HTML - Hyper Text Markup Language

inhalt

autor: **B. Hartard**

stand: **2.4 / 15.08.2023**

HTML - Hyper Text Markup Language

1. grundlagen

Die folgende dokumentation ist keine vollständige beschreibung von HTML, sondern vermittelt nur grundkenntnisse, die notwendig sind, um die dokumentationen von CSS, PHP und MySQL zu verstehen und darauf aufbauend einfache WEB-seiten zu gestalten.

1.1 formalismus

In dieser dokumentation werden die wichtigsten HTML-anweisungen beschrieben; dabei gilt folgender formalismus:

normale schrift	normal- oder fett-schrift: die angabe ist genau so zu schreiben
kursiv oder ***kursiv***	das ist ein platzhalter für einen wert, der hier anzugeben ist
[ ]	angaben in eckigen klammern können wahlweise gemacht werden.
. . .	die vorangehende angabe kann mehrfach wiederholt werden
`Courier`	beispiel in HTML-code

1.2 HTML-versionen

HTML ist eine beschreibungs- oder auszeichnungssprache, mit der inhalt und aussehen von WEB-seiten definiert werden. Die aktuelle version ist **HTML5**, die version **HTML4** wird aber nach wie vor in großem umfang eingesetzt. Die beiden versionen unterscheiden sich fundamental.

HTML4
enthält sprachelemente, mit denen der inhalt einer seite **und** die formatierung (schriftart, schriftdicke usw.) des inhalts beschrieben werden. Obwohl HTML4 inzwischen als veraltet gilt, werden in der vorliegenden dokumentation noch viele sprachelemente dieser version beschrieben.

HTML5
Hier fehlen alle sprachelemente zur formatierung des seiteninhalts, die formatierung muss vielmehr mit **CSS** erfolgen.

hinweis
Da CSS nicht bestandteil dieser dokumentation ist, werden elemente von CSS nur verwendet und beschrieben, wo es unumgänglich notwendig ist. Bei den meisten beispielen muss daher auf eine formatierung verzichtet werden. Das ist durchaus zulässig, der browser verwendet in diesem fall die für den browser festgelegte standardformatierung.

1.3 HTML-datei

Die beschreibung einer WEB-seite kann mit einem beliebigen texteditor erstellt werden und wird in einer normalen textdatei gespeichert, allerdings muss man als suffix für den dateinamen nicht **.txt**, sondern **.html**, **.htm** oder **.php** verwenden.

Der dateiname kann aus buchstaben, ziffern und den zeichen bindestrich und unterstrich bestehen. Er sollte nicht allzu lang sein und keine geschlossenen umlaute (Ä, ä, usw) oder das scharfe ß enthalten. Die datei wird auf einen WWW-server hochgeladen und zur ausführung von einem browser aufgerufen, d.h. zum browser übertragen und am bildschirm dargestellt.

Mit der ersten zeile einer HTML-datei wird die version von HTML festgelegt (siehe 2.1 - DOCTYPE). Ist HTML5 festgelegt, dürfen trotzdem sprachelemente von HTML4 verwendet werden, alle browser verstehen das. Puristen bezeichnen das als schlechten stil, aber es ist sehr praktisch. Deshalb wird in dieser dokumentation und den beispielen auch so verfahren.

In der regel besteht eine WEB-site, homepage oder WEB-anwendung aus mehreren seiten (dateien), die durch **links** miteinander verknüpft sind. Die startseite hat dabei meist den namen index.htm, index.html oder index.php, weil dann beim aufruf nur die URL aber nicht der dateiname nötig ist.

1.4 HTML-tag

Eine WEB-seite enthält in der regel mehrere und verschiedenartige elemente (überschriften, absätze, usw.). Alle elemente sind durch sog. **tags** ausgezeichnet, d.h. in tags eingeschlossen (anfangs- und ende-tag). Das anfangs-tag enthält in die zeichen **< >** eingeschlossen einen **HTML-befehl** und wahlweise **attribute**. Das ende-tag enthält nur den gleichen befehl wie das zugehörige anfangs-tag und vor dem befehl einen schrägstrich. Ausnahmsweise gibt es tags, denen kein inhalt und kein ende-tag folgt. In diesem fall sollte dem abschließenden zeichen > ein zwischenraum und ein schrägstrich vorangehen.

<*befehl* [*attribut* . . .] > *inhalt* </*befehl*>

<*befehl* [*attribut* . . .] />

1.5 attribute

1.5.1 format

Ein attribut enthält zusätzliche informationen für den HTML-befehl; einige wenige attribute bestehen nur aus einem schlüsselwort. Manche attribute sind an bestimmte HTML-befehle gebunden, andere sind bei fast allen befehlen zulässig. Die meisten attribute haben folgendes format:

attribut="*wert*"

attribut	schlüsselwort
wert	zeichenkette oder numerische angabe in anführungszeichen oder apostrophe eingeschlossen (bei numerischen werten geht es auch ohne, ist aber kein sauberer stil)

1.5.2 attribute zur formatierung

Es gibt zahlreiche attribute um elemente einer seite zu formatieren (breite, höhe, schriftart usw). Diese attribute sind spezifisch für **HTML4** und gelten als veraltet Sie werden in dieser unterlage nicht verwendet..

1.5.3 attribute zur identifikation

id="***bezeichnung***"	identifikation des tags.
name="***bezeichnung***"	name des tags.

Mit ***bezeichnung*** wird einem element einer seite ein eindeutiger name zugewiesen, d.h. der name darf in einer seite nur einmal in einem id- oder name-attribut erscheinen. In CSS-vereinbarungen oder Javascript-anweisungen kann man sich auf diese namen beziehen.

1.5.4 attribute für CSS

Auch wenn **CSS** nicht bestandteil dieser dokumentation ist, werden hier zwei attribute kurz vorgestellt, die zum formatieren von elementen benötigt werden.

style-attribut
Mit diesem attribut werden einem tag CSS-vereinbarungen zur formatierung zugewiesen. Diese vereinbarungen ersetzen die bei nr. 1.5.2 erwähnten, veralteten attribute.

style="***eigenschaft***: ***wert***; [***eigenschaft***: ***wert***;] . . . "

Eine vereinbarung enthält eine ***eigenschaft*** und getrennt durch einen doppelpungt einen ***wert*** für diese eigen-schaft, abgeschlossen durch einen strichpunkt. Ohne weitere erklärung zeigt das folgende beispiel ein **style-**attribut, mit dem einem **p**-tag eine bestimmte schriftart zugewiesen wird.

```
<p style="font-family: Arial; font-size: 10pt; font-weight: normal; ">
      schriftart Arial, 10 punkt, normale schriftdicke mit style-attribut</p>
```

schriftart Arial, 10 punkt, normale schriftdicke mit style-attribut

Nachstehend werden einige vereinbarungen gezeigt, die in dieser unterlage häufig verwendet werden.

width: ***wert***;	breite eines elements;
height***:*** ***wert***;	höhe eines elements
size: ***wert***;	schrift- oder strichdicke
	wert ist eine numerische angabe unmittelbar gefolgt von **pt** für punkt oder **px** für pixel. Es sind auch andere angaben möglich, auf die hier nicht eingegangen wird.
color: ***farbe***;	textfarbe des elements; angabe für ***farbe*** s.u.
background-color: ***farbe*****;**	hintergrundfarbe; angabe für ***farbe*** s.u.

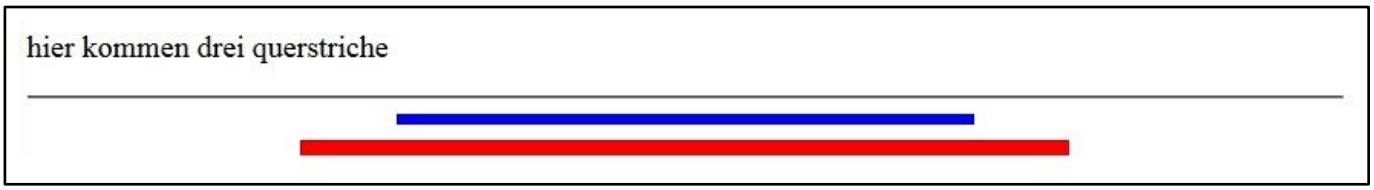
Eine ***farbe*** wird entweder als farbwort oder in sedezimaler schreibweise mit einem vorangestellten nummernzeichen angegeben, nachstehend die gängigsten werte:

farbwort	**sedezimal**	**farbwort**	**sedezimal**	**farbwort**	**sedezimal**	**farbwort**	**sedezimal**
black	#000000	silver	#BBBBBB	navy	#000080	blue	#0000FF
maroon	#800000	red	#FF0000	purple	#800080	fuchsia	#FF00FF
green	#008000	lime	#00FF00	teal	#008080	aqua	#00FFFF
olive	#808000	yellow	#FFFF00	gray	#808080	white	#FFFFFF

Weitere vereinbarungen, die mit dem style-attribut getroffen werden können, sind an geeigneter stelle erklärt.

Nachfolgend ein kleines beispiel für diese attribute: es werden drei querstriche angezeigt, dazu dient das **hr-tag**, ein allein stehendes tag (beschreibung siehe 4.3), einmal ohne attribut und dann mit dem style-attribut mit vereinbarungen für breite, höhe. farbe und hintergrundfarbe. Seltsamerweise kann man hier nicht mit **size** die strichdicke vereinbaren..

```
<p>hier kommen drei querstriche</p>
<hr>
<hr style="width: 300px; color: blue; height: 4px; background-color: blue;">
<hr style="width: 400px; color: #FF0000; height: 7px; background-color: #FF0000;">
```



class-attribut
Dem attribut wird als wert eine **CSS-klasse** zugewiesen, die an anderer stelle (meist im header oder einer eigenen datei) vereinbart ist. CSS-klassen enthalten oft viele vereinbarungen. Um die beispiele dieser dokumentation nicht unnötig zu verkomplizieren, wird gelegentlich das class-attribut verwendet. Das folgende beispiel zeigt die vereinbarung und verwendung der klasse **std**, mit der ein text mit schriftart Arial, 10 punkt angezeigt wird. So ähnlich sind auch die anderen klassen aufgebaut, die in der dokumentation verwendet, aber nicht näher erklärt werden.

```
<style type="text/css">
.std   {  font-family: Arial; font-style: normal;
          font-size: 10pt; font-weight: normal;
       }
</style>
```

```
<p class="std"> schriftart Arial, 10 punkt, normale schriftdicke mit class-
attribut</p>
```

schriftart Arial, 10 punkt, normale schriftdicke mit class-attribut

1.6 tags schachteln

Tags können beliebig tief geschachtelt werden, man muss nur darauf achten, dass man die ende-tags in genau der umgekehrten reihenfolge der anfangs-tags setzt, denn browser sind hier manchmal pingelig.

beispiel
Im beispiel werden p-, b- und i-tag geschachtelt (die erklärung der tags folgt noch).

```
<p>normal <b>fett <i>fett kursiv</i> wieder fett</b> wieder normal</p>
```

normal **fett** ***fett kursiv*** **wieder fett** wieder normal

2. HTML-seite

2.1 seitenstruktur

Eine seite im HTML-code beginnt mit dem **DOCTYPE**-tag, der rest der seite ist in das **html**-tag eingeschlossen. Dieser rest besteht aus zwei teilen: dem **header** (in das **head**-tag eingeschlossen) und dem **body** (in das **body**-tag eingeschlossen) mit dem anzuzeigenden teil der seite.

<!DOCTYPE . . .>	erste zeile der datei
<html [lang="de"] >	beginn der seite *)
<head>	beginn des seitenkopfs (header)
metadaten u.ä.	vgl. 2.2 header
</head>	ende des seitenkopfs
<body>	beginn der beschreibung
beschreibung der seite	
</body>	ende der beschreibung
</html>	ende der seite

*) Das wahlweise attribut **lang** ersetzt die entsprechende meta-anweisung im header.

!DOCTYPE

Die erste zeile einer HTML-datei muss diese zeile sein, sie teilt dem browser mit, dass die nachfolgende seite eine bestimmte version des HTML-codes enthält. Während es sonst gleichgültig ist, ob man die befehle der tags mit klein- oder großbuchstaben schreibt, sollte man sich hier genau an die folgende angaben und schreibweise halten. Das DOCTYPE-tag hat kein ende-tag.

achtung das ausrufezeichen ist zwingend erforderlich !!

<!DOCTYPE html>

Es folgt **HTML5**-code, das ist die moderne variante, die einige anweisungen enthält, die der ältere **HTML4**-code noch nicht kennt, außerdem sind einige völlig veraltete anweisungen nicht mehr zugelassen, aber auf die wird hier nicht eingegangen.

<!DOCTYPE HTML PUBLIC "-//W3C/DTD HTML 4.01 Transitional1//EN" "http://www.w3.org/TR/html4/loose.dtd>

Das ist ganz schön aufwendig und bedeutet **HTML4**-code, veraltet, wird aber von jedem browser verstanden. Da die browser HTML4-code auch dann verstehen, wenn HTML5 deklariert ist, kann man auf diese umständliche anweisung verzichten und grundsätzlich HTML5 deklarieren.

Hinweis
Was ist nun wenn man diese zeile wegläßt ? Auch damit werden die browser fertig, in der regel wird dann von HTML5 ausgegangen.

2.2 header

Der seitenkopf ist in das **head**-tag eingeschlossen und enthält angaben, die für den browser von bedeutung sind und die zum teil von den. suchmaschinen ausgewertet werden. Die angaben des header sind in der angezeigten seite nicht sichtbar.

2.2.1 zeichensatz

Mit der folgenden anweisung wird dem browser mitgeteilt, in welchem zeichencode die HTML-datei erstellt wurde. Die anweisung ist spezifisch für HTML5 und sollte die erste anweisung im header sein. Man handelt sich endlose schwierigkeiten mit geschlossenen umlauten und einigen sonderzeichen ein, wenn die datei nicht in dem hier angegebenen zeichensatz erstellt ist.

<meta charset="utf-8" | "iso-8859-1">

utf-8 das ist der zeichensatz für den sog. **Unicode**; mit diesem zeichensatz vermeidet man weitgehend die probleme mit den geschlossenen umlauten und einigen sonderzeichen.

iso-8859-1 das ist ein **Ansicode**-zeichensatz, immer noch weit verbreitet, aber man sollte möglichst darauf verzichten.

Weitere informationen zu zeichensätzen siehe 2.5.

2.2.2 seitentitel

<title> *titel der seite* </title>

Es ist ordentlicher stil, einer seite einen titel zu geben; der titel wird von den suchmaschinen ausgewertet.

2.2.3 metadaten

Es gibt zwei arten von metadaten

meta name
Sie haben das allgemeine format

<meta name="***bezeichnung***" content="***werte***">

bezeichnung bezeichnet den typ der metadaten

werte das sind die daten, die man dem browser mitteilen will.

Es gibt eine ganze reihe mit diesem typ, die wichtigsten werden nachstehend vorgestellt..

<meta name="**description**" content="***beschreibung***">

beschreibung ist eine schlagwortartige kurzbeschreibung der seite, die von suchmaschinen ausgewertet wird (oder auch nicht, darüber wird gestritten)

<meta name="**abstract**" content="***beschreibung***">

Entspricht der vorhergehenden anweisung, manche empfehlen beide mit gleichlautender ***beschreibung*** anzugeben.

<meta name="**keywords**" content="***schlagwort* [**, ***schlagwor***t, ...]">

Hier gibt man möglichst prägnante schlagworte über den inhalt der seite an. Die anweisung kann mehrfach vorkommen und wird von suchmaschinen ausgewertet.

<meta name="**author**" content="***autor der seite***"> selbsterklärend

<meta name="**date**" content="***datum***">

Das datum gibt man an in der form YYYY-MM-DD also jahr-monat-tag. Und wenn man sich vertut ? Der browser schluckt das.

<meta name="**robots**" content="*befehl* [, *befehl* , . . .] ">

Mit *befehl* wird dem browser mitgeteilt, wie er sich gegenüber suchmaschinen verhalten soll. Es gibt folgende befehle:

index | noindex	suchmaschinen dürfen die seite durchsuchen | dürfen die seite nicht durchsuchen
follow | nofollow	suchmaschinen dürfen bei der suche links auf der seite weiterverfolgen, bzw sie dürfen das nicht
noarchive	suchmaschinen dürfen die seite nicht kopieren und speichern. Das sollte man angeben, weil man damit verhindert, das längst überholte fassungen der seite in alle ewigkeit irgendwo noch verfügbar sind.
all	suchmaschinen können machen, was sie wollen.

meta http-equiv
Von diesem format gibt es drei anweisunge, die ggf. eine bedeutung haben.:

<meta http-equiv="**Content-Language**" content="**de**">

Damit sagt man suchmaschinen, in welcher sprache die seite geschrieben ist, hier also in deutsch.

<meta http-equiv="**Content-Type**" content="**text/html**; charset="iso-8859-1 **| utf-8**">

Damit sagt man dem browser, dass er eine textdatei erhält und mit welchem zeichensatz diese verfasst ist. Die anweisung ist spezifisch für **HTML4**, sie gilt aber auch bei HTML5, dort verwendet man aber besser die bei 2.2.1 gezeigte anweisung.

<meta http-equiv="**refresh**" content="**nn,** url=***adresse***">

adresse	beim aufruf der seite wird an diese adresse weitergeleitet.
nn	die weiterleitung erfolgt nach ***nn*** sekunden

Die adresse schreibt man in der form: http://www.**neue-seite.de/index.htm**, d.h. es wird an die adresse **neue-seite.de** weitergeleitet und dort die seite **index.htm** aufgerufen.

2.3 dateien einbinden – scripten einfügen

Der header ist der übliche ort, an dem dateien eingebunden oder scripten eingefügt werden.

CSS-datei einbinden
<link rel="stylesheet" type="text/css" href="***name***.css">

CSS-script einfügen
<style type="text/css">
CSS-anweisungen
</style>

Javascript-datei einbinden
<script type="text/javascript" [language="javascript"] src="***name***.js">
</script>

Javascript einfügen
<script type="text/javascript" [language="javascript"] >
Javascript-anweisungen
</script>

Das attribut language gilt in beiden fällen als veraltet und kann daher entfallen

PHP-script einfügen
<? php
PHP-anweisungen
?>

beispiel

Hier folgt ein muster für eine typische seite der homepage des verfassers.

```
<!DOCTYPE html>
<html>
<head>
<meta charset="utf-8">
<title>seite.php - standardseite</title>
<meta name="description" content="privater internetauftritt">
<meta name="abstract" content="privater internetauftritt">
<meta name="keywords" content="sokoban, schiffe versenken,
      verhextes puzzle, pipe, puzzle">
<meta name="author" content="Bernhard Hartard">
<meta name="date" content="2017-03-03">
<meta name="robots" content="nofollow, noarchive">
<meta http-equiv="Content-Language" content="de">

<link rel="stylesheet" type="text/css" href="hartard.css">
<script type="text/javascript" src="hartard.js">
</script>

</head>
<body>
<!--  hier folgen die anweisungen zur beschreibung der seite -->
. . .
</body>
</html>
```

2.4 bemerkungen

Überall wo es notwendig ist, kann man auf einer seite bemerkungen einfügen. Den browser interessieren sie nicht, der benutzer bekommt sie nicht zu sehen, aber dem unglücklichen, der eine seite ändern soll, die ein anderer verbrochen hat, können sie etwas helfen.

einzeilige bemerkung

```
<!-- irgend ein text -->
```

die beiden bindestriche stehen direkt nebeneinander !

mehrzeilige bemerkung

```
<!-- ein text der sich
  über mehrere zeilen
  erstreckt -->
```

2.5 zeichensätze

Es ist zweckmäßig, im header jeder seite einen zeichensatz zu vereinbaren (vgl. 2.2.1), weil oft nicht klar ist, welchen zeichensatz der browser ohne diese vereinbarung verwendet. Der browser erwartet dann, dass die seite und ggf. eingabe-dateien in dem vereinbarten zeichensatz codiert sind. Der vereinbarte zeichensatz wird auch verwendet, um dateiausgaben zu codieren. Wie die daten von formulareingaben weitergegeben werden ist einigermaßen unklar (siehe 9.1). Besondere bedeutung haben folgende zeichen-sätze:

Ansicode - ISO-8859-1
Alle zeichen werden mit einem byte codiert, dadurch ist der zeichensatz auf 256 zeichen beschränkt. In der praxis sind es aber weniger, weil die codierungen 0 – 31 für steuerzeichen reserviert sind, der code umfaßt aber alle mit einer üblichen "deutschen" tastatur eingebbaren zeichen.

Unicode - UTF-8
Die zeichen werden mit einem oder zwei bytes codiert, dadurch ist der verfügbare zeichenvorrat erheblich größer. Unicode gilt zu recht als moderner und sollte bevorzugt verwendet werden.

2.5.1 gruppe umlaute

Bezüglich der codierung des alphabets und der gängigen sonderzeichen gleichen sich die zeichensätze **Ansicode** und **Unicode** bis auf eine gravierende ausnahme: die hier als **gruppe umlaute** bezeichneten zeichen (geschlossenen umlaute und die sonderzeichen ß, €, ³, °, ², ³ µ) werden im Unicode mit **zwei** bytes codiert. Das führt zu problemen, wenn der zeichensatz, mit dem die HTML-datei erstellt wurde nicht mit dem zeichensatz übereinstimmt, der im header der seite mit **charset** vereinbart ist. Besonderheiten bei der verarbeitung von formularen werden unter ziffer 9.1 behandelt, probleme bei der verarbeitung von dateien werden in der **PHP**-beschreibung behandelt und in der **MySQL**-beschreibung finden sich hinweise zu den dort auftretenden schwierigkeiten.

2.5.2 zeichen maskieren

Probleme gibt es auch mit den sog. kritischen zeichen, das sind zeichen, die im HTML-code eine besondere bedeutung haben (", ', &, <, >). Diese zeichen müssen maskiert werden (siehe 2.5.3). Auch das leerzeichen wird oft maskiert, weil mehrere aufeinanderfolgende leerzeichen unbarmherzig auf eines gekürzt werden (das ist eine der gemeinheiten von HTML). Maskiert bleiben aber alle leerzeichen erhalten, man bezeichnet sie daher als **geschützte** leerzeichen.

2.5.3 Codes, Maskierung

zeichen	*bedeutung*	*ansi*	*maskierung*	*utf-8*	*hinweis*
"	anführ.zeichen	22	&quot;	22	müssen maskiert werden
&	kaufm. und	26	&amp;	26	
'	apostroph	27	&apos;	27	
<	kleiner	3C	&lt;	3C	
>	größer	3E	&gt;	3C	
	leerzeichen	20		20	geschütztes leerzeichen
\	backslash	5C	&#0092	5C	wird meist maskiert
Ä	umlaut	C4	&Auml;	C384	gruppe **umlaute**
Ö	umlaut	D6	&Ouml;	C396	
Ü	umlaut	DC	&Uuml;	C395	
ä	umlaut	E4	&auml;	C3A4	
ö	umlaut	F6	&ouml;	C3B6	
ü	umlaut	FC	&uuml;	C3BC	
ß	scharfes s	DF	&szlig;	C39F	
€	euro	80	&euro;	E282AC	**drei bytes !!**
§	paragraph	A7	&#0167	C2A7	
°	hochgest. 0	B0	&#0176	C2B0	
²	hochgest. 2	B2	&#0178	C2B2	
³	hochgest. 3	B3	&#0179	C2B3	
µ	my	B5	&#0181	C2B5	

Nach dem muster **&#nnnn** kann man jedes zeichen mit seiner positionsnummer aus der code-tabelle des Ansicodes maskieren.

hinweis
Bei der neuerstellung einer anwendung oder homepage sollte man sich für **einen** zeichensatz entscheiden, alle seiten mit diesem zeichensatz erstellen und diesen für den browser vereinbaren und dann dabei bleiben. Ein späterer wechsel des zeichensatzes kann sehr aufwendig werden.

2.5.4 beispiel

Es sind inhaltlich völlig gleichartige seiten definiert, die jeweils drei zeilen ausgeben:
- eine zeile mit den geschlossenen umlauten und dem zeichen ß
- eine zeile mit den gleichen zeichen, aber maskiert
- eine zeile mit den maskierten kritischen zeichen " € § & ' < >

```
<p>umlaute: <b>Ä Ö Ü ä ö ü ß</b></p>
<p>maskierte umlaute: <b>&Auml; &Ouml; &Uuml;
   &auml; &ouml; &uuml; &szlig;</b></p>
<p>kritische zeichen maskiert: <b>&quot; &euro; &#0167
   &amp; &apos; &lt; &gt;</b></p>
```

Test1
Die seite ist mit dem zeichensatz **Ansicode** erstellt und mit **charset** ist dieser zeichensatz vereinbart.

Test2
Die seite ist mit dem zeichensatz **Unicode** erstellt mit **charset** ist dieser zeichensatz vereinbart

Es gibt keine probleme, das ergebnis ist in beiden fällen identisch

umlaute: **Ä Ö Ü ä ö ü ß**

maskierte umlaute: **Ä Ö Ü ä ö ü ß**

kritische zeichen maskiert: **" € § & ' < >**

Test3
Die seite ist mit dem zeichensatz **Ansicode** erstellt, mit **charse**t ist aber **Unicode** vereinbart und es gibt probleme bei der darsellung der zeichen.

umlaute: **□ □ □ □ □ □ □**

maskierte umlaute: **Ä Ö Ü ä ö ü ß**

kritische zeichen maskiert: **" € § & ' < >**

Test4
Die seite ist mit dem zeichensatz **Unicode** erstellt, mit **charset** ist aber **Ansicode** vereinbart. Erstaunlicherweise gibt es keine probleme bei der darstellung der zeichen, das ergebnis gleicht dem von Test1 und Test2.

Fazit
Erstens man darf nicht unterschiedliche zeichensätze verwenden und zweitens mit maskierten zeichen hat man keine probleme, allerding ist das umständlich.

2.6 body-tag

Mit diesem tag beginnt die beschreibung einer seite (siehe nr. 2.1). Es ist üblich und zweckmäßig, im body-tag vereinbarungen zu treffen, die für die ganze seite gelten, soweit nicht für einzelne tags andere vereinbarungen getroffen werden. Auch im body-tag verwendet man das style- und/oder das class-attribut (siehe nr. 1.5.4).

```
<body style="background-color: aqua; color: black;">
```

```
<body class="std">
```

Im ersten fall wird die hintergrund- und die textfarbe vereinbart, im zweiten fall gelten die vereinbarungen der klasse **std**.

Manchmal ist es sinnvoll, folgende, an sich veraltete attribute zu verwenden:

```
<body link="red" vlink="gray" alink="yellow">
```

Hier werden die farben für links (siehe nr. 8.) vereinbart

link	noch nicht besuchter link
vlink	bereits besuchter (visited) link
alink	aktuell angeklickter (activated) link.

An sich kann man auch diese vereinbarungen mit **CSS** treffen, das ist aber etwas aufwendig und deshalb sind die veralteten attribute immer noch brauchbar.

2.7 fokus setzen

2.7.1 tabindex

Ein HTML-element, das einen **link** (siehe nr. 8.) oder ein **eingabefeld** eines formulars (siehe nr. 9.) enthält, wird aktiviert, indem man es anklickt. Eine andere möglichkeit ist, mit hilfe der tabulatortaste den fokus auf das element zu setzen. Dabei setzt der tabulator den fokus standardmäßig auf die elemente in der reihenfolge, wie sie in der seite angeordnet sind. Diese reihenfolge kann man aber mit dem attribut **tabindex** beeinflussen.

tabindex="n" n ist eine ganzzahl, mit der die reihenfolge bestimmt wird; üblicherweise beginnt man mit 1.

Das attribut kann in jedem HTML-element verwendet werden, außer bei den o.g. elementen bringt es aber nicht viel, d.h. es wird zwar sichtbar, dass der fokus auf ein element gesetzt wird, mehr geschieht aber nicht.

2.7.2 autofocus

Bei einem formular (siehe nr. 9.) kann man bei einem eingabefeld mit dem schlüsselwort **autofocus** als attribut festlegen, dass das feld beim laden der seite sofort den fokus erhält. Das ist in einer seite nur bei einem eingabefeld möglich.

3. abschnitte

Ein abschnitt enthält abhängig vom typ ein oder mehrere elemente; bei der anzeige kann er sich über mehrere zeilen erstrecken. Vor und nach der anzeige eines abschnitts erfolgt ein zeilenwechsel.

3.1 h-tag - überschrift

Eine überschrift ist der einfachste fall eines abschnitts; eine überschrift kann sich zwar über mehrere zeilen erstrecken, das ist aber eher unüblich.

<h***n***> *text* </h***n***>

Für ***n*** ist eine ziffer von 1 bis 6 anzugeben; damit wird die schriftgröße der überschrift festgelegt. Es ist zwar zulässig, aber schachteln sollte man überschriften nicht, das ergebnis ist unbefriedigend.

beispiele

```
<h1>überschrift stärke 1</h1>
<h2>überschrift stärke 2</h2>
<h3>überschrift stärke 3</h3>
<h4>überschrift stärke 4</h4>
<h5>überschrift stärke 5</h5>
<h6>überschrift stärke 6</h6>
```

3.2 p-tag - absatz

Ein absatz enthält in der regel einen fortlaufenden text, der sich über mehrere zeilen erstrecken kann, dabei wird der zeilenumbruch vom browser vorgenommen. Um die gezeigten effekte zu erzielen, erfolgt die ausgabe der beispiewle teilweise in div-containern (siehe 3.5).

3.2.1 format

** ** *beliebiger text* ** **

Absätze kann man schachteln, dabei gilt die formatierung des übergeordneten absatzes auch für den geschachtelten, sofern für den nicht eine andere formatierung definiert ist.

beispiele

```
<p>beispiel 1 ist ein ganz normaler absatz, der mit ziemlich sinnlosem
   inhalt gefüllt ist, damit gezeigt werden kann, wie der browser das mit
   dem zeilenumbruch schafft. Also ab sofort nur noch unsinn: aaaa bbbb
   cccc dddd eeee</p>

<p>beispiel 2 ist ein absatz, in den etwas geschachtel ist.
   <p>hier ist der geschachtelte teil</p> und hier geht der
   übergeordnete absatz weiter, sieht etwas komisch aus.</p>
```

beispiel 1 ist ein ganz normaler absatz, der mit ziemlich sinnlosem inhalt gefüllt ist, damit gezeigt werden kann, wie der browser das mit dem zeilenumbruch schafft. Also ab sofort nur noch unsinn: aaaa bbbb cccc dddd eeee

beispiel 2 ist ein absatz, in den etwas geschachtel ist.

Hier ist der geschachtelte teil

und hier geht der übergeordnete absatz weiter, sieht etwas komisch aus.

3.2.2 zeilenumbruch

Die beiden vorstehenden beispiele zeigen deutlich, dass der browser beim zeilenumbruch überhaupt nicht berücksichtigt, mit welchem zeilenumbruch ein absatz definiert wurde. Aber man kann das beeinflussen.

br-tag - zeilenumbruch erzwingen

 mit diesem tag wird ein zeilenumbruch erzwungen; das tag hat kein ende-tag.

beispiel

```
<p>beispiel 3 ist auch ein normaler absatz, aber hier wird<br /> ein
   zeilenumbruch erzwungen und weil das so schön ist <br />gleich nochmal</p>
```

beispiel 3 ist auch ein normaler absatz, aber hier wird
ein zeilenumbruch erzwungen und weil das so schön ist
gleich nochmal

zeilenumbruch verhindern

Beim automatischen zeilenumbruch durch den browser sind leerzeichen im text genau die stellen, an denen umgebrochen wird. Mit einem geschützen lehrzeichen ** ** verhindert man, dass umgebrochen wird, der browser sucht sich dann eben eine andere stelle.

beispiel

```
<p>beispiel 4 ist auch ein normaler absatz, in dem nun gezeigt wird, wie man
   einen umbruch verhindert, z.b. Bernhard Hartard, also vorname und name
   trennt man doch nicht, das ist doch nicht schön. </p>
```

beispiel 4 ist auch ein normaler absatz, in dem nun gezeigt wird, wie man einen umbruch verhindert, z.b. Bernhard Hartard, also vorname und name trennt man doch nicht, das ist doch nicht schön.

3.2.3 leerzeichen behandeln

Alle browser mißhandeln leerzeichen in gleicher weise: wo mehr als eines steht, werden sie bis auf eines unterdrückt. Das führt oft zu unschönen ergebnissen. Abhilfe schafft hier das geschützte leerzeichen, das seinen namen deshalb hat, weil es vom browser **nie** unterdrückt wird.

beispiel

```
<p>diese lücke     wird unterdrückt</p>
<p>diese lücke     bleibt erhalten</p>
```

diese lücke wird unterdrückt

diese lücke bleibt erhalten

3.3 absatz formatieren, ausrichten

3.3.1 formatieren

Das formatieren hat bei einem absatz, aber auch bei anderen elementen einer seite eine große bedeutung. Ursprünglich gab es dazu ein eigenes tag, das **font-tag**, das aber inzwischen als völlig veraltet gilt und daher in dieser unterlage nicht verwendet wird. In der praxis wird das attribut durch **CSS** (style- oder class-attribut) ersetzt. . Da CSS nicht bestandteil dieser unterlage ist, werden diese attribute nur ausnamsweise verwendet.

beispiel

Ohne weitere erklärung wird hier gezeigt, wie mit hilfe von CSS ein absatz formatiert wird. Mit dem **p-tag** wird eine zeile mit schriftart Arial, 10 punkt, fett und rot ausgegeben:

- einmal mit hilfe des **style**-attributs
- einmal mit der für diese dokumentation vereinbarten klasse **stdrot,** in der das gleiche vereinbart ist, wie in dem style-attribut.

```
<p style="font-family: Arial; font-size: 10pt; font-weight: bold; color: red;">
      schriftart Arial fett und rot mit style-attribut</p>
<p class="stdrot"> schriftart Arial fett und rot – mit class-attribut</p>
```

so sieht das ergebnis aus:

schriftart Arial fett und rot - mit style-attribut

schriftart Arial fett und rot - mit class-attribut

3.3.2 ausrichten

Ein absatz kann mit dem attribut **align** rechtsbündig oder zentriert ausgegeben werden. Das attribut ist veraltet und wird meist durch **CSS** ersetzt, zum beispiel durch eine vereinbarung im **style**-attribut.

beliebiger text	veraltetes attribut
beliebiger text	vereinbarung in style-attribut

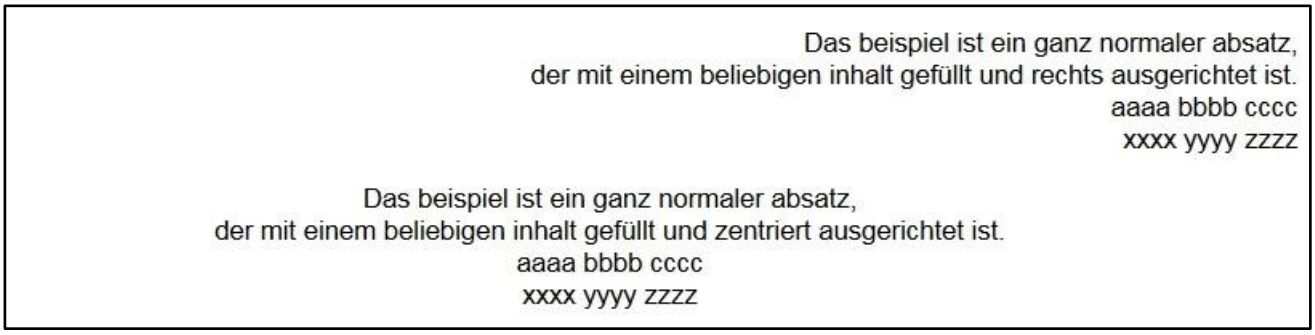
Für ***richt*** sind folgende angaben möglich:

right	rechtsbündig
center	zentriert
left	linksbündig, überflüssig, weil standard
justify	blocksatz, funktioniert meist nicht.

beispiel

In dem folgenden beispiel enthält ein absatz mehrere zeilenumbrüche und wird einmal rechtsbündig und einmal zentriert ausgegeben.

```
<p style="text-align: right;">Das beispiel ist ein ganz normaler absatz,<br />
   der mit einem beliebigen inhalt gefüllt und rechts ausgerichtet ist.<br />
   aaaa bbbb cccc<br />
   xxxx yyyy zzzz</p>
<p style="text-align: center;">Das beispiel ist ein ganz normaler absatz,<br />
   der mit einem beliebigen inhalt gefüllt und zentriert ausgerichtet ist.<br />
   aaaa bbbb cccc<br />
   xxxx yyyy zzzz</p>
```



3.4 span-tag - einschub in absatz

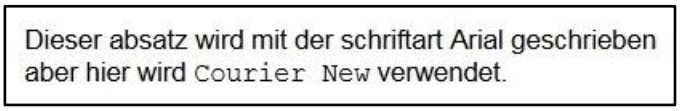
 text

Ohne dass der textfluss eines absatzes unterbrochen wird, kann man mit diesem tag etwas in einen absatz einschieben. Das wird gerne verwendet, wenn ein absatz in einer bestimmten weise formatiert ist, aber ein stück des absatzes durch eine andere formatierung hervorgehoben werden soll.

beispiel

Bei diesem beispiel erfolgt die formatierung der schriftarten **Arial** und **Courier New** mit den CSS-klassen **std** und **cour10**. Ohne CSS kann man mit diesem tag nicht allzuviel wirkungsvolles bewirken.

```
<p class="std">Dieser absatz wird mit der schriftart Arial geschrieben<br />
   aber hier wird <span class="cour10">Courier New</span> verwendet.</p>
```



3.5 div-tag - bereich / container

Mit dem **div-tag** wird ein bereich oder container definiert, in den man alle möglichen tags schachteln kann, mit der wirkung, dass eine formatierung des div-tags auch für alles, was hineingeschachtelt wird, gilt. Die besonderheit des tags ist, dass man es mit seinem gesamten inhalt wie ein einziges element behandlen kann. Das wird bei der gestaltung einer seite wichtig, sobald man mit hilfe von CSS elemente mit rahmen, hintergrundfarbe, schatten u.ä. versieht oder auf der seite zielgenau positioniert. Diese möglichkeiten werden in der CSS-dokumentation behandelt.

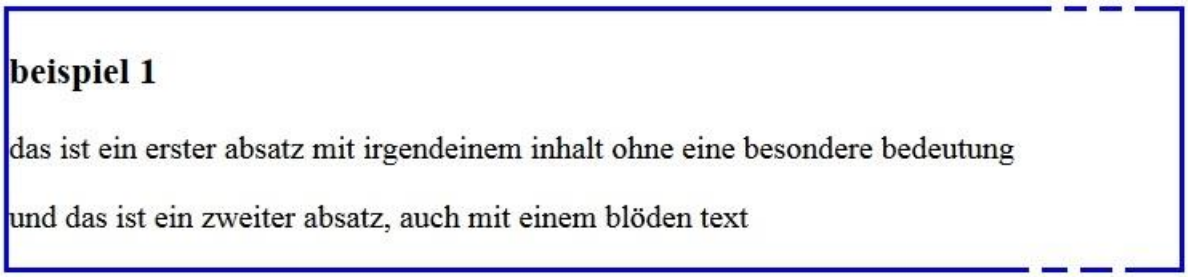
```
<div>
  alle möglichen tags
</div>
```

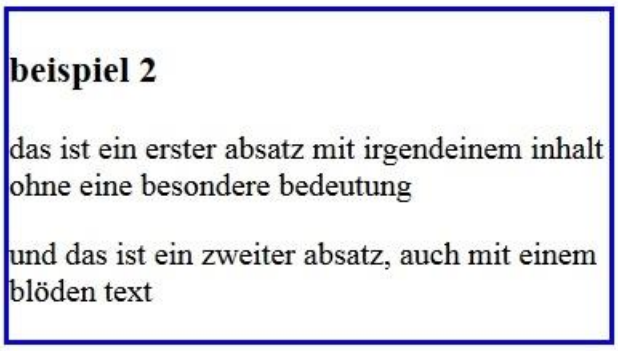
beispiele

Um die wirkung des div-tags zu verdeutlichen, wird mit hilfe einer CSS-vereinbarung (style-attribut) um jeden container ein blauer rahmen gezeichnet. Der zweite container wird zudem auf eine breite von 300 pixel beschränkt.

```
div style="border: 3px solid blue;">
   <h3>beispiel 1</h3>
   <p>das ist ein erster absatz mit irgendeinem inhalt ohne eine besondere
      bedeutung</p>
   <p>und das ist ein zweiter absatz, auch mit einem blöden text</p>
</div>

<div style="width: 300px; border: 3px solid blue;">
   <h3>beispiel 2</h3>
   <p>das ist ein erster absatz mit irgendeinem inhalt ohne eine besondere
      bedeutung</p>
   <p>und das ist ein zweiter absatz, auch mit einem blöden text</p>
</div>
```





3.6 logischer abschnitt

Mit **HTML5** wurden einige tags eingeführt, mit denen abschnitte deklariert werden können, durch die eine seite deutlicher struktutiert wird. In diesen abschnitten werden elemente zusammengefaßt, die logisch zusammen gehören, deshalb werden diese abschnitte hier als **logische abschnitte** bezeichnet. Bei der anzeige einer seite sind logische abschnitte zunächst nicht erkennbar, aber man kann mit CSS für diese abschnitte vereinbarungen treffen über schriftart, schriftfarbe, breite usw. und damit das aussehen des abschnitts bei der anzeige der seite einheitlich und von anderen abschnitten unterscheidbar gestalten.

header-tag - kopfbereich

Damit wird ein abschnitt als erster abschnitt der seite deklariert; man sollte ihn dann auch an den anfang stellen und auch nur einen abschnitt als kopfbereich deklarieren.

Man darf den kopfabschnitt nicht mit dem header der seite verwechseln, der mit dem **head**-tag deklariert wird.

footer-tag - fußbereich

Damit wird ein abschnitt als letzter abschnitt der seite deklariert und sollte daher am ende der seite stehen und nur einmal vorkommen.

section-tag - bereich

Damit deklariert man einen beliebigen teil der seite als eine logische einheit. Eine seite kann beliebig viele dieser bereiche enthalten.

beispiel

Das beispiel zeigt eine durch logische abschnitte gegliederte seite. Um diese gliederung sichtbar zu machen wurden die CSS-klassen **std** (schriftart Arial, 10 punkt) und **blaufett** (schriftart Arial 12 punkt, blau, fettschrift) definiert. Der kopf- und der fuß-bereich werden mit der klasse **blaufett** und der zweite bereich mit der klasse **std** formatiert. Im fußbereich wird gezeigt, dass man bei der formatierung von elementen auch von der formatierung des abschnitts abweichen kann.

```
<body>
<header class="blaufett">
   <h2>kopfbereich</h2>
   <p>das steht im kopfbereich</p>
</header
<section>
   <h3>abschnitt 1</h3>
   <p>das steht im ersten abschnitt</p>
</section>
<section class="std">
   <h3>abschnitt 2</h3>
   <p>das steht im zweiten abschnitt</p>
</section>

<footer class="blaufett">
   <h3>fußbereich</h3>
   <p>das steht im fussbereich</p>
   <p class="std">andere formatierung </p>
  </footer>
</body>
```

4. tags zur textgestaltung

4.1 text hervorheben

Man kann, wie gezeigt, innerhalb eines absatzes text mit hilfe des **span-tags** durch eine besondere formatierung hervorheben. Das ist aber etwas umständlich und in vielen fällen genügt eine hervorhebung mit einem der folgenden tags, die man auch kombinieren kann.

text	fettschrift
text	fettschrift, kein unterschied zum b-tag
text	kursiv
text	hervorgehoben, meist kursiv, hängt vom browser ab
<mark> text </mark>	text mit hintergrundfarbe, meist gelb; zwischenraum-zeichen vor oder nach dem text werden nicht hervorgehoben, außer geschützte leerzeichen (ab HTML5).
text	text verkleinern (ab HTML5)

beispiele

```
<p>durch <b>fettschrift</b> hervorgehoben</p>
<p>durch <strong>fettschrift</strong> hervorgehoben</p>
<p>durch <i>kursive schrift</i> hervorgehoben</p>
<p>durch <em>irgend etwas</em> hervorgehoben</p>
<p>durch <i><b>kursive fettschrift</b></i> hervorgehoben</p>
<p>durch <mark>    hintergrund farbe     </mark> hervorheben</p>
<p>durch <small>verkleinerung</small> hervorheben</p>
```

durch **fettschrift** hervorgehoben

durch **fettschrift** hervorgehoben

durch *kursive schrift* hervorgehoben

durch *irgend etwas* hervorgehoben

durch ***kursive fettschrift*** hervorgehoben

durch hintergrund farbe hervorheben

durch verkleinerung hervorheben

4.2 hoch- oder tief stellen

Die beiden tags sind wohl selbterklärend.

^{*text*} hochstellen

_{*text*} tiefstellen

beispiele

```
<p>eine potenz 2<sup>10</sup> schreibt man so</p>
<p>die formel H<sub>2</sub>O bedeutet pures wasser</p>
```

eine potenz 2^{10} schreibt man so

die formel H_2O bedeutet pures wasser

4.3 querstrich

Das ist eines der wenigen tags ohne ende-tag; es erzeugt einfach einen querstrich, den man mit attributen oder entsprechenden CSS-vereinbarungen formatieren kann.

<hr>

Nachfolgend werden drei querstriche angezeigt, einmal ohne attribut und dann mit dem style-attribut mit vereinbarungen für breite, höhe. farbe und hintergrundfarbe. Seltsamerweise kann man hier nicht mit **size** die strichdicke vereinbaren..

```
<p>hier kommen drei querstriche</p>
<hr>
<hr style="width: 300px; color: blue; height: 4px; background-color: blue;">
<hr style="width: 400px; color: #FF0000; height: 7px; background-color: #FF0000;">
```

.

hier kommen drei querstriche

4.4 pre-tag - nicht-proportionale schrift

Die üblichen schriftarten sind proportionale schriften, d.h. jedes zeichen nimmt nur so viel platz ein, wie es benötigt. Daneben gibt es einige schriftarten wie **Courier** oder **Courier New**, die nicht proportional (diktengleich) geschrieben werden. Zum vergleich:

proportionale schrift

```
nicht proportionale schrift
```

Um so etwas zu erreichen, muss man einen absatz entweder mit einer nicht proportionalen schriftart formatieren oder in das **pre-tag** einschließen. In beiden fällen werden übrigens die leerzeichen **nicht** unterdrückt.

beispiel

```
<pre>
   aaaa    bbbb    cccc
   xxxxx   yy      z
</pre>
```

```
aaaa    bbbb    cccc
xxxxx   yy      z
```

5. listen

Bei den tags zur definition einer liste handelt es sich um sog. geschachtelte tags, d.h. es müssen unterschiedliche, aber zusammengehörige tags so geschachtelt werden, dass der browser daraus eine liste bauen kann. Listen können mit attributen vielfältig formatiert werden. Weil man das in der praxis aber besser mit CSS macht, wird hier nicht näher darauf eingegangen.

5.1 ol-tag - numerierte liste / ordered list

Zur definition einer numerierten liste gehören das **ol-tag** und ein oder mehrere **li-tags** (list-idem).

<ol [type="***z***"] [start="***nn***"] [reversed] >

```
  <li> erster listeneintrag </li>
  . . .
  <li> letzter listeneintrag </li>
</ol>
```

type="***z***" mit dem attribut **type** wird das listensymbol festgelegt; ab HTML5 ist das attribut nicht mehr vorgesehen und soll durch eine entsprechende CSS-vereinbarung ersetzt werde, Das attribut wird aber weiterhin akzeptiert und in den folgenden beispielen verwendet..

keine ang.	numerierung mit 1. 2. usw.
A	numerierung mit A. B. usw.
a	numerierung mit a. b, usw
I	numerierung mit I. II. usw.
i	numerierung mit i. ii. usw.

start="***nn***" der startwert der numerierung kann mit dem attribut **start** festgelegt werden; ***nn*** ist dabei ein numerischer wert; auch bei alphabetischer oder lateinischer numerierung ist der startwert numerisch anzugeben, **5** bedeutet dann **E** (fünfter buchstabe) bzw. **V**

reversed mit diesem schlüsselwort beginnt die liste mit dem höchsten wert

beispiel

```
<ol>
  <li>erster eintrag</li>
  <li>zweiter eintrag</li>
  <li>dritter eintrag</li>
</ol>
```

1. erster eintrag
2. zweiter eintrag
3. dritter eintrag

```
<ol type="A">
  <li>erster eintrag</li>
  <li>zweiter eintrag</li>
  <li>dritter eintrag</li>
</ol>
```

A. erster eintrag
B. zweiter eintrag
C. dritter eintrag

Innerhalb der liste kann für einen listeneintrag mit dem attribut **value** ebenfalls ein startwert festgelegt werden. Seltsam, aber möglich ist es, dabei einen niedrigeren als bereits erreichten wert anzugeben (vgl. beispiel)

<li value="***nn***">

nn ist auch hier ein numerischer wert.

beispiel

```
<ol start="5">
  <li>eintrag aaaa</li>
  <li>eintrag bbbb</li>
  <li value="10">eintrag cccc</li>
  <li>eintrag xxxx</li>
  <li value="2">eintrag yyyy</li>
  <li>eintrag yyyy</li>
</ol>
```

5. eintrag aaaa
6. eintrag bbbb
10. eintrag cccc
11. eintrag xxxx
2. eintrag yyyy
3. eintrag yyyy

5.2 ul-tag - aufzählungsliste / unordered list

Bei einer aufzählungsliste werden die listeneinträge durch ein sog. listensymbol markiert. Zur definition der liste benötigt man das **ul-tag** und ein oder mehrere **li-tags**.

<ul [type="***bez***"] >
   erster eintrag
  . . .
   letzter eintrag

Mit dem attribut **type** wird das listensymbol bestimmt; ab HTML5 sollte stattdessen eine CSS-vereinbarung verwendet werden.

keine ang.	gefüllter kreis
circle	kreis
disc	gefüllter kreis
square	quadrat

beispiel

```
<ul type="disc">
  <li>eintrag aaaa</li>
  <li>eintrag bbbb</li>
  <li>eintrag cccc</li>
</ul>
```

- eintrag aaaa
- eintrag bbbb
- eintrag cccc

5.3 definitionsliste

Darunter versteht man eine liste, bei der jeweils einem begriff eine erläuterung folgt (glossar). Zur definition benötigt man das **dl-tag** und eine oder mehrere folgen des **dt**- und **dd-tags**, dabei.können einem **dt**-tag mehrere **dd**-tags folgen.

<dl>
  <dt> *erster begriff* </dt>
    <dd> *erläuterung des begriffs* </dd>
  . . .
  <dt> *letzter begriff* </dt>
    <dd> *erläuterung die begriffs* </dd>
</dl>

beispiel

```
<dl>
   <dt>EDV</dt>
       <dd>allgemein anerkanntes kürzel<br />
           für ende der vernunft</dd>
   <dt>Computer</dt>
       <dd>- ein gerät, das probleme löst, die es
           ohne das gerät gar nicht gäbe</dd>
       <dd>- lockruf für einen truthahn<br />
           (komm puter) aua</dd>
</dl>
```

EDV
    allgemein anerkanntes kürzel
    für ende der vernunft
Computer
    - ein gerät, das probleme löst, die es ohne das gerät gar nicht gäbe
    - lockruf für einen truthahn (komm puter) aua

5.4 geschachtelte listen

Man kann listen, auch mit unterschiedlichem typ, ineinander schachteln

beispiel
Hier sind drei listen geschachtelt; dass im code hier mit einrückungen gearbeitet wird, hat keinen einfluß darauf, wie der browser das darstellt.

```
<ol>
  <li>erster eintrag</li>
  <li>zweiter eintrag</li>
     <ol type="A">
       <li>eintrag A</li>
       <li>eintrag B</li>
       <li>eintrag C</li>
          <ul type="circle">
            <li>eintrag xxxx</li>
            <li>eintrag yyyy</li>
            <li>eintrag zzz</li>
          </ul>
       <li>eintrag D</li>
       <li>eintrag E</li>
     </ol>
  <li>dritter eintrag</li>
  <li>vierter eintrag</li>
</ol>
```

1. erster eintrag
2. zweiter eintrag
    A. eintrag A
    B. eintrag B
    C. eintrag C
        - eintrag xxxx
        - eintrag yyyy
        - eintrag zzz
    D. eintrag D
    E. eintrag E
3. dritter eintrag
4. vierter eintrag

6. table-tag - tabellen

6.1 definition einer tabelle

Tabellen werden mit dem **table-tag** konstruiert; dabei werden folgende tags ineinander geschachtelt:

tr-tag definiert eine tabellenzeile und enthält ein oder mehrere th- oder td-tags

th-tag definiert eine tabellenzelle mit dem typ spaltenüberschrift; üblicherweise ist das die erste zeile einer tabelle und eine tabelle enthält nur eine zeile mit spaltenüberschriften. Außerdem mischt man in einer zeile nicht zellen mit überschriften und normale zellen.

td-tag definiert eine normale tabellenzelle

Die anzahl der th- oder td-tags in einem tr-tag legt die spaltenzahl der tabelle fest, man sollte für jede zeile der tabelle die gleiche anzahl von spalten definieren, sonst liefert der browser in der regel ziemlichen unsinn.

```
<table  [ border="n" ]  [ cellspacing="nn" ]  [ cellpadding="nn" ]  >
  <tr>                                          spaltenüberschriften
      <th> text </th>
      . . .
      <th> text </th>
  </tr>
  <tr>                                          erste zeile
      <td> text </td>
      . . .
      <td> text </td>
 </tr>
  . . .                                         weitere zeilen
</table>
```

border="***nn***" ***nn*** ist ein numerischer wert, um die tabelle wird ein rahmen mit der dicke **n** pixel gezeichnet, die zellen der tabelle erhalten einen rahmen von 1 pixel. Fehlt die angabe oder wird 0 angegeben, werden kein rahmen gezeichnet.

cellspacing="***nn***" ***nn*** ist ein numerischer wert und legt den abstand der zellen untereinander fest. Fehlt die angabe liegen die zellen eng zusammen.

cellpadding="***nn***" ***nn*** ist ein numerischer wert und legt den abstand des zelleninhalts vom zellenrand fest

hinweise

- Alle attribute sind veraltet, sie werden in den beispielen nur verwendet, um zu zeigen, was man mit tabellen grundsätzlich tun kann. In der praxis verwendet man **CSS**-vereinbarungen, die noch weitaus mehr möglichkeiten der gestaltung bieten.
- Spaltenüberschriften (th-tag) werden standardmäßig zentriert und mit einer hervorgehobenen schriftart ausgegeben. Eine zeile mit spaltenüberschriften ist nicht notwendig.
- Leere zellen (<td> </td>) sind zulässig, das ergebnis ist aber meist unschön. Besser ist es für eine leere zelle ein geschütztes leerzeichen ( ) anzugeben.

beispiel
Das beispiel zeigt eine tabell einmal ohne und einmal mit rahmen.

```
<table>
  <tr>
     <th>spalte 1</th>
     <th>spalte 2</th>
     <th>spalte 3</th>
  </tr>
  <tr>
     <td>aaaa</td>
     <td>bbbbbbbbbbbbbbbb</td>
     <td>cccc</td>
  </tr>
  <tr>
     <td>aaaa</td>
     <td>bbbb</td>
     <td>ccccccccccc</td>
  </tr>
</table>
```

spalte 1	spalte 2	spalte 3
aaaa	bbbbbbbbbbbbbbbb	cccc
aaaa	bbbb	ccccccccccc

Die definition der zweiten tabelle unterscheidet sich nur durch das table-tag:
`<table border="3">`

spalte 1	spalte 2	spalte 3
aaaa	bbbbbbbbbbbbbbbb	cccc
aaaa	bbbb	ccccccccccc

6.2 tabellenbreite, spaltenbreite

Standardmäßig ergibt sich die breite einer tabelle aus der anzahl und breite der spalten, die breite einer spalte ergibt sich aus der breite der breitesten zelle in der spalte.Das ergibt aber oft sehr unansehnliche tabellen. Deshalb legt man meist im **table**-tag die breite der tabelle fest und für eine spalte in einem **td**- oder **th**-tag der spalte. Statt des veralteten **width**-attributs verwendet man besser das **style**-attribut mit einer entsprechenden vereinbarung.

<table style="width: ***nn***px;">	tabellenbreite in ***nn*** pixel
<td style="width: ***nn***px;">	spaltenbreite in ***nn*** pixel
<td style="width: ***nn***%;">	spaltenbreite in ***nn*** prozent der tabellenbreite

Wenn eine tabellenbreite festgelegt ist, sollte die summe der spaltenbreite diese nicht überschreiten. Das vermeidet man, indem man für eine spalte keine breite definiert, diese spalte erhält dann die verbleibende restbreite. Die für eine spalte festgelegte breite wird immer eingehalten, ggf. erfolgt in einer zelle ein zeilenumbruch.

beispiel
In dem beispiel werden tabellen- und spaltenbreite festgelegt. Außerdem wird der abstand der zellen voneinander und der abstand des zelleninhalts vom rand festgelegt.

```
<table border="3" cellspacing="6" cellpadding="4"
       style="width: 300px;">
  <tr>
     <th style="width: 33%;">
         spalte 1</th>
     <th style="width: 33%;">
         spalte 2</th>
     <th>spalte 3</th>
  </tr>
  <tr>
     <td>aaaaa aaaaa aaaa aaaaa</td>
     <td>bbbb</td>
     <td>cccc</td>
  </tr>
  <tr>
     <td>aaaa</td>
     <td>bbbb</td>
     <td>cccc</td>
  </tr>
</table>
```

spalte 1	spalte 2	spalte 3
aaaaa aaaaa aaaa aaaaa	bbbb	cccc
aaaa	bbbb	cccc

6.3 zellen verbinden

Man kann nebeneinander oder übereinander liegende zellen miteinander zu einer zelle verbinden.

<td colspan="***n***"> *text* </td>	***n*** ist die anzahl der **nebeneinander** liegenden zellen, die miteinander verbunden werden.
<td rowspan="***n***"> *text* </td>	***n*** ist die anzahl der **untereinander** liegenden zellen, die miteinander verbundnen werden
<td rowspan="**n**" colspan="**n**"> *text* </td>	**nebeneinander** und **untereinander** liegende zellen verbinden

Wenn man zellen miteinander verbindet, muss man das natürlich bei der definition weiterer zellen in der zeile oder spalte berücksichtigen, das ist manchmal gar nicht so einfach, das folgende beispiel versucht, das anschaulich zu machen.

beispiel
Bei dieser tabelle haben die fünf spalten alle die gleiche breite, außerdem wird gezeigt, wie spalten und zeilen zusammengefaßt werden..

```
<table border="3" style="width: 500px">
  <tr>
     <th width="20%">spalte A</th>
     <th width="20%">spalte B</th>
     <th width="20%">spalte C</th>
     <th width="20%">spalte D</th>
     <th>spalte E</th>
  </tr>
  <tr>
     <td>aaaa</td>
     <td colspan="2">2 spalten</td>
     <td>dddd</td>
     <td>eeee</td>
  </tr>
  <tr>
     <td rowspan="3">3 zeilen</td>
     <td>bbbb</td>
     <td rowspan="2" colspan="2">2 spalten 2 zeilen</td>
     <td>eeee</td>
  </tr>
  <tr>
      <td>bbbb</td>
      <td>eeee</td>
  </tr>
  <tr>
     <td>bbbb</td>
     <td>cccc</td>
     <td>dddd</td>
     <td>eeee</td>
  </tr>
</table>
```

spalte A	spalte B	spalte C	spalte D	spalte E
aaaa	2 spalten		dddd	eeee
3 zeilen	bbbb	2 spalten 2 zeilen		eeee
	bbbb			eeee
	bbbb	cccc	dddd	eeee

6.4 tabellen im HTML5-format

Mit **HTML5** wurden neue tags eingeführt, mit denen tabellen besser strukturiert werden können.

caption-tag - titelzeile

<caption> ***text*** </caption>

Damit wird für eine tabelle eine titelzeile definiert, Das tag muss dem **table**-tag unmittelbar folgen.

tags zur strukturierung

Mit diesen tags kann die struktur einer tabelle deutlicher gemacht werden. Die tags funktionieren wie die tags für logische abschnitte (siehe 3.6), d.h. solange sie keine CSS-vereinbarungen zur formatierung enthalten, machen sie den code der tabelle übersichtlicher, haben aber keine auswirkung auf die darstellung der tabelle.

thead-tag - überschriftszeile

In dieses tag werden die tags für die spaltenüberschriften eingeschlossen.

tbody-tag - hauptteil der tabelle

Das tag schließt den hauptteil der tabelle ein

tfoot-tag - fußzeile

Das tag schließt die tags für die fußzeile der tabelle ein.

beispiel

In dem beispiel werden wieder CSS-klassen zur formatierung verwendet, um die neuen tags bei der anzeige der tabelle kenntlich zu machen.

```
<table border="3" width="300">
  <caption class="blaufett">tabelle im HTML5-format</caption>
  <thead class="std">
    <tr>
       <th>spalte 1</th>
       <th>spalte 2</th>
       <th>spalte 3</th>
    </tr>
  </thead>
  <tbody>
    <tr>
       <td>aaaa</td>
       <td>bbbb</td>
       <td>cccc</td>
    </tr>
    <tr>
       <td>aaaa</td>
       <td>bbbb</td>
       <td>cccc</td>
    </tr>
  </tbody>
  <tfoot class="std">
    <tr>
       <td>fuss 1</td>
       <td>fuss 2</td>
       <td>fuss 3</td>
    </tr>
  <tfoot>
</table>
```

tabelle im HTML5-format

spalte 1	spalte 2	spalte 3
aaaa	bbbb	cccc
aaaa	bbbb	cccc
fuss 1	fuss 2	fuss 3

7. grafiken

7.1 img-tag - grafik definieren

Mit dem **img-tag** stellt man eine grafik in eine seite; das tag hat kein endetag. Es sind alle grafikformate zulässig, meist wählt man die formate gif, jpg, png oder bmp.

pfadname wenn die grafik im gleichen ordner gespeichert ist wie die seite, genügt die angabe des dateinamens, andernfalls muss man den pfadnamen der grafik angeben

unterordner / dateiname	grafik in einem unterordner
.. / .. / dateiname	grafik in einem ordner zwei stufen höher
.. / unterordner / dateiname	grafik in einem unterordner des übergeordneten ordners

text text, der angezeigt wird, wenn der browser keine grafiken anzeigen kann

hinweis wenn man die angezeigte grafik mit der schreibmarke berührt, wird dieser hinweistext als sog. quickinfo eingeblendet.

Man kann das **img-tag** unmittelbar in die seite stellen, meist aber schachtelt man es in ein anderes element (p-tag, div-tag oder wie in den folgenden beispielen in eine zelle einer tabelle) weil man so die grafik besser positionieren kann.

7.2 grafik anzeigen

Ohne weitere angaben versucht der browser eine grafik genau in der größe anzuzeigen, wie sie gespeichert ist. Das ist nicht immer optimal, besonders wenn eine grafik sehr groß ist. Oft ist es besser, eine grafik schon vor dem speichern mit einem bildbearbeitungsprogramm auf eine passende größe zu bringen. Es ist auch möglich, im **img**-tag für die breite und höhe angaben zu machen. Die dafür vorgesehenen attribute **width** und **heigh**t sind aber veraltet, man macht das besser mit dem **style**-attribut. Das wird in den folgenden beispielen gezeigt. Auf diese art kann man eine grafik in jeder beliebigen größe anzeigen, die proportionen müssen natürlich mit denen der gespeicherten grafik übereinstimmen.

beispiel 1
Es gibt zwei grafiken: **wappen1** mit den maßen 150 x 200 pixel und **wappen2** mit 75 x 100 pixel. Die grafiken werden in einer tabelle dargestellt. In der ersten zeile wird wappen1 in genau halber größe (75 x 100) und wappen2 in der originalgröße, also auch 75 x 100 angezeigt; hier ein ausschnitt aus der tabellendefintion:

```
<tr>
   <td><img src="doku/HTML/im/wappen1.jpg" style="width: 75px; height: 100px;"
       title="wappen1 verkleinert"/></td>
   <td><img src="doku/HTML/im/wappen2.jpg" title="wappen2 original/></td>
</tr>
```

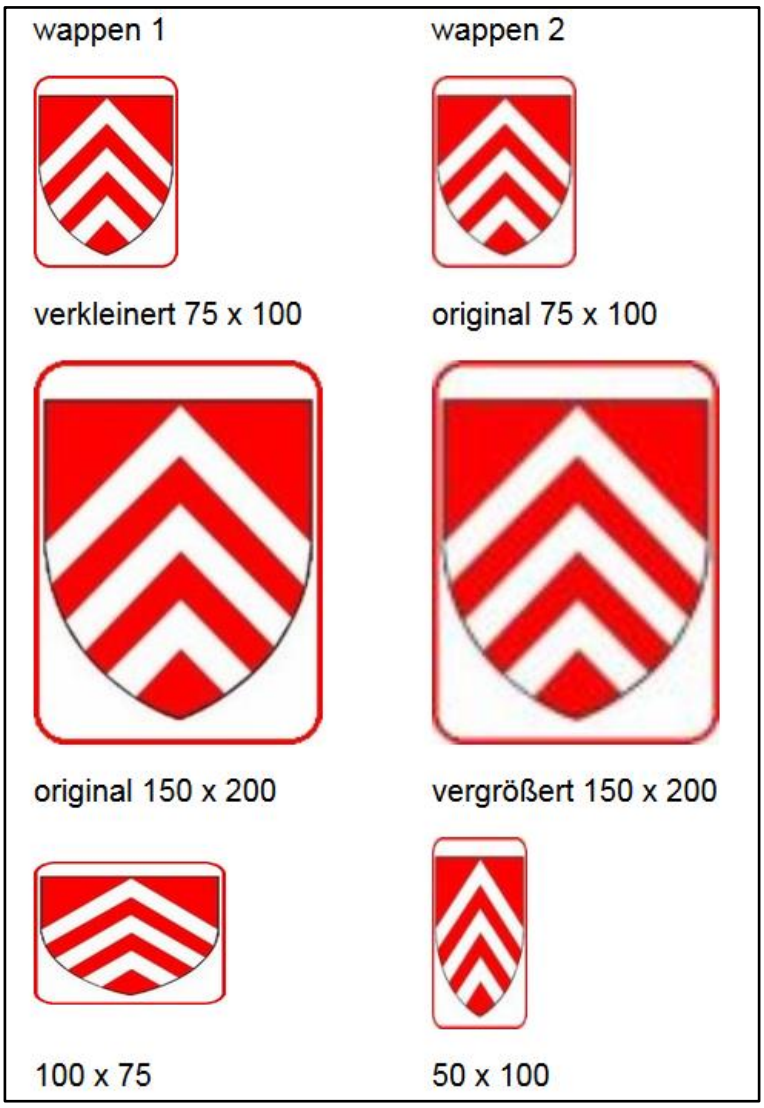
beispiel 2
Jetzt wird wappen1 in der originalgröße 150 x 200 angezeigt und wappen2 wird auf diese maße vergrößert, d.h.auf die doppelte größe. Das funktioniert, aber die bildqualität leidet etwas.

```
<tr>
   <td><img src="doku/HTML/im/wappen1.jpg" title="wappen1 original"/></td>
   <td><img src="doku/HTML/im/wappen2.jpg" style="width; 150px; height: 200px;"
       title="wappen2 vergrößert"/></td>
</tr>
```

beispiel 3
Hier wurden für breite und höhe arg willkürliche größen gewählt, die zu den proportionen der originale nicht passen.

```
<tr>
   <td><img src="doku/HTML/im/wappen1.jpg" style="width: 100px; height: 75px;"
       title="wappen1 verzerrt"/></td>
   <td><img src="doku/HTML/im/wappen2.jpg" style="width: 50px; height: 100px;"
       title="wappen2 verzerrt" /></td>
</tr>
```

zu beispiel 1 – 3
Die grafiken werden hier ohne rahmen dargestellt, der abgerundete rahmen, der hier zu sehen ist, ist bestanddteil der grafik, um die jeweilige größe zu verdeutlichen.



8. hyperlinks

8.1 a-tag - format

Allgemein gesagt stellt man mit einem **hyperlink**, ab sofort nur noch als **link** bezeichnet, die verbindung zu einer anderen datei her, in den meisten fällen wohl zu einer anderen seite, die dann vom browser angezeigt wird.

text

ziel	adresse der datei, die aufgerufen wird
text	dieser text wird entweder unterstrichen angezeigt oder meist mit **CSS** hervorgehoben dargestellt (so wird das in dieser dokumentation gemacht). Diesen text muß man anclicken, damit der link ausgeführt wird. An stelle eines textes kann hier auch eine grafik definiert werden.
hinweis	wen man mit dem cursor ***text*** berührt, tut sich normalerweise nichts, es sei denn es wurde mit **CSS** etwas gebastelt. Gibt man aber das attribut **title** an, wird ***hinweis*** als quickinfo angezeigt.
_**blank**	das ist ein schlüsselwort; was es bewirkt s.u. gleich im anschluss.
tabindex="***n***"	***n*** gibt die reihenfolge an, in der das element mit dem tabulator den fokus erhält. siehe nr. 2.7 fokus setzen

rückkehr zur aufrufenden seite

Normalerweise (also ohne target="_blank") wird die aufgerufene seite im gleichen fenster gezeigt, wie die aufrufende, man erkennt das an dem reiter ganz oben, in dem jetzt der name der gerufenen seite steht. Will man zurück zur aufrufenden seite, klickt man links oben den nach links zeigenden pfeil an, nur ja nicht den reiter, damit schließt man das fenster und man ist raus aus dem spiel. Oft wird die rückkehr auch mit Javascript realisiert, das wird aber in dieser unterlage nicht erklärt..

Wurde aber target="_blank" angegeben, wird die gerufene seite in einem neuen fenster gezeigt, d.h. die rufende seite ist weiterhin vorhanden. Ganz folgerichtig wird nun auch ein weiterer reiter mit dem namen der gerufenen seite angezeigt. Die rückkehr zur rufenden seite ist nun nicht über den o.g. pfeil möglich, man muss vielmehr die gerufene seite über den reiter schließen.

a-tag verwenden

```
<a src="testseite.htm"> testseite</a>
<a src="testseite.htm"><img src="grafik.gif" /></a>
```

Man kann das a-tag, wie hier gezeigt, einfach in eine seite stellen, dann wird bei der anzeige der seite der link so erscheinen

testseite	mit standardformatierung
testseite	formatierung mit CSS
	link mit grafik verknüpft

Der benutzer kann dann raten, was das bedeutet. Es ist dann doch besser, das a-tag in ein anders tag (p-tag, div-tag oder td-tag in einer tabelle) zu schachteln, weil man jetzt das ganze mit einem passenden text verknüpfen und einigermaßen selbsterklärend gestalten kann.

```
<p><a src="testseite.htm">hier klicken</a>   zum aufruf der testseite</p>
<p><a src="testseite.htm"><img src="grafik.gif" /></a>
         hier klicken zum aufruf der testseite</p>
```

Jetzt wird auf der seite erscheinen

hier klicken zum aufruf der testseite	mit standardformatierung
hier klicken zum aufruf der testseiten	formatierung mit CSS
 hier klicken zum aufruf der testseite	link mit grafik verknüpft

8.2 eine seite als ziel

Je nachdem, wo die aufgerufene seite gespeichert ist, gibt es verschiedene möglichkeiten, das ziel anzugeben:

dateiname	die seite befindet sich im gleichen ordner wie die rufende seite
ordner / dateiname	die seite befindet sich in einem unterordner
.. / dateiname	die seite befindet sich in einem übergeordneten ordner
.. / unterordner / dateiname	die seite befindet sich in einem unterordner des übergeordneten ordners
http://www.adresse / pfadname	die seite befindet sich irgendwo in den weiten des Internet.

beispiele

Das erste beispiel ist ein ganz normaler link mit dem attribut **title**. Im zweiten beispiel wird das attribut **target** verwendet und beim dritten beispiel ist der link mit einer grafik verknüpft, außerdem wird auch hier das attribut **title** verwendet. Man könnte dieses attribut auch in das img-tag stellen, aber bitte nicht in beide tags.

Dass der dateiname der aufgerufenen seite das suffix **.php** hat und durch die seltsame angabe (?art= …) nach dem pfadnamen sollte man sich nicht verwirren lassen, das ist nur wichtig für die aufgerufene seite und hat nichts mit **HTML** sondern mit **PHP** zu tun.

```
<p><a href="doku/HTML/html-test.php?art=1" title="testseite aufrufen">
   hier klicken</a>  zur testseite</p>
<p><a href="doku/HTML/html-test.php?art=2" target="_blank">
    hier klicken</a>  zur testseite</p>
<p><a href="doku/HTML/html-test.php?art=3" title="testseite aufrufen">
   <img src="doku/HTML/im/n-links.gif" /></a>  hier klicken
   zum aufruf der testseite</p>
```

hier klicken zur testseite

hier klicken zur testseite


hier klicken zum aufruf der testseite

8.3 andere ziele

Man kann mit einem link nicht nur eine WEB-seite zur anzeige bringen, sondern beliebige dateien sofern der browser zugriff zu einem programm hat, das die datei anzeigen kann. Die angezeigte datei kann man dann ausdrucken oder speichern. Praktische bedeutung hat das für text-dateien, PDF-dateien und grafiken. Zurück kommt man wieder über den pfeil links oben. Eine WORD-datei wird nicht angezeigt, sondern nur zum speichern angeboten und zurück kommt man mit ***Abbrechen***. (So ist das bei **Word365**, bei **Word** möglicherweise anders).

beispiele

Bei den folgenden beispielen werden eine reine textdatei, eine WORD-datei, eine PDF-datei und eine grafik aufgerufen..

```
<p><a href="doku/HTML/html-hilf.txt">text-datei</a>
      hier klicken</p>
<p><a href="doku/HTML/html-hilf.docx">
    WORD-datei</a>  hier klicken</p>
<p><a href="doku/HTML/html-hilf.pdf">PDF-datei</a>
      hier klicken</p>
<p><a href="doku/HTML/im/wappen1.jpg">grafik</a>
     hier klicken</p>
```

text-datei hier klicken

WORD-datei hier klicken

PDF-datei hier klicken

grafik hier klicken

8.4 anker als ziel

Man kann in einer seite mit dem **a-tag** sog. anker setzen und dann einen link auf einen solchen anker einrichten. Es ist auch möglich zu einem anker zu verlinken, der sich auf einer anderen seite befindet.

anker setzen

 [***text***]

anker das ist eine beliebige alphanumerische bezeichnung, die auf der seite aber nicht sichtbar ist

text dieser text ist sichtbar, aber nicht notwendig, d.h. es muß auf der seite gar nicht erkennbar sein, wo ein anker gesetzt ist.

anker verlinken

text link zu einem anker auf der gleichen seite

text link zu einem anker auf der seite *dateiname*

#anker name eines ankers, auf das nummernzeichen sollte man achten, es ist zwingend notwendig

text das ist wieder der text, auf den man klicken muss.

beispiel
Am beginn dieser seite ist ein anker gesetzt und mit dem link kann man zum anfang springen.

```
<p><a name="anfang"></a>das ist ein anker</p>
<p><a href="#anfang">zum seitenanfang</a></p>
```

8.5 map-tag - image-map

Das **map-tag** ist ein tag, in das **area-tags** geschachtelt werden. Mit einem **area-tag** definiert man in einer grafik einen sog. **hotspot**, das ist eine teilflächen der grafik, die mit einem **link** verknüpft ist. Das area-tag hat kein ende-tag. Damit das ganze funktioniert, muss die image-map der grafik verbunden werden.

definition der image-map

<map name="***mapname***" [id="***mapname***"] >
  <area shape="***typ***" coords="***koordinaten***" href="***pfadname***" [title="***hinweis***"] >
  weitere area-tags
</map>

mapname das **map-tag** muss mit dem attribut **name** einen eindeutigen namen erhalten. Das attribut **id** ist ggf. für CSS notwendig, dabei kann man den gleichen namen verwenden wie im attribut **name**.

typ es gibt drei typen des hotspots
- rect rechteck
- circle kreis
- poly vieleck

koordinaten koordinaten des hotspots innerhalb der grafik; die angaben hängen vom typ ab.

hinweis wenn man mit dem cursor einen hotspot berührt, wird hinweis als quickinfo angezeigt.

image-map und grafik verbinden

pfadname pfadname der datei, die die grafik enthält

#mapname name, der im **map-tag** festgelegt wurde, hier aber mit dem vorangestellten #

angabe der koordinaten

Die koordinaten eines hotspots werden von der linken oberen ecke der grafik (das ist 0, 0) in pixel gemessen und als zahlenpaar angegeben, als erstes der abstand horizontal, als zweites der abstand vertikal. Mit hilfe eines bildbearbeitungsprogrammes können die koordinaten eines hotspots unschwer ermittelt werden. Die einzelnen werte sind durch komma getrennt. Abhängig vom typ des hotspots sind folgende angaben nötig:

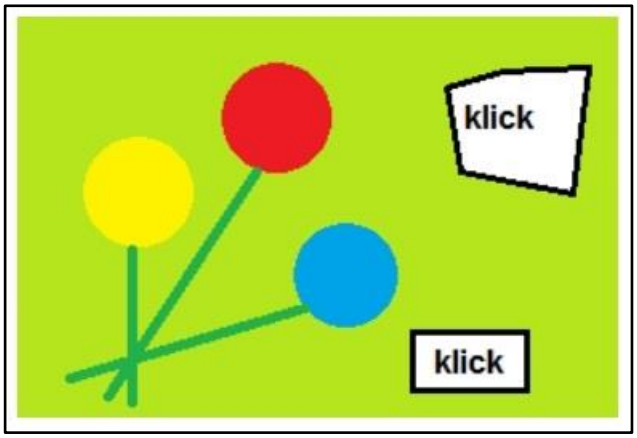
rect	die koordinaten der linken oberen und der rechten unteren ecke.
circle	die koordinaten des mittelpunks und der radius
poly	die koordinaten aller eckpunkte

beispiel
In der folgenden grafik sind der rote luftballon und die mit **klick** beschrifteten flächen als hotspots definiert.

```
<map name="maptest">
<area shape="rect" coords="194, 158, 252, 184"
   href="doku/HTML/html-test.php?art=1" title="hier klicken" >
<area shape="circle" coords="128, 52, 30"
   href="doku/HTML/html-test.php?art=1" title="hier klicken" >
<area shape="poly" coords="214, 34, 242, 30, 282, 24, 278, 85, 223, 77"
   href="doku/HTML/html-test.php?art=1" title="hier klicken" >
</map>

<p><img src="doku/HTML/im/testbild.png" usemap="#maptest" />
```

Beim aufruf der seite wird die nebenstehende grafik gezeigt, wenn man mit dem cursor einen der hotspots berührt, ändert sich der mauszeiger und es wird eine quick-info angezeigt. Bei einem klick auf einen hotspot wird die seite html-test.php aufgerufen.



8.6 Javascript-funktion als ziel

Auf etwas komplexeren seiten ist es sehr beliebt, mit einem **link** eine Javascript-funktion aufzurufen, weil man damit allerlei trickreiche dinge auf der seite treiben kann. Es ist nicht möglich, im rahmen dieser dokumentation auch Javascript zu behandeln, trotzdem wird hier wenigstens ein kleines beispiel gezeigt. Die wichtigsten möglichkeiten, das aussehen einer seite beim oder nach dem laden der seite zu verändern, sind in der Javascript-dokumentation beschrieben.

beispiel
Der angezeigte text **grafik zeigen** enthält einen **link** auf die Javascript-funktion **hinein**. Ein klick auf diese anzeige macht einen zunächst unsichtbaren container sichtbar, der eine grafik enthält. Diese grafik ist mit einem **link** verbunden, der ebenfalls die funktion **hinein** aufruft, ein klick auf die grafik bringt sie wieder zum verschwinden. Die funktion wird hier zwar gezeigt, eine erklärung würde aber viel zu weit führen.

```
<a href="Javascript: hinein('jstest', 1)">grafik zeigen</a>

<div id="jstest" style="width: 200px; hight: 200px; visibility: hidden;">
<p><a href="Javascript: hinein('jstest', 0)">
   <img src="doku/HTML/im/wappen3.jpg" /></a></p>
<p>zum schließen der anzeige auf das wappen clicken</p>
</div>
```

Die Javascript-funktion **hinein** ist direkt in die seite eingebaut.

```
<script language="javascript">
function hinein(hin, par)
{   if (par == 0)                //  container ausblenden
    {   hinakt = document.getElementById(hin).style;
        hinakt.visibility = "hidden";
    }
    else                         //  container einblenden
    {    hinakt = document.getElementById(hin).style;
         hinakt.visibility = "visible";
    }
}
</script>
```

grafik zeigen

zum schließen der anzeige auf das wappen clicken

9. formulare

Formulare werden in komplexen WEB-anwendungen vielfach eingesetzt, um informationen zwischen verschiedenen seiten auszutauschen. Es ist relativ einfach, mit **HTML** formulare aufzubauen und an eine andere seite weiter zu geben, wobei die weitergabe immer über einen server realisiert wird. Für die auswertung eines übergebenen formulars reichen die mittel von **HTML** nicht aus. Vielfach wird dazu die serverseitig eingesetzte sprache **PHP** verwendet. Eine behandlung von **PHP** erfolgt nicht in dieser dokumentation. Daher wird hier nur der aufbau von formularen behandelt, weitergehende informationen über die verarbeitung von formularen sind in der **PHP-dokumentation** zu finden.

9.1 form-tag - formular definieren

<form [name="***fmname***"] action="***pfadname***" [accept-charset="***code***"] [autocomplete="ON | OFF"]
method="POST | GET">
ein oder mehrere formularlemente (siehe 9.2 und folgende)
</form>

fmname	name des formulars; die angabe ist nur nötig, wenn die aufgerufene seite den namen verlangt.
pfadname	Mit der angabe des pfadnamens einer seite wird festgelegt, dass diese seite aufgerufen und das formular an die seite übergeben wird. Es sind auch andere aktionen mit dem formular möglich, die aber hier nicht behandelt werden.
code	zeichensatz mit dem formulardaten übergeben werden; erst ab **HTML5** möglich. Einzelheiten dazu s.u..
ON | OFF	standardmäßig speichert der server die übergebenen formulardaten. Ab **HTML5** kann man mit der angabe OFF das unterbinden. Ob sich alle server daran halten ist nicht sicher.
POST | GET	eines dieser beiden schlüsselwörter ist anzugeben. Damit wird festgelegt, mit welcher methode die werte der formularelemente übergeben werden. Eine weitergehende erklärung ist erst in der PHP-dokumentation möglich.

zeichensatz für formulardaten

Bei ziffer 2.5 wurde ausgeführt, dass der zeichensatz, mit dem eine seite erstellt wurde der gleiche sein sollte, der im **header** der seite mit **charse**t vereinbart wird. In der regel wird das der zeichensatz **UTF-8** (**Unicode**) sein, bei älteren seiten auch **Ansicode** (**ISO-8859-1** oder ähnliches). Die weitergabe der eingabedatedaten von formularen ist recht unübersichtlich, wie das tests mit der zeichenfolge **Ä Ö Ü € €** zeigen (ergebnis siehe unten). Es wird empfohlen, sowohl für sender wie empfänger Unicode (UTF-8) festzulegen und ab **HTML5** im form-tag das attribut **accept-charset** zu verwenden. Mit maskierten zeichen hat man auf alle fälle nie probleme.

seite erstellt in	im header festgelegt	accept-charset	empänger erwartet	empfänger zeigt an
Ansicode	Ansicode	nicht angegeben	Ansicode	`[illegible]`
Ansicode	Ansicode	nicht angegeben	Unicode	`[illegible]`
Ansicode	Ansicode	Ansicode	Ansicode	`[illegible]`
Ansicode	Ansicode	Unicode	Unicode	Ä Ö Ü € €
Unicode	Unicode	nicht angegeben	Ansicode	Ä Ö Ü € €
Unicode	Unicode	nicht angegeben	Unicode	Ä Ö Ü € €
Unicode	Unicode	Ansicode	Ansicode	`[illegible]`
Unicode	Unicode	Unicode	Unicode	Ä Ö Ü € €

9.2 steuerbutton

Die steuerbutton werden mit dem **input-tag** definiert; das tag hat kein endetag und kann in einem formular mehrfach vorkommen.

<input type="**submit** | **reset**" [name="***elemname***"] [value="***text***"] [tabindex="***n***"] />

submit	das schlüsselwort bestimmt, dass ein button erzeugt wird, mit dem die im form-tag definierte aktion ausgeführt wird. Die dokumentation behandelt nur den fall, in dem eine seite aufgerufen und der seite das formular übergeben wird.
reset	das schlüsselwort bestimmt, dass ein button erzeugt wird, mit dem alle eingaben in das formular zurückgesetzt werden.
elemname	name des formularelements; bei **reset** eigentlich nicht nötig, bei **submit** nötig, wenn die aufgerufene seite den namen erwartet, was oft der fall ist. Auf alle fälle nötig, wenn das formular mehr als ein **submit**-element enthält.
text	beschriftung des buttons; fehlt die angabe, wird eine browserspezifische beschriftung gewählt.
tabindex="***n***"	***n*** gibt die reihenfolge an, in der das element mit dem tabulator den fokus erhält. siehe nr. 2.7 fokus setzen

hinweis
Meist wird das element in ein anderes tag (p-tag, div-tag, td-tag in einer tabellen) geschachtelt.

 [***vorsatz***] <input . . . . . /> [***nachsatz***]

vorsatz	text, der vor dem element angezeigt wird
nachsatz	text, der nach dem element angezeigt wird.

Die beschreibung der weiteren formularelemente erfolgt in dieser form also mit ***vorsatz*** und ***nachsatz*** und das ganze in ein p-tag geschachtelt.

Formatierungen, die mit attributen oder CSS im **p-tag** vorgenommen werden, gelten nur für ***vorsatz*** und ***nachsatz***. Die beschriftung der button und ggf. die eingaben in das formular können im **input-tag** mit attributen oder CSS formatiert werden.

Dieser hinweis gilt für die weiteren formularelemente entsprechend.

beispiel siehe **9.3 einzeilige texteingabe**

9.3 einzeilige texteingabe

9.3.1 HTML4 standard

Mit dieser form des **input-tags** wird ein einzeiliges feld zur texteingabe erzeugt. Das tag hat kein endetag und kann beliebig oft in einem formular vorkommen.

vorsatz<input type="**text** | **password | hidden**" name="***elemname***" [size="***nn***"] [maxlength="***max***"]
[value="***wert***"] [readonly] [tabindex="***n***"] [autofocus] />*nachsatz*

text	es wird ein einzeiliges feld für texteingabe erzeugt.
password	es wird ein einzeiliges feld für texteingabe erzeugt, die eingabe wird verschlüsselt angezeigt
hidden	es wird ein einzeiliges feld für texteingabe erzeugt, das aber nicht sichtbar ist und in das in diesem zustand keine eingabe möglich ist. Der inhalt des feldes wird aber an die aufgerufene seite übergeben. (vgl. hinweise)
elemname	name des elements, notwendig
nn	numerische angabe; damit wird die größe des eingabefeldes festlegt; eine längere eingabe wird gescrollt.
max	numerische angabe; damit wird die maximale anzahl von zeichen festlegt, die eingegeben werden können.
wert	vorbelegung des eingabefeldes, die überschrieben werden kann.
readonly	es ist keine eingabe möglich, eine vorbelegung kann nicht überschrieben werden, der inhalt wird aber an die aufgerufene seite übergeben (vgl. hinweise)
tabindex="***n"***	***n*** gibt die reihenfolge an, in der das element mit dem tabulator den fokus erhält. siehe nr. 2.7 fokus setzen
autofocus	beim laden der seite erhält das element sofort den fokus; siehe nr. 2.7 fokus setzen.

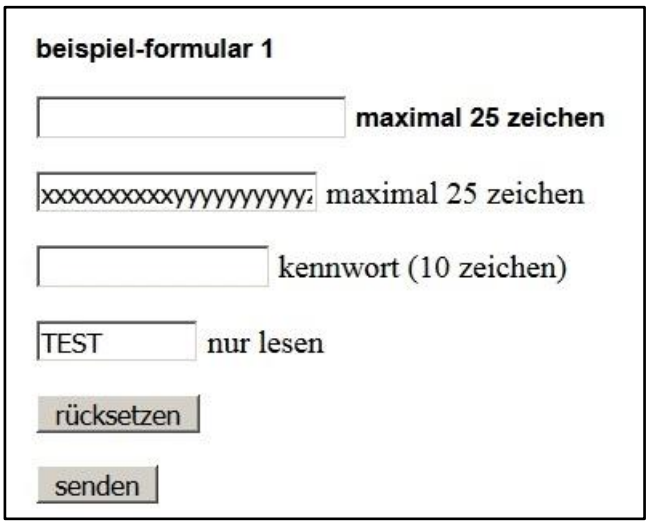
hinweise

- Ein unsichtbares eingabefeld erscheint zunächst nicht sinnvoll, aber es wird gerne verwendet, um an die aufgerufene seite informationen zu übergeben, die der anwender nicht sehen soll. In mit **PHP** gestalteten seiten kann ein unsichtbares feld auch von einer bedingung abhängig sichtbar gemacht werden.
- Ein eingabefeld, in das nichts eingegeben werden kann (**readonly**), wird gerne verwendet, um an die aufgerufene seite informationen zu übergeben, die der anwender zwar sehen, aber nicht verändern kann.

beispiel-formular 1

Im eingabefeld feld1 werden vorsatz, nachsatz und die eingegebenen daten mit der CSS-klasse **std** formatiert, das eingabefeld feld2 enthält eine vorbelegung, mit dem eingabefeld feld3 wird ein kennwort eingegeben und das eingabefeld feld4 enthält eine vorbelegung, die nicht überschrieben werden kann..

```
<form name="form1"
   action="doku/HTML/form-test.php?art=1" method="POST">
<p class="std">beispiel-formular 1<br /><br >
   <input type="text" name="feld1" size="20"
   maxlength="25" autofocus class="std" />
   maximal 25 zeichen</p>
<p><input type="text" name="feld2" size="20"
   maxlength="25"
   value="xxxxxxxxxxyyyyyyyyyyzzzzz" />
   maximal 25 zeichen</p>
<p><input type="password" name="feld3"
   size="16" maxlength="10" />
   kennwort (10 zeichen)</p>
<p><input type="text" name="feld4" size="10"
   value="TEST" readonly />nur lesen</p>
<p><input type="reset" value="rücksetzen" /></p>
<p><input type="submit" name="sender"
   value="senden" /> </p>
</form>
```



9.3.2 neues bei HTML5

Ab HTML5 gibt es neuerungen, von denen hier die wichtigsten vorgestellt werden.

übersichtlicher code

<label> [*vorsatz*] <input . . . [placeholder="***hinweis***"] [**required**] /> [*nachsatz*] **</label>**

label das gesamte **input**-tag wird in das **label**-tag eingeschlossen. Das hat zunächst keine erkennbare wirkung, das neue tag entspricht vielmehr den logischen abschnitten (siehe 3.4), es macht nur den code etwas übersichtlicher. Man kann das tag aber zur formatierung verwenden, das hat dann die gleiche wirkung wie eine formatierung im **p**-tag.

Das label-tag kann bei allen formularelementen verwendet werden, es darf aber nur ein element einschließen.

hinweis der text des platzhalters wird im eingabefeld angezeigt, ist aber keine vorbelegung. Der text verschwindet, sobald man in das feld etwas eingibt. Der platzhalter wird nie zum server übertragen.

required mit diesem schlüsselwort wird eine eingabe in das feld erzwungen, es ist nicht sinnvoll, wenn das feld eine vorbelegung enthält.

numerische eingabe

Zu diesem zweck gibt es zwei neue angaben für das **type**-attribut und einige dazu gehörige weitere attribute.

<label> [vorsatz] <input . . .type="**numbe**r | **range**"
min="**minimum**" max="**maximum**" step="***schritt***" value="***wert***" . . . /> [nachsatz] </label>

number - numerisches eingabefeld
Es ist nur eine ganzzahlige eingabe zulässig; mit ***minmum*** und ***maximum*** kann man den niedrigsten und den höchsten zulässigen wert angeben. Bei einer unzulässigen eingabe wird das absenden des formulars verweigert.

Die eingabe des wertes ist aber auch ganz anders möglich: das eingabefeld wird als **zahlenauswahlfeld** angezeigt. Ohne weitere attribute gibt man mit schrittweite **eins** auf- oder abwärts schaltend den gewünschten wert ein. Mit ***minimum*** und ***maximum*** gibt man auch hier den niedrigsten und höchsten wert, mit ***schritt*** die schrittweite an. Mit ***wert*** gibt man an, welcher anfangswert für die auswahl eingestellt ist. Es können auch negative werte angegeben werden. Eine unzulässige eingabe wird hier zuverlässig verhindert.

range – numerischer schieberegler
Hier muss man mit den attributen **min**, **max**, **schritt** und **value** passende werte einstellen. Dann wird das eingabefeld als **schieberegler** angezeigt, mit dem man einen wert auswählt. Leider zeigen alle browser derzeit nur einen leeren, d.h. unbeschrifteten schieberegler an, d.h. man kann weder den eingestellten wertebereich noch den gewählten wert erkennen. Das macht das ganze eigentlich unpraktikabel.

beispiel 1a

feld1 texteingabefeld, in dem ein platzhalter angezeigt wird; eine eingabe wird erzwungen, das feld erhält beim aufruf der seite den fokus. Der nachsatz und die eingabe werden mit einer CSS-klasse formatiert.

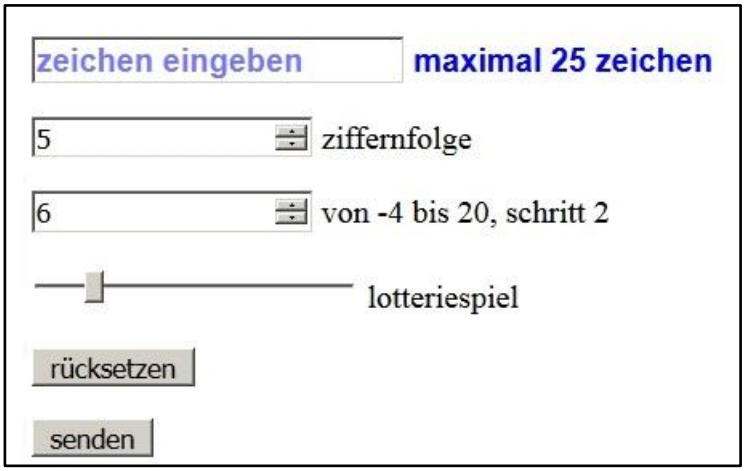
feld2 numerisches eingabefeld, eine eingabe zwischen -4 und 100 ist nötig

feld3 numerisches eingabefeld, werte zwischen -4 und 20, schrittweite 2

feld4 numerische eingabefeld, werte wie feld3, aber anzeige als schieberegler

```
<form name="form1a" action="form-test.php?&art=1" method="POST">
<p><label class="blaufett"><input type="text" name="feld1" size="20"
   maxlength="25" placeholder="zeichen eingeben" autofocus required
   class="blaufett" /> maximal 25 zeichen</label></p>
<p><label><input type="number" name="feld2" size="20" required
   min="-4" max="100" /> ziffernfolge</label></p>
<p><label><input type="number" name="feld3" size="20"
   value="0" min="-4" max="20" step="2" />
   von -4 bis 20, schritt 2</label></p>
<p><label><input type="range" name="feld4" size="20"
   value="0"  min="-4" max="20" step="2" /> lotteriespiel</label></p>
<p><label><input type="reset" value="rücksetzen" /></label> </p>
<p><label><input type="submit" name="sender" value="senden" /></label></p>
</form>
```

Die abbildung zeigt das formular, nachdem in den beiden numerischen eingabefelder bereits ein wert ausgewählt wurd. Im text-eingabefeld **feld1** steht noch der sog. platzhalter und der schieberegler in **feld4** steht auf einem nicht erkennbaren startwert.



9.4 textarea-tag - mehrzeilige texteingabe

Mit diesem tag wird ein eingabefeld für eine mehrzeilige texteingabe definiert.

 [*vorsatz*] <textarea name***="elemname***" cols="***cc***" rows="***rr***" maxlength="***max***" [readonly]
[tabindex="***n***"] [autofocus] > ***vorbelegung*** </textarea> [*nachsatz*]

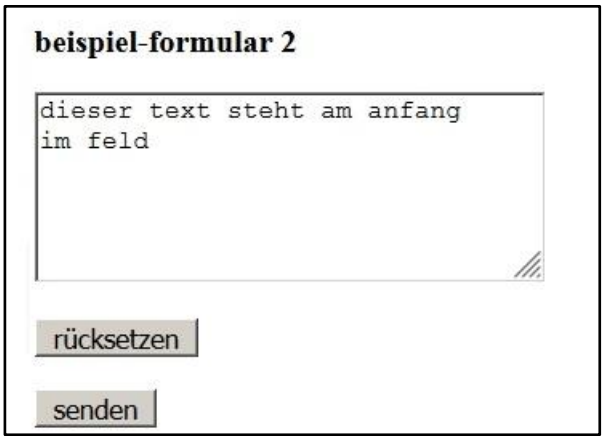
elemname	name des elements
cc	anzahl der zeichen je zeile
rr	anzahl der zeilen, die angezeigt werden
max	anzahl der zeichen, die eingegeben werden können; fehlt diese angabe, ist die anzahl unbegrenzt, das eingabefeld wird ggf. gescrollt.
readonly	es ist keine eingabe möglich, eine etwaige vorbelegung kann nicht überschrieben werden, das feld wird aber an die aufgerufene seite übergeben.
vorbelegung	das feld kann mit text vorbelegt werden.
tabindex="***n***"	***n*** gibt die reihenfolge an, in der das element mit dem tabulator den fokus erhält. siehe nr. 2.7 fokus setzen
autofocus	beim laden der seite erhält das element sofort den fokus; siehe nr. 2.7 fokus setzen.

hinweise

- Das tag hat ein ende-tag
- Die vorbelegung wird nicht mit dem attribut **value** angegeben, sondern steht zwischen dem anfangs- und dem ende-tag **textarea**
- In der vorbelegung kann mit dem maskierten zeilenumbuch **&#0010** ein zeilenumbruch erzeugt werden, der aber ebenfalls nur als leerzeichen weitergegeben wird.
- In der eingabe kann mit der enter-taste ein zeilenumbruch erzeugt werden, der aber nur als leerzeichen weitergegeben wird.
- in der vorbelegung kann mit dem maskierten zeilenumbuch **&#0010** ein zeilenumbruch erzeugt werden, der aber ebenfalls nur als leerzeichen weitergegeben wird.
- Die vorbelegung und die texteingabe werden mit einer vom browser abhängigen schriftart formatiert. In der praxis wird man die formatierung mit CSS vornehmen.

beispiel-formular 2

```
<p><b>beispiel-formular 2</b></p>
<form name="form2" action="doku/HTML/form-test.php?art=2"
  method="POST">
<p><textarea name="feld" cols="30" rows="5" maxlength="250" class="font12">
dieser text steht am anfang&#0010im feld </textarea></p>
<p><input type="reset" value="rücksetzen" /> </p>
<p><input type="submit" name="sender" value="senden" /> </p>
</form>
```



9.5 radiobutton

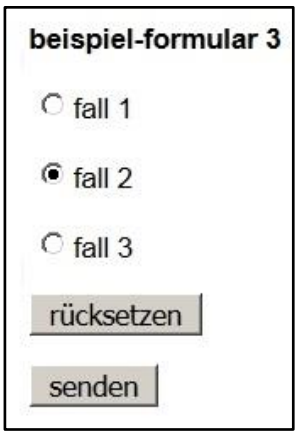
Mit dieser form des **input-tags** wird ein radiobutton erzeugt; das tag hat kein ende-tag und kann beliebig oft in einem formular vorkommen.

 [*vorsatz*] <input type="**radio**" name="***elemname***" value="***wert***" [checked="**checked**"]
[tabindex="***n***"] [autofocus] /> [*nachsatz*]

elemname	name des elements; haben in einem formular mehrere radiobutton den gleichen namen, bilden sie eine gruppe, aus der immer nur ein element ausgewählt werden kann.
wert	dieser wert wird an die aufgerufene seite übergeben.
checked	schlüsselwort, mit dem ein element als ausgewählt markiert wird. In einer gruppe kann nur ein element markiert werden. Da diese vorauswahl nur durch die auswahl eines anderen radiobuttons der gruppe aufgehoben werden kann, ist sichergestellt, dass bei der übergabe des formulars an die aufgerufene seite in jedem fall ein radiobutton ausgewählt ist.
tabindex="***n***"	***n*** gibt die reihenfolge an, in der das element mit dem tabulator den fokus erhält. siehe nr. 2.7 fokus setzen
autofocus	beim laden der seite erhält das element sofort den fokus; siehe nr. 2.7 fokus setzen.

beispiel-formular 3

```
<p><b>beispiel-formular 3</b></p>
<form name="form3" action="doku/HTML/form-test.php?art=3"  method="POST">
<p><input type="radio" name="radbut" value="button1" /> fall 1 </p>
<p><input type="radio" name="radbut" value="button2"
   checked="checked" />fall 2</p>
<p><input type="radio" name="radbut" value="button3" /> fall 3 </p>
<p><input type="reset" value="rücksetzen" /> </p>
<p><input type="submit" name="sender" value="senden" /> </p>
</form>
```



9.6 checkbox (kontroll-kästchen)

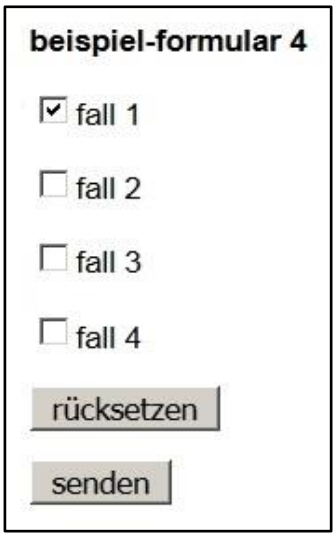
Mit dieser form des **input-tags** wird eine checkbox erzeugt; das tag hat kein ende-tag und kann beliebig oft in einem formular vorkommen. Eine checkbox wird durch anklicken ausgewählt bzw. die auswahl wird dadurch zurückgenommen. Enthält ein formular mehrere checkboxen, können mehr als eine ausgewählt werden.

 [vorsatz] <input type="**checkbox**" name="***elemname***" value="***wert***" [checked="**checked**"]
[tabindex="***n***"] [autofocus] /> [nachsatz]

elemname	name des elements; im unterschied zu den radiobutton müssen hier mehrere checkboxen in einem formular unterschiedliche namen haben.
wert	dieser wert wird an an die aufgerufene seite übergeben.
checked	schlüsselwort, mit dem ein element als vorausgewählt markiert wird. Enthält ein formular mehrere checkboxen, können mehr als eine markiert werden. Durch das schlüsselwort kann **nicht** sichergestellt werden, dass bei der übergabe des formulars an die aufgerufene seite tatsächlich eine checkbox ausgewählt ist.
tabindex="***n***"	***n*** gibt die reihenfolge an, in der das element mit dem tabulator den fokus erhält. siehe nr. 2.7 fokus setzen
autofocus	beim laden der seite erhält das element sofort den fokus; siehe nr. 2.7 fokus setzen.

beispiel-formular 4

```
<p><b>beispiel-formular 4</b></p>
<form name="form4" action="doku/HTML/form-test.php?art=4" method="POST">
<p><input type="checkbox" name="box1" value="ABCD"
   checked="checked" />fall 1 </p>
<p><input type="checkbox" name="box2" value="XYZ" /> fall 2 </p>
<p><input type="checkbox" name="box3" value="irgend etwas" /> fall 3 </p>
<p><input type="checkbox" name="box4" value="unsinn" /> fall 4</p>
<p><input type="reset" value="rücksetzen" /> </p>
<p><input type="submit" name="sender" value="senden" /> </p>
</form>
```



9.7 select-tag - auswahl-liste

Das **select-tag** erzeugt mit hilfe von **option-tags** eine liste von texten, denen jeweils ein wert zugeordnet ist. Aus der liste kann eine zeile ausgewählt werden.

 [*vorsatz*] <select name="***elemname***" size="***nn***">

<option [value="***wert***"] [selected="selected"] > ***text*** </option>

weitere option-tags

</select> [*nachsatz*]

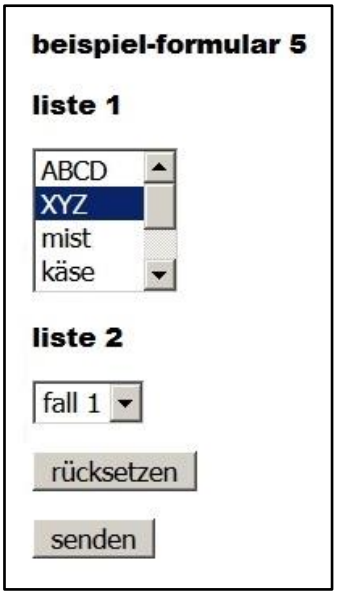
elemname	name des select-elements
nn	numerische angabe, mit der die anzahl der option-elemente festgelegt wird, die als zeilen der liste angezeigt werden. Sind mehr option-tags vorhanden, enthält die liste einen scroll-balken. Wird der wert 1 angegeben, wird nur die erste zeile angezeigt. Diese zeile enthält ein symbol, auf das geklickt werden kann, um die ganze liste aufzuklappen.
wert	dieser wert wird an die aufgerufene seite übergeben, wenn die zeile ausgewählt wurde. Fehlt die angabe, wird ***text*** übergeben.
text	dieser text wird als zeile in der liste angezeigt und wird an die aufgerufene seite übergeben, wenn kein ***wert*** mit dem attribut **value** angegeben ist.
selected	damit wird eine zeile der liste als ausgewählt markiert; es kann nur eine zeile markiert werden. Da diese vorauswahl nur durch die auswahl einer anderen zeile aufgehoben werden kann, ist sichergestellt, dass bei der übergabe des formulars an die aufgerufene seite in jedem fall eine zeile ausgewählt ist.

beispiel-formular 5

Die **liste 1** hat 6 zeilen, es werden aber nur 4 angezeigt, außerdem wird das attribut **value** nicht verwendet. Die **liste 2** hat 4 zeilen, es wird aber nur eine angezeigt. Hier wird das attribut **value** verwendet.

```
<p><b>beispiel-formular 5</b></p>
<form action="doku/HTML/form-test.php?art=5" method="POST">
<p><b>liste 1</b></p>
<p><select name="liste1" size="4">
<option> ABCD </option>
<option selected="selected"> XYZ </option>
<option> mist </option>
<option> käse </option>
<option> quatsch </option>
<option> ende </option>
</select></p>

<p><b>liste 2</b></p>
<p><select name="liste2" size="1">
<option value="allerlei"> fall 1 </option>
<option value="sonderbares"> fall2 </option>
<option value="und unklares"> fall 3 </option>
ption value="gelumpe"> fall 4 </option>
</select></p>
<p><input type="reset" value="rücksetzen" /> </p>
<p><input type="submit" name="sender" value="senden" /> </p>
</form>
```



9.8 select multiple-tag - mehrfach-auswahlliste

Das **select-multiple-tag** erzeugt mit hilfe von **option-tags** eine liste von texten, denen jeweils ein wert zugeordnet ist. Aus der liste können mehrere zeilen ausgewählt werden, indem man den mauszeiger über mehrere zeilen zieht oder eine zeile anklickt, die taste **umschalten rücksetzen** drückt und eine weitere zeile anklickt.

 [*vorsatz*] <select **multiple** name="***elemname[ ]***" size="***nn***">

<option [value="***wert***"] [selected="**selected**"] > ***text*** </option>

weitere option-tags

</select> [*nachsatz*]

multiple	durch diesen zusatz wird eine mehrfach-auswahlliste definiert.
elemname[ ]	name des select-elements; Da die werte von mehreren zeilen übergeben werden können, muß der elementname als **feld** definiert werden, das geschieht durch die eckigen klammern.
nn	numerische angabe, mit der die anzahl der option-elemente festgelegt wird, die als zeilen der liste angezeigt werden. Sind mehr option-tags vorhanden, enthält die liste einen scroll-balken. Wird der wert 1 angegeben, wird nur die erste zeile angezeigt. Diese zeile enthält ein symbol, auf das geklickt werden kann, um die ganze liste aufzuklappen.
wert	dieser wert wird an die aufgerufene seite übergeben, wenn die zeile ausgewählt wurde. Fehlt die angabe, wird ***text*** übergeben.
text	dieser text wird als zeile in der liste angezeigt und wird an die aufgerufene seite übergeben, wenn kein ***wert*** mit dem attribut **value** angegeben ist.
selected	damit wird eine zeile der liste als ausgewählt markiert; es können mehrere zeilen markiert werden.

Der anwender kann aus der anzeige der liste nicht erkennen, ob es sich um eine einfache oder mehrfache auswahlliste handelt.

beispiel

```
<p><b>beispiel-formular 6</b></p>
<form action="doku/HTML/form-test.php?art=6"
      method="POST">
<p><select multiple name="auswahl[]" size="6">
<option value="wert1">auswahl 1</option>
<option value="wert2">auswahl 2</option>
<option value="wert3">auswahl 3</option>
<option value="wert4">auswahl 4</option>
<option value="wert5">auswahl 5</option>
<option value="wert6">auswahl 6</option>
</select></p>
<p><input type="submit" name="sender"
    value="senden" /> </p>
</form>
```