bekannt ist.

Bereichsform (SHAPE)	**Benötigte Koordinaten (COORDS)**
circle (Kreis)	x_M,y_M,r (Mittelpunkt M und Radius r)
rect (Rechteck)	$x_{ol},y_{ol},x_{ur},y_{ur}$ (Eckpunkte oben links und unten rechts)
poly (Vieleck)	$x_1,y_1,x_2,y_2,x_3,y_3,...,x_n,y_n$ (aufeinanderfolgende Eckpunkte)

Hinweis: Für die Koordinaten gilt die obere linke Ecke des Bildes als Nullpunkt. Die x-Werte sind nach rechts, die y-Werte nach unten gerichtet. Die Angaben erfolgen in Pixeln und sind positiv bis zum Wert der Breite bzw. der Höhe des Bildes.

Übung

Betrachte ein Bild und notiere dessen Größe. Denke dir eine Karte darübergelegt. Fertige evtl. eine Skizze hierfür an. Suche kreisförmige, rechteckige oder beliebige Bereiche, die beim Anklicken *Links* erzeugen sollen. Schätze möglichst genau an Hand der Bildgröße die Koordinaten ab. Erstelle dann die zugehörige Bild-MAP.

Tabellen

Eine einfache Methode eine Seite in rechteckige Bereiche einzuteilen bieten Tabellen. Dabei können die Bereiche unsichtbar sein, womit z.B. Texte neben Grafiken erstellt werden können, oder die Darstellung mehrspaltigen Textes ermöglicht wird, als auch sichtbar mit Randbegrenzung, also im Sinne einer Tabelle erfolgen.

Zur Kennzeichnung dient das TABLE-Element: <TABLE> ..... </TABLE>

Die Reihen der Tabelle werden mit dem Table-*Row*-Element festgelegt: <TR> .... </TR>

Die einzelnen Felder einer Reihe – in der Gesamtheit ergeben diese die Spalten – werden mit dem Table-*Data*-Element definiert: <TD> .... </TD>

Beispiel:

Diese Tabelle besteht aus drei Reihen mit je zwei Feldern, also zwei Spalten.

```
<TABLE BORDER="1">
  <TR>
    <TD>Daten1</TD>
    <TD>Daten2</TD>
  </TR>
  <TR>
    <TD>Daten3</TD>
    <TD>Daten4</TD>
  </TR>
  <TR>
    <TD>Daten5</TD>
    <TD>Daten6</TD>
  </TR>
</TABLE>
```

Die Tabelle ist also von folgender Art:

Daten1	Daten2
Daten3	Daten4
Daten5	Daten6

Beachten, dass die Anzahl der Felder in jeder Reihe gleich ist, da sonst die Tabelle am Ende Lücken aufweist. Falls kein Rand gesetzt wird, sind Lücken allerdings unerheblich.
Als Daten können auch Bilder etc. dienen. Das Setzen des Rahmens (BORDER) wird durch das Attribut des TABLE-Elements festgelegt. Auch das TD-Element verfügt über einige interessante Attribute.

Tabellen können auch geschachtelt werden, d.h. dass innerhalb eines TD-Elements auch eine Tabelle eingefügt werden kann.

Außerdem sind Tabellenfelder möglich, die Spalten (Column) überspannen. Hierzu dient das Attribut COLSPAN="Spaltenzahl" im TD-Element. Ebenso kann ein Datenfeld mehrere Reihen (Row) überspannen. Dazu wird ROWSPAN="Reihenzahl" des TD-Elements verwendet. Überspannte Felder dürfen dann in der Struktur nicht auftauchen.
Damit sind der Darstellungsart von Tabellen kaum Grenzen gesetzt.

Beispiel:

```
<TABLE BORDER="1">
  <TR>
    <TD COLSPAN="2">Daten1</TD>
    <TD> </TD>  (entfällt, da
                 überspannt)
  </TR>
  <TR>
    <TD>Daten2</TD>
    <TD>Daten3</TD>
  </TR>
  <TR>
    <TD>Daten4</TD>
    <TD>Daten5</TD>
  </TR>
</TABLE>
```

Darstellung der Tabelle:

Daten1	
Daten2	Daten3
Daten4	Daten5

Übung:

1. Erzeuge folgende Tabelle:

Daten1	Daten2	Daten3	
Daten4	Daten5	Daten6	Daten7
	Daten8	Daten9	Daten10

Übung:

2. Verwende eine Tabelle um einen Text seitlich von einem Bild über mehrere Zeilen zu erstellen. Richte den Text wahlweise oben, unten oder in der Mitte zentriert neben dem Bild aus.
3. Verteile mehrere Wörter (z.B. 9) auf einer Seite in Form eines X. Verwende hierfür eine geeignete Tabelle ohne Rand. Benutze auch die Attribute für die Größe der Tabelle, bzw. der Felder.

Frames

Frames erlauben eine ähnliche Aufteilung einer Seite wie Tabellen, jedoch stehen die einzelnen Datenbereiche unabhängig zur Anzeige von HTML-Dateien zur Verfügung. So können einzelne Bereiche statisch, die anderen dynamisch genutzt werden. Z.B. bleibt beim Weiterblättern einer Seite das Inhaltsverzeichnis im andern Bereich statisch, also in Ruhe. Zur Nutzung von Frames muss eine HTML-Datei erstellt werden, die nur den Aufbau, also die Einteilung der Anzeigefläche enthält, sowie die Dateiangaben, die in die verschiedenen Bereiche geladen werden sollen. Gewöhnlich wird hierfür die Start-Datei also z.B. *index.html* benutzt. Zur Festlegung dient das FRAMESET-Element:

```
<FRAMESET ROWS="110,*" BORDERCOLOR="#FFFFFF" BORDER="1" FRAMEBORDER="1"
          FRAMESPACING="0"> ..... </FRAMESET>
```

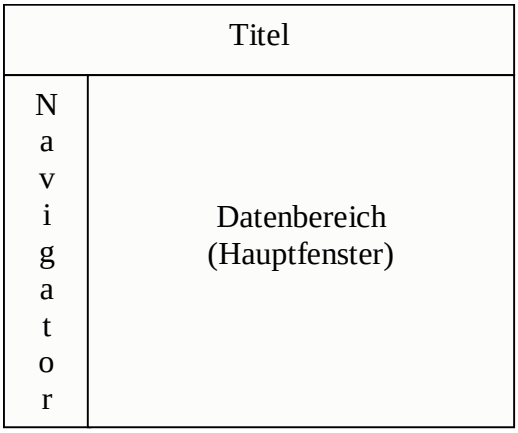
Mit Attributen wird z.B. die Anzahl und Größe der Reihen festgelegt:

`ROWS="110,*"` bedeutet: erste Reihe 110 Punkte hoch, zweite Reihe erhält als Höhe den restlichen Platz.
`FRAMESPACING="0"` legt den Abstand zwischen den Frames fest, in diesem Fall kein Abstand.

Das FRAME-Element selbst legt schließlich die Angabe der anzuzeigenden HTML-Datei fest. Außerdem kann ein Rahmen und Ränder definiert werden oder Scroll-Balken:

```
<FRAME SRC="titel.html" NORESIZE SCROLLING="NO" MARGINHEIGHT="0"
       MARGINWIDTH="0" FRAMEBORDER="NO">
```

Eine typische Anordnung sieht etwa so aus:




```
<FRAMESET ROWS="110,*"
  BORDERCOLOR="#FFFFFF" BORDER="1"
  FRAMEBORDER="1" FRAMESPACING="0">
  <FRAME SRC="titel.html" NORESIZE
    SCROLLING="NO" MARGINHEIGHT="0"
    MARGINWIDTH="0" FRAMEBORDER="NO">
  <FRAMESET COLS="160,*"
    BORDERCOLOR= "#FFFFFF" border="1"
    FRAMEBORDER="1" FRAMESPACING="0">
    <FRAME SRC="navigator.html"
      NAME="Navigator" NORESIZE
      FRAMEBORDER="NO">
    <FRAME SRC="main.html"
      NAME="HauptFenster"
      FRAMEBORDER="NO">
  </FRAMESET>
</FRAMESET>
```

Hierbei enthält die zweite Reihe des ersten Framesets wieder ein Frameset aus zwei Spalten. Die Namen dienen später als Verweise, in welchem Fenster eine Aktion ablaufen soll. Hierzu wird z.B. im Navigatorbereich mit `<BASE TARGET="HauptFenster">` das Standardausgabefenster festgelegt.

Die komplette Deklaration erfolgt im HEAD-Bereich. Im BODY-Bereich kann ein alternativer Text angezeigt werden, für Browser, welche Frames nicht unterstützen.

Lädt man in einem Frame-Fenster eine fremde Adresse, so bleibt auch hierbei die Umgebung erhalten, d.h. der Betrachter hat den Eindruck, dass er immer noch Seiten des ursprünglichen Servers nutzt. Es gehört zum guten Ton, solche Dateien in einem neuen Fenster zu laden.

Hierzu wird im Anker-Element das Attribut TARGET="_BLANK" benutzt