HTML Tabellen

Stand: 15.08.2022

Tabelle mit HTML
Aufbau einer Tabelle

HTML unterstützt alle wichtigen Features, um tabellarische Informationen auszuzeichnen. **Tabellen** haben in HTML freilich eine lange und bewegte Geschichte. Viel zu lange hatte das W3-Konsortium gar keine Tabellenauszeichnungen in HTML
vorgesehen. Seit sich jedoch 1995 der revolutionäre Browser Netscape 1.1 über den damals aktuellen HTML-Standard 2.0 hinwegsetzte und erstmals Tabellen in HTML unterstützte, tat sich für all jene, die sich endlich mehr Freiheiten bei der Gestaltung
von Webseiten wünschten, eine neue Tür auf. Vor allem die rahmenlosen Tabellen erlaubten es, Inhalte wie Navigation und Information attraktiv nebeneinander zu positionieren. Die so genannten Tabellenlayouts waren geboren. Ganze Webdesigner-Generationen
lernten, ins *HTML-Grundgerüst* gleich mal ein paar verschachtelte Tabellen einzubauen, um so das gewünschte Basislayout zu bestimmen. Es gab sogar renommierte WYSIWYG-Editoren, die am Bildschirm ein Zentimeterraster darstellten, in dem der
Designer seine Webseiteninhalte frei positionieren konnte. Intern wurde das Raster dann als komplexe Tabelle aufgelöst. Der dabei entstehende HTML-Code war für Menschen kaum noch genießbar und hatte mit der ursprünglichen Idee von HTML nicht mehr
viel zu tun.

> Aus moderner Sicht sind solche Art von Tabellenlayouts längst verpönt.

Es spricht zwar nichts dagegen, für nebeneinander darzustellende Informationen auch mal eine rahmenlose Tabelle zu verwenden, etwa, wenn ein Bild mit nebenstehendem Text dargestellt werden soll, oder ein Formular mit ordentlich untereinander stehenden
Feldern und unterschiedlich langen Feldbeschriftungen davor. Doch Tabellen wie selbstverständlich als Basis für die grundsätzliche Informationsverteilung auf einer Seite einzusetzen, ist keine saubere Praxis mehr. Dazu gibt es mittlerweile in CSS die
Möglichkeit, beliebige Elemente und Bereiche wie gewünscht zu positionieren und zueinander anzuordnen. Gewöhnen Sie sich also an, Tabellen nur für tabellarische Informationen zu verwenden und rahmenlose Tabellen für nebeneinander darzustellende Informationen
nur für sinnvolle Einzelfälle innerhalb einer Webseite, jedoch nicht als Basis für das gesamte *Seitenlayout*.

Tabelle

Das nachfolgende Quelltextbeispiel zeigt eine kleine Tabelle:

```
<table border="1">
```

```
<caption>Vergleich von Äpfel und Birnen</caption>
<thead>
<tr>
<th>Äpfel</th>
<th>Birnen</th>
</tr>
</thead>
<tfoot>
<tr>
<td>beliebter</td>
<td>weniger beliebt</td>
</tr>
</tfoot>
<tbody>
<tr>
<td>12 Einheiten Vitamin C</td>
<td>5 Einheiten Vitamin C</td>
</tr>
<tr>
<td>regt die Verdauung an</td>
<td>wirkt entwässernd</td>
</tr>
</tbody>
</table>
```

Info Eine vollständige Tabelle in **HTML** besteht aus einer Tabellenüberschrift, einem Tabellenkopf, einem Tabellenkörper und einem Tabellenfuß. Die gesamte Tabelle wird in die Tags `<table>` bzw. `</table>`eingeschlossen. Soll die Tabelle einen sichtbaren Rahmen und ein sichtbares Gitternetz haben, müssen Sie im einleitenden Tag das Attribut `border="1"` notieren (die Zahl steht für die Rahmendicke in Pixel; für dickere Rahmen können Sie also
auch einen höheren Wert angeben). Fehlt dieses Attribut, bleiben alle Ränder der Tabelle unsichtbar. Die etwas klobig wirkende Default-Darstellung von Rahmen und Gitternetz im Browser, wie in der Abbildung zu sehen, können Sie später mithilfe von CSS
beeinflussen. Betrachten wir nun den internen Aufbau der Tabelle. Die Tabellenüberschrift, markiert durch `<caption>...</caption>`, muss direkt innerhalb von `<table>...</table>`notiert werden, am besten direkt nach dem einleitenden `<table>`-Tag. Die Tabellenüberschrift kann Text und andere `Inline`-Elemente enthalten. Selbstverständlich lässt sie sich später mithilfe von CSS formatieren und auch im Verhältnis zur Tabelle genau ausrichten und positionieren. Die drei Bereiche für Tabellenkopf, Tabellenfuß und Tabellenkörper werden durch `<thead>...</thead>`, `<tfoot>...</tfoot>`und `<tbody>...</tbody>` markiert. Beachtenswert ist die Reihenfolge der Notation von Tabellenkopf, Tabellenfuß und Tabellenkörper in eben dieser Reihenfolge. Obwohl die Fußdaten am Ende der Tabelle erscheinen, müssen
sie vor den Daten des Tabellenkörpers notiert werden!

Falls Sie eine Tabelle erstellen möchten,

dabei aber weder eine Tabellenüberschrift wünschen noch explizit zwischen Tabellenkopf, Tabellenfuß und Tabellenkörper unterscheiden möchten, können Sie beruhigt sein: Es ist durchaus erlaubt, all diese Elemente wegzulassen und nur die Elemente zum Definieren
von Reihen und darin enthaltenen Zellen zu notieren. Bedenken Sie jedoch, dass vor allem im Fall von komplexen Tabellen, bei denen sich z.B. bereits der Kopfbereich über mehrere Zeilen erstreckt, die explizite Unterscheidung der Tabellenteile im Markup
durchaus sinnvoll sein kann. Immerhin können Sie auch für diese Elemente eigene CSS-Formatierungen definieren. Noch wichtiger aber sind die entsprechenden Auszeichnungen in Hinblick auf eine rein akustische Ausgabe der Webseite, wie etwa stark Sehbehinderte
sie zum Surfen im Web benutzen. Weiter unten werden wir noch genauer auf diese Problematik eingehen. Die Tabelle aus dem obigen Beispiel darf jedenfalls auch wie folgt aussehen und wäre dennoch korrektes HTML:

```
<table border="1">
<tr>
<th>Äpfel</th>
<th>Birnen</th>
</tr>
<tr>
<td>12 Einheiten Vitamin C</td>
<td>5 Einheiten Vitamin C</td>
</tr>
<tr>
<td>regt die Verdauung an</td>
<td>wirkt entwässernd</td>
</tr>
<tr>
<td>beliebter</td>
<td>weniger beliebt</td>
</tr>
</table>
```

Info In dieser Variante der Tabelle fehlen die Auszeichnungen für Tabellenüberschrift, Tabellenkopf, Tabellenkörper und Tabellenfuß. Die semantische Information zum Tabellenfuß geht dabei komplett verloren. Die Tabellenzeile, die in der Variante
zuvor als Tabellenfuß ausgezeichnet wurde, wird nun einfach unten als letzte Zeile notiert. Die Kopfinformationen sind dagegen erhalten geblieben, wofür das `th`-Element verantwortlich ist.

Doch der Reihe nach:

Tabelleninhalte werden in HTML Tabellenzeile für Tabellenzeile notiert. Jede Tabellenzeile wird durch `<tr>` eingeleitet und endet mit `</tr>` – `tr` steht für `table row` (zu Deutsch: Tabellenzeile).
Zwischen `<tr>` und `</tr>` werden jeweils die Spalten in der betreffenden Zeile notiert, also die Tabellenzellen dieser Tabellenzeile. Es gibt zwei Arten, nämlich Kopf- und Datenzellen. Kopfzellen werden durch `<th>...</th>`markiert und Datenzellen durch `<td>...</td>`. Dabei steht `th` für `table head` (zu Deutsch: Tabellenkopf) und `td` für `table data` (zu Deutsch: Tabellendaten). Mit `th`ausgezeichnete Zellen werden in den meisten Browsern optisch anders dargestellt als `td`-Zellen – in der Regel erscheinen Kopfzellen zentriert und in Fettschrift, während Datenzellen linksbündig und mit normalem Schriftgewicht dargestellt werden.
Auch hier gilt wieder: Stören Sie sich nicht an den Default-Formatierungen. Mithilfe von **CSS** werden Sie später alle Zellen, Zellen eines bestimmten Typs oder einzelne Zellen nach Wunsch formatieren, ausrichten usw. können. In unserem
einfachen Beispiel enthalten die einzelnen Tabellenzellen für Kopfdaten und Tabellendaten nichts als kurzen Textinhalt. HTML-Tabellen sind jedoch sehr mächtig. Jede einzelne Tabellenzelle darf neben reinem Text auch beliebige `Block`– und `Inline`-Elemente
enthalten, also strukturierte Bereiche, Überschriften, Listen und Textabsätze, Bild- und Multimediareferenzen und sogar komplette weitere Tabellen!

Erscheinungsbild von Außenrahmen und Gitternetz

Wenn Sie beispielsweise ein „offenes“ Gitternetz in Ihrer **Tabelle** wünschen, also ohne sichtbaren Außenrahmen, oder beispielsweise nur sichtbare Linien zwischen den Tabellenspalten, nicht aber zwischen den Zeilen, so stehen hierfür eine
Reihe von HTML-Attributen zur Verfügung.

Um für eine Tabelle ein offenes Gitternetz ohne sichtbaren Außenrahmen zu definieren, notieren Sie im einleitenden `<table>`-Tag:

```
<table border="1" frame="void">
```

Info Durch das Attribut `frame=` lässt sich eine Regel definieren, wie der Außenrahmen dargestellt werden soll. Mit der Wertzuweisung `void` wird der Außenrahmen komplett unterdrückt. Möglich sind aber auch die Wertzuweisungen
`above` (Rahmen nur oben), `below` (Rahmen nur unten), `hsides` (Rahmen nur oben und unten), `vsides` (Rahmen nur links und rechts), `lhs` (Rahmen nur links) und `rhs` (Rahmen nur rechts).
Um nur zwischen den Spalten der Tabelle sichtbare Linien darzustellen, nicht aber zwischen den Zeilen, können Sie notieren:

```
<table border="1" frame="void" rules="cols">
```

Info Um zu definieren, welche Linien des **Gitternetzes** angezeigt werden sollen, dient das Attribut `rules=` im einleitenden `<table>`-Tag. Mit der Wertzuweisung `cols` bestimmen Sie, dass nur Linien zwischen Spalten angezeigt werden. Umgekehrt erreichen Sie mit dem Wert `rows`, dass nur Linien

zwischen den Tabellenzeilen, nicht aber zwischen den Spalten angezeigt werden. Wenn Sie das Markup für Tabellenkopf, Tabellenfuß und Tabellenkörper
vollständig notiert haben, ist außerdem noch der Wert `groups` zulässig. Damit werden nur waagerechte Linien zwischen den Teilen der Tabelle angezeigt. Die Abbildung zeigt, wie sich eine solche Beeinflussung der Liniendarstellung auswirkt.
In diesem Fall haben wir folgende Angaben notiert:

```
<table border="1" frame="void" rules="groups">
```

Info Die Darstellung im Browser hat nun das gewünschte Grundaussehen. Feinheiten der Formatierung wie Abstände innerhalb von Zellen usw. lassen sich nachträglich mit CSS realisieren.

Die wichtigsten Elemente

Eine Tabelle in HTML setzt sich aus einer oder mehreren Tabellenzeilen und einer oder mehreren Tabellenzellen zusammen. Die folgende Abbildung zeigt eine **Tabelle** mit drei Reihen und vier Spalten (bzw. vier Zellen pro Reihe).

Beim Definieren einer Tabelle geht man sehr strukturiert vor. Zuerst wird das Grundgerüst der Tabelle eingeleitet. Nach dem Tabellenbeginn lässt sich noch eine Tabellenüber- oder -unterschrift zuweisen.
Info In HTML werden Tabellen nach Zeilen beschrieben. Von oben nach unten werden die einzelnen Reihen und deren Zellen definiert. Für die Tabellenzellen stehen zwei Zellentypen zur Verfügung: Datenzellen und Kopfzellen. Der Unterschied zwischen
den beiden besteht in der Darstellung. Text, der in Kopfzellen steht, wird in den Browsern meistens fett und zentriert dargestellt. Möchten Sie also die Tabelle in Abbildung 1 mittels HTML darstellen, müssen Sie zuerst die Tabelle einleiten, ihr eine
Tabellenunterschrift zuweisen und die erste Reihe definieren, gefolgt von vier Tabellenzellen in der Reihenfolge „Fahrer/Strecke“, „Australien“, „Brasilien“ und „Malaysia“. Anschließend folgen die zweite und dritte Reihe, bei denen Sie exakt wie bei der
ersten Reihe vorgehen, also immer mit der ganz linken Zelle beginnen und mit der ganz rechten Zelle aufhören.
Info Beachten Sie: Tabellen werden immer von links nach rechts und von oben nach unten definiert. Welcher Inhalt in einer Tabellenzelle dargestellt wird, ist für die Definition unerheblich, sei es nun Text, eine Grafik oder gar ein Video.

Die wichtigsten Elemente

Die wichtigsten Elemente, die Sie für eine Tabelle benötigen, lauten:

- `table` (dt. Tabelle) bildet das Grundgerüst einer jeden Tabelle und darf die Elemente `tr` und `caption` enthalten.
- `tr` (table row, dt. Tabellenreihe) dient der Definition einzelner Tabellenreihen und darf nur die Elemente `td` und `th` enthalten.
- `td` (table data, dt. Tabellendaten) das entsprechende Element für eine Datenzelle einer Tabelle; es

darf verschiedene weitere HTML-Elemente enthalten.
- `th` (table header, dt. Tabellenkopf) das Element zum Definieren einer Kopfzelle; es darf verschiedene weitere HTML-Elemente enthalten.
- `caption` (dt. Titel) dient zur Definition einer Tabellenüber- oder -unterschrift.

Diese eben vorgestellten Elemente dürfen nur in einer bestimmten Reihenfolge verwendet werden. Abbildung 2 soll diese Rangordnung verdeutlichen. Die Zellen mit den drei Punkten stehen für verschiedene andere **HTML-Elemente** (auch weitere
Tabellen). Um nun die Tabelle aus Abbildung 1 in HTML zu erstellen, leiten Sie mit dem Tag `<table>` das Grundgerüst der Tabelle ein. Als Nächstes folgt die Tabellenunterschrift, die Sie mit der Anweisung

```
<caption align="bottom">
Tabelle: Ergebnisse der ersten drei Rennen der Formel-1-Saison 2002
</caption>
```

definieren. Ob die Tabellenbezeichnung über oder unter der Tabelle stehen soll, legen Sie mit dem `align`-Attribut und dem Parameter `top` oder `bottom` fest. Standardmäßig wird die Bezeichnung über der Tabelle dargestellt.
Sie müssen das Attribut `align` also nur dann an `caption` übergeben, wenn Sie die Bezeichnung unter der Tabelle darstellen möchten. Nach der Überschrift definieren Sie die erste Tabellenreihe und notieren die vier Zellen. Da es
Kopfzellen sind, verwenden Sie das `th`-Element. Die entsprechende Anweisung lautet dann folgendermaßen:

```
<tr>
<th>Fahrer/Strecke</th>
<th>Australien</th>
<th>Brasilien</th>
<th>Malaysia</th>
</tr>
```

Die zweite Reihe unterscheidet sich von der ersten durch den Textinhalt und die verschiedenen Zellentypen. Sie benötigen eine Kopf- und drei Datenzellen.

```
<tr>
<th>M. Schumacher</th>
<td>1. Platz</td>
<td>3. Platz</td>
<td>1. Platz</td>
</tr>
```

Die dritte Reihe entspricht prinzipiell der zweiten, nur dass sie andere Texte enthält.

```
<tr>
<th>R. Barrichello</th>
<td>Unfall</td>
<td>Motorschaden</td>
<td>Hydraulikschaden</td>
</tr>
```

Zum Schluss beenden Sie die Tabellendefinition mit dem Tag `</table>`. Das vollständige HTML-Dokument finden Sie in Code.

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<!-- Tabellen -->
<html>
<head>
<title>Die wichtigsten Elemente</title>
</head>
<body>
<table>
<caption align="bottom">
Tabelle: Ergebnisse der ersten drei Rennen der Formel-1-Saison 2002
</caption>
<tr>
<th>Fahrer/Strecke</th>
<th>Australien</th>
<th>Brasilien</th>
<th>Malaysia</th>
</tr>
<tr>
<th>M. Schumacher</td>
<td>1. Platz</td>
<td>3. Platz</td>
<td>1. Platz</td>
</tr>
<tr>
<th>R. Barrichello</th>
<td>Unfall</td>
<td>Motorschaden</td>
<td>Hydraulikschaden</td>
</tr>
</table>
</body>
</html>
```

Das Ergebnis im Browser sieht noch nicht berauschend aus. Die einzelnen Zellen sind nur sehr schwer voneinander zu unterscheiden. Ein Rahmen könnte Abhilfe schaffen.

Größe, Rahmen und Abstände

Für die optische Gestaltung einer Tabelle stellt HTML verschiedene Attribute zur Verfügung. Die wichtigsten lauten:

- `border` (dt. Rahmen) legt die Dicke des Tabellenrahmens mit dem angegebenen Wert in Pixeln fest.
- `width` (dt. Breite) legt die Breite der Tabelle oder Zelle fest. Sie können sowohl eine prozentuale als auch eine pixelgenaue Angabe machen. Die prozentuale Angabe orientiert sich immer an der Breite des Browserfensters. Um eine solche
  Angabe zu verwenden, müssen Sie hinter den Wert das Zeichen `%` (Prozent) notieren. Für Pixelangaben notieren Sie lediglich die gewünschte Breite ohne weitere Zeichen.
- `height` (dt. Höhe) legt die Höhe der Tabelle oder Zelle fest. Die möglichen Angaben entsprechen denen des Attributs `width`. Anzumerken ist hier, dass das Attribut im offiziellen HTML-Standard für die Höhe der gesamten Tabelle
  zulässig ist. Erfahrungsgemäß kann man allerdings ohne Probleme damit arbeiten weil es von allen Browsern unterstützt wird. Sollten Sie eine Datei mit einer Höhenangabe für eine Tabelle mal validieren lassen, wird sich der Validator allerdings
  darüber beschweren.
- `cellpadding` (dt. Zellenpolster) definiert den Abstand zwischen dem Zellenrand und dem Zelleninhalt. Angaben erfolgen in Pixel oder zusätzlichen Zeichen.
- `cellspacing` (dt. Zellenabstand) mit ihm können Sie den Abstand zwischen den Zellen festlegen. Auch hier erfolgen Angaben in Pixel.

Mit diesen neuen Attributen passen Sie nun das Code an, um dem gewünschten Ergebnis näher zu kommen (Abbildung 1). Da Sie die Zellen nicht alle einzeln verändern möchten, übergeben Sie die Werte im Start-Tag des `table`-Elements. Den **Tabellenrahmen**verändern Sie auf 1 Pixel Breite, die Breite der Tabelle auf 75 % (des Browserfensters), die Höhe auf 25 % (des Browserfensters), den Zellenabstand auf 1 Pixel und den Innenabstand der Zellen auf 2 Pixel. Das veränderte Start-Tag lautet dann:

```
...
<table border="1" width="75%" height="25%" cellspacing="1" cellpadding="2">
...
```

Die Darstellung in Abbildung 4 entspricht schon viel eher der Referenztabelle aus Abbildung 1.

Elemente positionieren

In einem Layoutprogramm wie Adobe InDesign oder Quark XPress ziehen Sie Absätze und Bilder einfach an die Stellen, an die Sie sie haben möchten. Ähnlich läuft das Zeichnen von Folien in Microsoft PowerPoint oder in Open Office Impress. Im Web, das heißt
mit HTML und CSS, ist die Sache nicht ganz so einfach. Zwar wollen viele WYSIWYG-Editoren wie

Macromedia Dreamweaver oder Microsoft FrontPage diesen Eindruck erwecken, aber er täuscht.

Ursprünglich gab es zum Positionieren in HTML nur einen Trick: **Tabellen** – eigentlich zur Datenpräsentation gedacht, lassen sie sich ohne Rahmen und Abstände zum Platzieren von Elementen verwenden. Eine echte Positioniermöglichkeit kam
erst in CSS hinzu. Allerdings ist sie in älteren Browsern, vor allem dem Netscape Navigator 4.x, zumindest problematisch und nach wie vor eine aufwändige Angelegenheit. Beide Varianten lernen Sie in diesem Artikel kennen. Wenn Sie Tabellen zum Positionieren
verwenden möchten, müssen Sie nur wissen, dass Sie die Breite und Höhe von Tabellenzellen angeben können. Damit lässt sich dann eine Layouttabelle basteln.

Unsichtbar

Wichtigste Voraussetzung für eine Tabelle zum Positionieren ist, dass der Nutzer sie nicht sehen kann. Und tatsächlich, wenn Sie sich den Quellcode verschiedenster Websites ansehen, finden Sie meist eine Tabelle und dort immer die Angabe `border="0"`.
Eine rahmenlose Tabelle reicht allerdings zum pixelgenauen Platzieren nicht. Sie müssen auch die Zellenabstände entfernen:

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

Den Rest erledigen die Größenangaben. Wollen Sie z.B. ein Bild um 100 Pixel nach rechts und um 100 Pixel nach unten auf der Seite platzieren, verwenden Sie eine solche Tabelle:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<!-- tabellen_posi.html -->
<html>
<head>
<title>Positionieren mit Tabellen</title>
</head>
<body>
<table>
<tr>
<td width="100" height="100"></td>
<td></td>
</tr>
<tr>
<td> </td>
<td>
<img src="vogel.png" alt="Echter Vogel zwischen gusseisernen Vögeln auf einem
Denkmal." title="Vogel auf Denkmal" />
</td>
```

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

Abstand zum Fensterrand

Wenn Sie im letzten Abschnitt genau hingeschaut haben, konnten Sie sehen, dass das Bild nicht exakt auf 100 Pixel positioniert ist, sondern ein wenig weiter in das Fenster hineinragt. Dies liegt daran, dass Browser automatisch einen Seitenrand in ihrem
Fenster vergeben. Da dieser nicht unbedingt einheitlich festgelegt ist und beim akkuraten **Positionieren** nur stört, sollten Sie ihn loswerden. Dafür gibt es zwei Methoden:

- Per CSS, wenn Sie für das `<body>`-Tag die Werte 0 für `margin` und `padding` vergeben.
- Per HTML mit den offiziell nicht vorgesehenen Attributen `leftmargin="0"` und `topmargin="0"` für den Internet Explorer und `marginwidth="0"` und `marginheight="0"` für alle anderen Browser.

Die reine CSS-Variante können Sie verwenden, wenn Sie auf den Netscape Navigator 4.x verzichten. Beim Einsatz von stark abwärtskompatiblen Tabellen ist die HTML-Variante zu bevorzugen. Hier ein Beispiel, in dem ein Bild wirklich exakt im oberen Eck des
Browserfensters platziert wird:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<!-- fensterabstand.html -->
<html>
<head>
<title>Fensterabstand</title>
</head>
<body leftmargin="0" topmargin="0" marginwidth="0" margin-height="0">
<div><img src="vogel.png" alt="Echter Vogel zwischen gusseisernen Vögeln auf
einem Denkmal." title="Vogel auf Denkmal" /></div>
</body>
</html>
```

Zentrieren

Wenn Sie aber eine komplette Tabelle zentrieren oder rechtsbündig ausrichten möchten, können Sie dazu `align` verwenden. Mit diesem einfachen Trick erreichen Sie ein komplett zentriertes Layout:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<!-- tabellen_zentrieren.html -->
```

```
<html>
<head>
<title>Positionieren mit Tabellen</title>
</head>
<body leftmargin="0" topmargin="0" marginwidth="0" marginheight="0">
<table border="0" cellpadding="0" cellspacing="0" width="300" align="center"
style="background-color: #BBBBDD">
<tr>
<td style="text-align: center">Der Vogelzüchterverein</td>
</tr>
<tr>
<td style="text-align: center">
<img src="vogel.png" alt="Echter Vogel zwischen gusseisernen Vögeln auf einem
Denkmal." title="Vogel auf Denkmal" /></td>
</tr>
</table>
</body>
</html>
```

Komplettes Layout

Natürlich ist jedes **Seitenlayout** individuell. Dennoch haben sich im Web schon einige Gemeinsamkeiten herausgebildet. z.B. ist die Navigationsleiste meist links oder oben und Kopf- bzw. Fußzeile sind sehr üblich geworden. Dementsprechend
gibt es einige Grundlayouts. Eines davon finden Sie hier als praktisches Beispiel für eine komplette Lösung:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<!-- tabellen_layout.html -->
<html>
<head>
<title>Layout mit Tabellen</title>
</head>
<body leftmargin="0" topmargin="0" marginwidth="0" marginheight="0">
<table border="0" cellpadding="0" cellspacing="0" width="100%">
<tr>
<td colspan="2" height="100" style="background-color: yellow;">Kopfzeile</td>
</tr>
<tr>
<td width="150" height="500" style="background-color: red;">Navigation</td>
<td style="background-color: blue;">Inhalt</td>
</tr>
<tr>
<td colspan="2" height="60" style="background-color: yellow;">Fußzeile</td>
</tr>
```

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

Info Wenn Sie ein eigenes Layout entwickeln, gibt es zwei Möglichkeiten: Entweder zeichnen Sie sich die Tabelle auf und bauen sie dann Schritt für Schritt. Oder Sie nehmen den vorliegenden Code oder einen anderen Code und ändern ihn. Ersteres liefert
meist bessere Resultate, Letzteres geht schneller.

Farben und Schrift

Tabellen lassen sich natürlich auch in **HTML** farblich gestalten. Der Text des Inhalts kann über das `font`-Element oder über CSS formatiert werden. Dabei entsprechen die Regeln denen von Textabsätzen und Überschriften. Aber
auch die Hintergrundfarbe einer Tabelle lässt sich verändern. Dafür wird das Attribut `bgcolor` verwendet.

Tabellenhintergrund

Um die Hintergrundfarbe einer ganzen Tabelle festzulegen, wird im Start-Tag des `table`-Elements das `bgcolor`-Attribut notiert. Als Parameter für dieses Attribut können Sie entweder eines der Farbworte oder einen Hex-Tripel-Wert
angeben. Möchten Sie beispielsweise die Hintergrundfarbe einer Tabelle blau färben, müsste die Anweisung mittels Farbwort folgendermaßen lauten:

```
<table bgcolor="blue">
<!-- blauer Tabellenhintergrund -->
```

Bei einem Hex-Tripel-Wert für ein dunkles Blau (`#003399`) lautet die Anweisung:

```
<table bgcolor="#003399">
<!-- dunkelblau -->
```

Zellenhintergrund

Wenn Sie nicht der gesamten Tabelle eine **Hintergrundfarbe** zuweisen möchten, stehen Ihnen noch zwei weitere Möglichkeiten offen: die Anpassung einer ganzen Tabellenzeile oder einer einzelnen Zelle. Das Attribut `bgcolor` wird
dabei jeweils im Start-Tag des `tr`-, `td`– oder `th`-Elements notiert. Natürlich können Sie sowohl Farbworte als auch Hex-Tripel-Werte angeben. Einige Beispiele:

```
<tr bgcolor="red">
<!-- ganze Zeile rot -->
```

```
<td bgcolor="yellow">
<!-- nur die Zelle gelb -->

<th bgcolor="#009900">
<!-- nur die Zelle grün -->
```

Info Achten Sie bei der Farbwahl auch immer auf die Farbe des Textes. So ist eine schwarze Schrift auf einem dunkelblauen Hintergrund nur sehr schwer bis gar nicht zu lesen. Sie sollten auf Farbkombinationen wie rote Schrift auf blauem Hintergrund
ebenso verzichten wie auf zu viele Farben. Hier gilt die Devise: Weniger ist mehr. In der Praxis haben sich gediegene Farben sehr bewährt. Auch sollten Sie nicht unbedingt mehr als zwei, maximal drei Farben für die Hintergründe von Tabellen, Zeilen und
Zellen verwenden, denn sie sollen der Übersicht und nicht zum Abschrecken der Besucher dienen.

Schriftformatierung

Für die Formatierung der Schrift innerhalb einer Zelle kann das `font`-Element verwendet werden. Ich möchte Ihnen an dieser Stelle nur ein paar Beispiele anführen:

```
<td><font face="monospace">Courier New</font></td>
<th><font size="5">Schriftgröße 5</font></th>
```

Info Nun wäre es sehr mühselig, jede einzelne Zelle mit dem `font`-Element zu formatieren, vor allem, da spätere Änderungen an der Farbe des Textes oder der Schriftart gerade bei großen **Tabellen** sehr zeitaufwändig sind.
Sie können eine Tabelle oder Zeile jedoch auch im Gültigkeitsbereich eines `font`-Elements notieren. Somit wird der gesamte Text der Tabelle oder Zeile in der definierten Formatierung dargestellt.

```
<font face="sans-serif">
<table border="1">
<tr>
<th>Kopfzelle 1</th>
<th>Kopfzelle 2</th>
</tr>
<tr>
<td>Datenzelle 1</td>
<td>Datenzelle 2</td>
</tr>
</table>
</font>
```

Bei dieser Vorgehensweise kann es Ihnen allerdings passieren, dass einige ältere Browser nicht mitspielen.

Diese neigen dazu, die Schriftart in den Zellen wieder auf die Standardeinstellung des Browsers zu setzen. Daher würde ich Ihnen empfehlen, Tabellen
mithilfe von CSS zu formatieren.

Farb- und Schriftbeispiel

Für das folgende Beispiel habe ich mich auf Grautöne beschränkt.

Features für die nicht visuelle Darstellung von Tabellen

Auch sehbehinderte Menschen nehmen am Informationsangebot im Web teil. Wenn die Sehbeeinträchtigung so stark ist, dass auch Lupen und Lichtverstärkung nichts mehr nutzen oder zu schnell ermüden, dann greifen solche Benutzer häufig zu Browser-Tools, die
Webseiten mit synthetischer Stimme vorlesen, statt sie am Bildschirm anzuzeigen.

Wie Sie sicher bemerkt haben, muss man beim Notieren der einzelnen Zellen einer Tabelle im **HTML-Quelltext** gut mitdenken. Bei größeren Tabellen und umfangreichen Zelleninhalten kann auch schnell mal die Übersicht verloren gehen. Genau
das gleiche Problem besteht, wenn Tabelleninhalte vorgelesen werden. Dabei geht nämlich der eigentliche Segen von Tabellen, also die übersichtliche Darstellung auf einen Blick, verloren. Stattdessen müssen die Zelleninhalte im Zeitkontinuum vermittelt
werden. Vorgelesen werden die Zelleninhalte dabei so, wie sie im HTML-Quelltext stehen, also Tabellenzeile für Tabellenzeile. Doch wie erfährt ein Zuhörer, welche Inhalte zum Tabellenkopf gehören und welche zum Tabellenkörper? Oder wann sich eine Zelle
über mehrere Spalten oder Zeilen erstreckt? Für diese Probleme stellt HTML eine Reihe von Attributen zur Verfügung. Wie die darin enthaltenen Informationen in einer nicht visuellen Repräsentation der Tabelle tatsächlich umgesetzt werden (z.B. durch eine
zweite Stimme, die nur solche Meta-Informationen ausgibt, während die Hauptstimme die Tabellennutzdaten vorliest), ist Sache der verwendeten Software. Als HTML-Autor können Sie jedoch die erforderlichen Meta-Daten notieren.

> Da das Vorlesen umfangreicher Tabellen etwas länger dauert, ist es sinnvoll,

den Zuhörer darauf vorzubereiten und ihm vorweg eine kurze Zusammenfassung des Tabelleninhalts zu geben. Dazu stellt HTML das Attribut `summary=` für das einleitende `<table>`-Tag zur Verfügung. Ein Beispiel:

```
<table border="1"
summary="Es folgt eine Kreuztabelle mit Entfernungen zwischen großen deutschen
Städten. Die Tabelle hat 25 Zeilen und 25 Spalten.">
```

Info Auch für einzelne **Tabellenzellen** lässt sich eine zusammenfassende Kurzbeschreibung notieren. Das bietet sich an, wenn die Inhalte der Zellen selbst recht lang und komplex sind. Auch hierzu ein Beispiel:

```
<td abbr="Zeile 4 Spalte 8, Informationen zu den Lotto-Einnahmen des
Bundeslandes Niedersachsen der letzten 10 Jahre">
```

Info Das Attribut `abbr=` können Sie in einleitenden `<td>`– oder `<th>`-Tags notieren. Beim Vorlesen des Tabelleninhalts sollte eine entsprechende Software diese Kurzbeschreibungen mit ausgeben.

Besonders problematisch bei der zeilenweisen akustischen Präsentation von Tabellen

ist die Zuordnung von Spalteninhalten zur zugehörigen Spaltenüberschrift. Auch dazu bietet HTML einen Mechanismus an. In den Spaltenüberschriften (Kopfdaten) erhält jede Zelle einen eindeutigen `id`-Namen. Die Datenzellen der jeweiligen Spalten
können dann auf diese `id`-Namen Bezug nehmen. Ein einfaches Beispiel:

```
<table border="1">
<tr>
<th id="Mo">Montag</th>
<th id="Di">Dienstag</th>
<th id="Mi">Mittwoch</th>
</tr>
<tr>
<td headers="Mo">Hühnerfrikassee mit Gemüsereis</td>
<td headers="Di">Rostzwiebelbraten mit Pommes Frites</td>
<td headers="Mi">Allgäuer Käsespätzle</td>
</tr>
</table>
```

Info In den Kopfzellen `<th>`, von denen jede eine Spaltenüberschrift darstellt, wird ein `id`-Attribut notiert. Dies ist eigentlich ein Allgemein-Attribut, das jedem **HTML-Element** zugewiesen werden kann.
Wichtig ist, dass die Wertzuweisung ein dokumentweit eindeutiger Name ist. Im Beispiel sind es Abkürzungen für Wochentage. Der Bezug zu den Spaltenüberschriften kann dann in den Datenzellen `<td>` über das Attribut `headers=`hergestellt werden.

Falls Sie eine Kreuztabelle definieren,

also eine Tabelle, bei der sowohl die erste Zeile als auch die erste Spalte Kopfdateninformationen enthält

und jede Datenzelle eine Schnittinformation aus den jeweiligen Zeilen-/Spaltenkoordinaten darstellt, können Sie auch bei den Zeilenüberschriften
`id`-Attribute notieren. In den Datenzellen können Sie dann beim `headers`-Attribut beide `id`-Namen angeben, durch Leerzeichen getrennt. Einen noch einfacheren Weg, Kopfzelleninformation in Datenzellen zu wiederholen,
bietet das `scope`-Attribut an. Ein Beispiel:

```
<table border="1">
<tr>
<th scope="col">Montag</th>
<th scope ="col">Dienstag</th>
<th scope ="col">Mittwoch</th>
</tr>
<tr>
<td>Hühnerfrikassee mit Gemüsereis</td>
<td>Rostzwiebelbraten mit Pommes Frites</td>
<td>Allgäuer Käsespätzle</td>
</tr>
</table>
```

Info Bei dieser Variante wird in den Kopfzellen `<th>` das Attribut `scope=` notiert. Falls es sich um Spaltenüberschriften handelt, wie im Beispiel, muss `scope="col"` notiert werden. Falls es sich um Zeilenüberschriften
handelt, lautet die Angabe `scope="row"`. Auch die Angaben `scope="colgroup"` (bei Spaltengruppen) und `scope="rowgroup"` (für Tabellenkopf, Tabellenkörper und Tabellenfuß) sind zulässig. Akustische Ausgabesysteme sollten
bei Verwendung des `scope`-Attributs den Zelleninhalt der zugehörigen Kopfzelle bei jeder Datenzelle wiederholen, um den Bezug transparent zu machen.

Positionieren mit CSS

Sie können mit CSS dasselbe erreichen wie mit Tabellen. Der Weg dorthin ist unter Umständen etwas schwieriger, vor allem wenn Sie Tabellen schon kennen. Außerdem sind ältere Browser wie der Netscape Navigator 4.x nur schwer mit einem solchen Layout zum
Arbeiten zu bewegen.

Die Vorteile eines CSS-Layouts sind allerdings auch vielfältig: Sie trennen das Layout besser vom Inhalt. Außerdem ist ein CSS-Layout behindertengerecht. Hierfür steht der Begriff Accessibility (dt. Zugänglichkeit). Ein Tabellenlayout ist in dieser Hinsicht
nicht so gut, da z.B. Screenreader, die eine Website vorlesen, in Layouttabellen schnell durcheinander kommen.

Absolut und relativ

Die Positionierung mit CSS kann auf verschiedene Arten erfolgen. Welche Art Sie verwenden möchten, entscheiden Sie mit dem CSS-Befehl position. Diese Werte stehen zur Wahl:

- `absolute` positioniert absolut. Sie geben einen Pixelwert für `top` (y-Koordinate) und für `left` (x-Koordinate) an. Die Positionierung erfolgt relativ zum übergeordneten Element, das heißt relativ zu dem Blockelement,
  in dem ein Element verschachtelt ist. Befindet sich ein Element auf der obersten Ebene, wird absolut zur linken oberen Ecke des Browserfensters positioniert.
- `relative` positioniert relativ zur Position eines Elements im normalen HTML-Fluss. Auch hier geben Sie `top` (y-Koordinate) und `left` (x-Koordinate) an, allerdings sind die Angaben dann relativ zum vorangegangenen
  Element zu verstehen.
- `fixed` fixiert ein Element an einer Position wie bei der absoluten Positionierung. Allerdings bewegt sich das Element beim Scrollen nicht mit. Dies wird aber nicht von allen Browsern unterstützt.
- `static` ist die normale Positionierung im HTML-Fluss. Dieser Wert muss nicht angegeben werden.

In der Praxis kommen absolute und relative Positionierung zum Einsatz. Der Nachteil der absoluten Positionierung ergibt sich aus ihrem wichtigsten Vorteil: Dadurch, dass Sie die Elemente exakt platzieren können, sind sie auch völlig unabhängig voneinander.
Sie können sich also auch bei größeren Inhalten einfach überlappen. Bei der relativen Positionierung kann das nicht passieren. Dafür haben Sie keine exakte Kontrolle über die Koordinaten eines Elements, da es ja von anderen abhängig ist. Sehen Sie sich
dazu jeweils ein Beispiel an. Zuerst Schritt für Schritt die absolute Positionierung:

- 1. Erstellen Sie zwei `<div>`-Blöcke.

Sie können auch andere Elemente absolut positionieren. In der Praxis werden allerdings sehr oft `<div>`-Blöcke verwendet, da sie selbst nicht formatieren.

- 2. Vergeben Sie für die `<div>`-Blöcke jeweils eine ID, damit Sie sie per Style Sheet ansprechen können.

```
<div id="absatz1">Ein Absatz, der positioniert wird.</div>
<div id="absatz2">Noch ein Absatz, der positioniert wird.</div>
```

- 3. Erstellen Sie ein Style Sheet im Kopf der Seite.
- 4. Greifen Sie auf die zwei IDs zu.

```
#absatz1 {
}
#absatz2 {
}
```

- 5. Vergeben Sie für beide Bereiche eine Größe von 200 * 300 Pixel und jeweils eine Hintergrundfarbe.
- 6. Positionieren Sie den ersten Bereich absolut auf 50 und 100 Pixel.

```
#absatz1 {
width: 200px; height: 300px;
background-color: red; position: absolute;
top: 50px; left: 100px;
}
```

- 7. Den zweiten Bereich positionieren Sie nach demselben Muster auf 150 und 200 Pixel.

Hier der Code in der Übersicht:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<!-- css_absolute.html -->
<html>
<head>
<title>Positionierung mit CSS</title>
<style type="text/css"><!--
#absatz1 {
width: 200px; height: 300px;
background-color: red; position: absolute;
top: 50px; left: 100px;
}
#absatz2 {
width: 200px; height: 300px;
background-color: yellow; position: absolute;
top: 150px; left: 200px;
}
--></style>
</head>
<body>
<div id="absatz1">Ein Absatz, der positioniert wird.</div>
<div id="absatz2">Noch ein Absatz, der positioniert wird.</div>
</body>
</html>
```

Nun ändern Sie zum Vergleich nur die absolute Positionierung in die relative. Die geänderten Stellen sind hervorgehoben:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<!-- css_relative.html -->
<html>
<head>
<title>Positionierung mit CSS</title>
<style type="text/css"><!--
#absatz1 {
width: 200px; height: 300px;
background-color: red; position: relative;
top: 50px; left: 100px;
}
#absatz2 {
width: 200px; height: 300px;
background-color: yellow; position: relative;
top: 150px; left: 200px;
}
--></style>
</head>
<body>
<div id="absatz1">Ein Absatz, der positioniert wird.</div>
<div id="absatz2">Noch ein Absatz, der positioniert wird.</div>
</body>
</html>
```

Sie sehen im Ergebnis, dass der zweite Absatz nun am ersten orientiert verschoben wird.

Überlappung

Wenn sich Elemente überlagern, liegt immer das Element oben, das zuletzt im HTML-Quelltext steht. Im Beispiel aus dem letzten Abschnitt ist das der zweite Absatz. Sie können allerdings auch explizit festlegen, auf welcher Ebene ein Element liegt. Dafür
dient der `z-index`. Als Wert geben Sie eine ganze Zahl an. Das Element mit der höheren Zahl liegt oben – bei gleicher Zahl entscheidet wieder die Reihenfolge im Quellcode. Der folgende Code vertauscht die Rangfolge für die zwei Absätze:

```
#absatz1 {
width: 200px; height: 300px;
background-color: red; position: absolute;
top: 50px; left: 100px; z-index: 2;
}
#absatz2 {
width: 200px; height: 300px;
background-color: yellow; position: absolute;
```

```
top: 150px; left: 200px; z-index: 1;
}
```

Zentrieren

Elemente mit CSS zu zentrieren ist ein wenig schwieriger als mit einer Tabelle. Wobei schwierig wohl nicht das richtige Wort ist. Die Wahrheit ist, dass der Trick nicht weithin bekannt ist. Zum Zentrieren verwenden Sie für `margin-left` und
`margin-right` jeweils den Wert `auto`:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<!-- css_zentrieren.html -->
<html>
<head>
<title>Positionierung mit CSS</title>
<style type="text/css"><!--
body {
margin: 0px; padding: 0px;
}
#absatz1 {
width: 400px; height: 200px;
background-color: red;
margin-left: auto; margin-right: auto; z-index: 2;
}
--></style>
</head>
<body>
<div id="absatz1">Ein zentrierter Absatz.</div>
</body>
</html>
```

Komplettes Layout

Für ein komplettes Layout mit CSS müssen Sie alle gewünschten Bereiche der Seite als `<div>`-Blöcke anlegen. Dann ist allerdings nichts mehr völliger Standard. Es gibt verschiedene Techniken, die Elemente anzuordnen. Die vertikale Anordnung
erfolgt bei absoluter Positionierung und fester Angabe der Höhe mit festen Abständen. Für die Positionierung der Navigationsleiste verwendet das folgende Beispiel einen Trick: Der Inhalt in der Mitte der Seite wird mit `margin-left` um die
Breite der Navigationsleiste nach rechts gerückt. Insgesamt ergibt sich genau dasselbe Layout, wie Sie es vorher im Abschnitt mit Tabellen gesehen haben:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
```

```
<!-- css_layout.html -->
<html>
<head>
<title>Positionierung mit CSS</title>
<style type="text/css">
<!--
body {
margin: 0px; padding: 0px
}
#kopf {
position: absolute;
left: 0px; top: 0px; z-index: 3;
width: 100%; height: 100px;
background-color: yellow;
border: none
}
#mitte {
position: absolute;
left: 0px; top: 100px; z-index: 1;
width: 100%; height: 500px;
background-color: blue;
border: none
}
#mitte_inhalt {
margin-left: 150px
}
#navigation {
position: absolute;
left: 0px; top: 100px; z-index: 2;
width: 150px; height: 500px;
background-color: red;
border: none;
}
#fuss {
position: absolute;
left: 0px; top: 600px; z-index: 4;
width: 100%; height: 60px;
background-color: yellow;
border: none;
}
-->
</style>
</head>
<body>
<div id="kopf">Kopfzeile</div>
<div id="navigation">
```

```
<p>Navigation</p>
</div>
<div id="mitte">
<div id="mitte_inhalt">Inhalt</div>
</div>
<div id="fuss">Fußzeile</div>
</body>
</html>
```

Tabellen und Inhalte

Sie können die gesamte Tabelle oder den Inhalt einzelner Zellen verschieden ausrichten – und das sowohl horizontal als auch vertikal.

Tabelle horizontal ausrichten

Um eine **Tabelle** auszurichten, kommt das bereits gut bekannte Attribut `align` zum Einsatz. Zur Erinnerung: Mit diesem Attribut können Sie auch Textabsätze und Überschriften ausrichten. Dabei entspricht das Vorgehen bei Tabellen
dem bei Absätzen und Überschriften. Im Start-Tag des `table`-Elements notieren Sie das Attribut `align` und weisen ihm einen von drei Parametern zu.

Parameter	Bedeutung
`left`	Richtet die Tabelle am linken Fensterrand aus.
`right`	Richtet die Tabelle am rechten Fensterrand aus.
`center`	Richtet die Tabelle zentriert aus.

```
<table align="left">   <!-- richtet Tabelle links aus -->
<table align="right">  <!-- richtet Tabelle rechts aus -->
<table align="center"> <!-- zentriert Tabelle -->
```

Info Beachten Sie, dass Sie mit dem `align`-Attribut im Start-Tag des `table`-Elements die Tabelle ausrichten und nicht den Inhalt der Zellen.

Zelleninhalt horizontal ausrichten

Auch an dieser Stelle kommt das `align`-Attribut wieder zum Einsatz. Diesmal wird das Attribut aber im Start-Tag der **Tabellenzelle** (`td` oder `th`) notiert und mit einem der drei Parameter `left`,

`center` oder `right` versehen. Die Anweisungen für Kopfzellen lauten:

```
<th align="left">   <!-- richtet Inhalt links aus -->
<th align="right">  <!-- richtet Inhalt rechts aus -->
<th align="center"> <!-- zentriert Inhalt -->
```

Für Datenzellen lauten sie:

```
<td align="left">   <!-- richtet Inhalt links aus -->
<td align="right">  <!-- richtet Inhalt rechts aus -->
<td align="center"> <!-- zentriert Inhalt -->
```

Zelleninhalt vertikal ausrichten

Um den Inhalt einer Zelle vertikal auszurichten, wird ein neues, bisher unbekanntes Attribut verwendet: `valign` (engl. vertical align, dt. vertikale Ausrichtung). Dieses Attribut besitzt vier neue Parameter.

Parameter	Übersetzung	Bedeutung
`top`	oben	Richtet den Zelleninhalt obenbündig, also am oberen Zellenrand aus.
`middle`	mittig	Richtet den Zelleninhalt mittig aus.
`bottom`	unten	Richtet den Zelleninhalt untenbündig, also am unteren Rand der Zelle aus.
`baseline`	Grundlinie	Richtet den Zelleninhalt an der Grundlinie aus, so dass die erste Textzeile einer Zeile immer auf gleicher Höhe liegt.

Die Anweisungen für die vertikale Ausrichtung lauten dann für Kopfzellen:

```
<th valign="top">      <!-- Inhalt oben -->
<th valign="middle">   <!-- Inhalt mittig -->
<th valign="bottom">   <!-- Inhalt unten -->
<th valign="baseline"> <!-- Inhalt an Grundlinie -->
```

und für Datenzellen:

```
<td valign="top">      <!-- Inhalt oben -->
```

```
<td valign="middle">   <!-- Inhalt mittig -->
<td valign="bottom">   <!-- Inhalt unten -->
<td valign="baseline"> <!-- Inhalt an Grundlinie -->
```

Ausrichtung an Dezimalzeichen

Seit der HTML-Version 4.0 ist es auch möglich, den **Zelleninhalt** an einem bestimmten Zeichen auszurichten. Jedoch wird diese Funktion von bisher keinem Browser unterstützt. Sie soll aber an dieser Stelle trotzdem aufgeführt werden, da
sie besonders bei Zahlen mit Dezimalwerten interessant ist und kommende Browserversionen diese Funktion vielleicht unterstützen werden. So lassen sich Zelleninhalte beispielsweise am Dezimalzeichen (in Deutschland das Kommazeichen) ausrichten. Dazu werden
dem Attribut `align` der Parameter `char` und dem zusätzlichen Attribut `char` das entsprechende Zeichen übergeben.

```
<col align="char" char=",">
```

Das `col`-Element ist Ihnen an dieser Stelle sicherlich noch unbekannt, ich werde es Ihnen an geeigneter Stelle jedoch noch genauer erklären. Der Code demonstriert sehr schön, wie sich die Attribute `align` und `valign`zum Ausrichten von Inhalten einer Tabellenzelle nutzen lassen.

Tabellenbeschriftung und Tabellenausrichtung

Es gibt verschiedene Möglichkeiten, **Tabelleninhalte** logisch zu unterteilen und in unterschiedlichen Gruppen zusammenzufassen. So haben Sie die Möglichkeit, die Anzahl der Spalten vorzudefinieren oder eine Tabelle in einen Kopf-, Körper-
und Fußteil zu unterteilen.

Spaltengruppen

Für das Vordefinieren von Spalten werden zwei neue Elemente benötigt: `colgroup` (engl. column group, dt. Spaltengruppe) und `col` (engl. column, dt. Spalte). Es gibt jedoch zwei unterschiedliche Möglichkeiten, Spaltengruppen zu
definieren. In der ersten Variante notieren Sie das Element `colgroup` und in dessen Gültigkeitsbereich so viele Elemente des Typs `col`, wie Spalten in der Tabelle vorhanden sind. Anstatt jeder einzelnen Zelle einer Spalte die Breite
explizit zuweisen zu müssen, können Sie nun die `col`-Elemente einsetzen. Soll beispielsweise eine Tabelle vier Spalten mit den Breiten 15 %, 20 %, 35 % und 30 % enthalten, würde die entsprechende Definition so aussehen:

```
<colgroup>
<col width="15%">
<col width="20%">
```

```
<col width="35%">
<col width="30%">
</colgroup>
```

Wie Sie sehen können, wurde für das `col`-Element kein Ende-Tag notiert. Die Angabe von `</col>` ist auch nicht erlaubt bzw. vorgesehen. Natürlich können Sie für die Breite auch Pixelangaben machen. Es gibt aber noch eine dritte
Möglichkeit: relative Breitenangaben. So können Sie einer Spalte eine Breite von 5 Anteilen der Gesamttabelle zuweisen. Hinter der Zahl muss lediglich ein Sternchen notiert werden.

```
<colgroup>
<col width="1*">
<col width="3*">
<col width="5*">
<col width="7*">
</colgroup>
```

Die Gesamtbreite der **Tabelle** beträgt 16/16. Die erste Spalte ist 1/16, die zweite 3/16, die dritte 5/16 und die vierte 7/16 breit. Dies kommt aber erst dann zur Geltung, wenn die Gesamtbreite der Tabelle im Start-Tag von `table`mittels `width` angegeben wurde. Möchten Sie, dass alle Spalten die gleiche Breite besitzen, können Sie auf die `col`-Elemente verzichten und müssen nur das `colgroup`-Element verwenden.

```
<colgroup span="4" width="150">
</colgroup>
```

Das Attribut `span` definiert dabei die Anzahl der Spalten (insgesamt also vier), und das Attribut `width` definiert die jeweilige Spaltenbreite von 150 Pixel. Natürlich können Sie auch hier prozentuale oder anteilige Angaben machen.

Logische Gruppierung

Wie bereits erwähnt, können Sie eine Tabelle in einen Kopf-, einen Körper- und einen Fußteil unterteilen. Die entsprechenden Elemente lauten:

- `thead` (engl. table head, dt. Tabellenkopf) definiert den Kopfteil der Tabelle.
- `tfoot` (engl. table foot, dt. Tabellenfuß) definiert den Fußteil der Tabelle.
- `tbody` (engl. table body, dt. Tabellenkörper) definiert den Körperteil der Tabelle.

Die Reihenfolge dieser Elemente ist explizit festgelegt und lautet `thead > tfoot > tbody`. Die entsprechenden Inhalte, die den einzelnen Gruppen zugewiesen werden sollen, müssen im Gültigkeitsbereich der einzelnen Elemente

(also zwischen Start- und Ende-Tag) notiert werden. Besondere Beachtung muss in diesem Beispiel das Attribut `rules` im `table`-Start-Tag finden. Für die Verwendung dieses Attributs muss außerdem das Attribut `borderim` Start-Tag von `table` notiert werden. Mit `rules` können Sie die Darstellung des **Tabellenrahmens** beeinflussen. Dafür stehen Ihnen fünf verschiedene Attribute zur Verfügung.

Parameter	Erklärung
`none`	Verhindert die Darstellung der inneren Linien. Der Außenrahmen wird jedoch dargestellt.
`all`	Zeigt alle Linien an, sowohl innen als auch außen.
`cols`	Zeigt die Linien zwischen den Spalten, aber nicht zwischen den Zeilen an.
`rows`	Zeigt die Linien zwischen den Reihen, aber nicht zwischen den Spalten an.
`groups`	Zeigt Linien zwischen den einzelnen Tabellengruppen an.

Im Code wurde der Parameter `groups` für `rules` verwendet. Dadurch werden die Linien des Rahmens in Abbildung nur horizontal zwischen den Gruppen dargestellt.

Tabellenbreite und Spalten vordefinieren

Wenn umfangreiche Tabellen über eine schwache Netzverbindung übertragen werden müssen, hat der Browser das Problem, dass er mit der Darstellung der Tabelle am Bildschirm warten muss, bis die gesamte Tabelle übertragen ist, da er vorher nicht weiß, welche
Darstellungsbreite er für die Tabelle reservieren muss. Tabellen werden nämlich per Voreinstellung so breit dargestellt, wie es ihr Inhalt verlangt. Das Gleiche gilt für die einzelnen Spalten der Tabelle. Jede Spalte wird so breit, wie es ihr breitester,
nicht umzubrechender Inhalt verlangt. Um dem Browser bei diesem Problem zu helfen, können Sie in HTML die Breiten von Tabelle und Spalten vordefinieren. Außerdem können Sie Spalten in Spaltengruppen aufteilen und das Breitenverhältnis der Spalten untereinander
bestimmen. Die gewünschte Gesamtbreite der Tabelle können Sie durch das `width`-Attribut im einleitenden `<table>`-Tag angeben. Ein Beispiel:

```
<table border="1" width="100%">
```

Info Erlaubt sind als Wertzuweisung reine Zahlen, die als Pixel interpretiert werden, oder Prozentangaben, wie im Beispiel gezeigt. 100% bedeutet: Die Tabelle soll die gesamte verfügbare Breite einnehmen. Bei einer solchen Angabe dehnt der Browser

die Spalten nach Gutdünken aus. Wenn Sie mehr Kontrolle über die Spaltenbreiten haben möchten, ist es besser, zusätzlich zur `width`-Angabe im `<table>`-Tag Spalten vorzudefinieren und ihnen eine Breite zuzuweisen. Dazu das
Beispiel einer kompletten kleinen Tabelle:

Info Um Spalten vorzudefinieren, notieren Sie unterhalb des einleitenden `<table>`-Tags ein Konstrukt, das durch `<colgroup>` eingeleitet und mit `</colgroup>` beendet wird. Innerhalb davon können
Sie für jede Spalte jeweils ein alleinstehendes `<col>`-Tag definieren. Mit dem Attribut `width=` definieren Sie die Spaltenbreite. Auch hierbei sind absolute Zahlen für Pixel und Prozentangaben erlaubt. Als Besonderheit ist
jedoch auch eine weitere Notationsform zulässig, wie das Beispiel zeigt. Durch Angabe einer Zahl mit Sternzeichen dahinter können Sie das logische Breitenverhältnis der Spalten untereinander festlegen. Im obigen Beispiel nimmt die erste Spalte 2/9 Breite
der Gesamttabellenbreite ein, die zweite Spalte 4/9 und die dritte Spalte 3/9 (also 1/3).

> Wie Ihnen vielleicht aufgefallen ist,

haben wir in unseren Beispielen ausschließlich relative Angaben verwendet. Relative Angaben sollten Sie gegenüber absoluten Pixelangaben bevorzugen. Zu absoluten Angaben sollten Sie auch wissen, dass Browser solche Angaben nicht zwingend umsetzen. Jede
Tabelle oder Spalte wird so breit, wie es ihr breitester Inhalt verlangt. Zu geringe Breitenangaben werden dabei ignoriert. Die Möglichkeit, überbreite Inhalte abzuschneiden oder Spaltenbreiten wirklich festzuzementieren, besteht nur mithilfe von **CSS**.

Allgemeines zu Tabellen für Webseitenlayouts

Wenn Sie bei Webseiten auf sauber ausgerichtete, aber relativ frei verteilte Elemente mit meistfarbigen Flächen stoßen, wurde in vielen Fällen mit der Technik der „blinden Tabellen“ gearbeitet. Blinde **Tabellen** haben keine sichtbaren Gitternetzlinien.
Dadurch merkt der Betrachter gar nicht, dass es sich in Wirklichkeit um eine Tabelle handelt.

Einen Nachteil sollten Sie jedoch beachten, wenn Sie beabsichtigen, den gesamten anzuzeigenden Inhalt einer Webseite mit Hilfe einer blinden Tabelle anzuordnen: Tabellen werden vom Webbrowser erst angezeigt, nachdem ihr kompletter Inhalt eingelesen wurde
bzw. erst in dem Augenblick, wo der Webbrowser genau weiß, wie groß und wie breit die Tabelle angezeigt werden muss. Das bedeutet bei der Datenübertragung aus dem Web, dass ein Anwender am Bildschirm länger wartet, bis überhaupt etwas angezeigt wird –
es sei denn, Sie benutzen die Möglichkeit, Spalten vorzudefinieren.

> Ein weiteres Problem besteht darin, dass Browser mit gewünschten Breiten- und Höhenangaben nicht unbedingt so umgehen, wie Sie es wünschen.

Inhalte werden dann plötzlich gestaucht, auseinandergezogen usw. Dies hat zur Erfindung des so genannten „blinden Pixels“ geführt, einer kleinen GIF-Grafik, die nur aus einer einfarbigen Fläche besteht, deren Farbe wiederum als transparent definiert ist.
Zusammen mit der Möglichkeit, Breite und Höhe einer Grafik in **HTML** unabhängig von der tatsächlichen Größe der Grafik anzugeben, lässt sich eine solche Grafik scheinbar unsichtbar und unauffällig einbauen. So lassen sich vor allem feste
Mindestbreiten für einzelne Spalten einer Tabelle erzwingen, und der Browser kann die Inhalte darin nicht mehr stauchen. Diese Technik gilt einerseits als unsaubere Trickserei, ist aber andererseits Ausdruck für fehlende Formatiermöglichkeiten. Mittlerweile
stellen Stylesheets solche Möglichkeiten zwar mit `min-width` bereit, doch das nutzt nicht viel, solange die weit verbreiteten Browser noch keine Unterstützung dafür anbieten. In diesem Abschnitt wird anhand eines Beispiels gezeigt, wie Seitenlayouts
mit blinden Tabellen im Prinzip funktionieren. Auf die Technik des blinden Pixels wird hier verzichtet. Doch eine andere, typische Technik kommt zum Einsatz: das Verschachteln von Tabellen. Denn meistens lassen sich saubere Tabellenlayouts nur erreichen,
wenn man mit Tabellen innerhalb von Tabellen arbeitet.

Beispiel eines typischen Tabellen-Layouts

Das folgende Beispiel zeigt ein Layout, wie man es auf vielen Webseiten findet: oben eine farbige Fläche, wo die Seitenüberschrift steht und andere Elemente wie Grafiken für Logos, Banner oder dergleichen Platz haben, unterhalb davon links eine gleichfarbige,
direkt angeschlossene, schmale Fläche für Navigationsverweise innerhalb des Webprojekts, und rechts davon eine breite Fläche für den eigentlichen Inhalt der Seite. Das Beispiel ist gültiges HTML 4.0, benutzt aber diverse Attribute, die als deprecated
gekennzeichnet sind, weshalb als HTML-Variante „*Transitional*“ gewählt werden muss.

Info Das Beispiel bestimmt im einleitenden `<body>`-Tag zunächst dateiweite Farben fur Hintergrund, Text und Verweise. Innerhalb des Dateikörpers wird die übergeordnete Tabelle definiert. Sie erhält die Attribute `border="0"`,
damit kein Rahmen und keine Gitternetzlinien angezeigt werden, sowie `cellpadding="0"` und `cellspacing="0"`, damit sich die einzelnen Zellen der Tabelle nahtlos aneinander fugen. Das ist wichtig, damit zwischen den Farbflächen keine
sichtbaren Lücken entstehen. Weil es sich um eine die ganze Seite überspannende Tabelle handelt, werden für Browser, die das verstehen, die gewünschten Spalten mit `<col>`-Tags vordefiniert. Die Tabelle soll zwei Spalten haben, wobei
die erste eine Breite von 200 Pixeln haben soll. Das soll die Spalte fur Navigationsverweise werden. Die zweite Spalte soll den Rest des verfügbaren Platzes in Anspruch nehmen. Die gesamte Tabelle soll sich über die volle verfügbare Breite erstrecken
– angegeben durch `width="100%"` im einleitenden `<table>`-Tag der übergeordneten Tabelle.

Die gesamte übergeordnete Tabelle bekommt ferner mit Hilfe von

bqcolor="#A050A0" einen matt-violetten Hintergrund zugewiesen. Damit erhalten erst mal alle ihre Inhalte diese Hintergrundfarbe. Um andersfarbige Flächen innerhalb der Tabelle zu erzeugen, sind bei den einzelnen Zellen weitere Angaben zur
Hintergrundfarbe erforderlich, die dann diese globale Hintergrundfarbe überschreiben. Die erste Zelle der ersten Zeile der übergeordneten Tabelle ist zugleich die einzige Zelle in dieser Zeile, da durch das Attribut colspan="2"erreicht wird, dass sich die Zelle über zwei Spalten erstreckt. Der Inhalt dieser Zelle steht nun fur den überschriftenteil zur Verfugung. Da mit cellpadding="0" und cellspacing="0" gearbeitet wird, würde der Inhalt allerdings
ganz am Außenrand der Tabelle kleben, was unsauber aussieht. Deshalb kommt bereits bei der überschrift zum ersten Mal die Technik der Tabelle in der Tabelle zum Einsatz: <table border="0" cellpadding="10" cellspacing="0"> definiert
innerhalb der über zwei Spalten gestreckten Tabellenzelle eine weitere Tabelle. Auch diese ist wiederum rahmenlos <border="0">. Durch cellpadding="10" wird jedoch ein Innenabstand von l0 Pixeln zum Zellenrand erzwungen.
Diese Angabe ist der eigentliche Grund, warum die innere Tabelle definiert wird. Denn die innere Tabelle hat nur eine einzige Zeile mit einer einzigen Spalte, also nur eine Zelle, und darin ist die h1-überschrift für die Seite notiert („Willkommen“).
Die überschrift erhält mit Hilfe der CSS-Eigenschaft color= eine weiße Farbe, um sich ordentlich von dem violetten Hintergrund der Tabelle abzuheben.

> In der zweiten Zeile der übergeordneten Tabelle werden fur beide definierten Spalten je eine Zelle definiert.

Die erste (linke) Zelle ist dann der Behälter für die gesamte linke Spalte – also fur die Spalte, in der die Navigationsverweise stehen. Die zweite (rechte) Zelle ist der Behälter für den gesamten Inhalt des Hauptbereichs, also des Bereichs für den eigentlichen
Seiteninhalt. Die zweite Zelle, also die für den Hauptbereich, erhält mit bgcolor="#FFFFFF" als Hintergrundfarbe weiß zugewiesen. So entsteht letztlich der optische Eindruck, als ob die Seite nur oben und links einen farbigen Balken enthalten
würde, während der gesamte Rest der Seite einen weißen Hintergrund hat. In beiden Zellen der zweiten Zeile der übergeordneten Tabelle wird auch wieder je eine innere Tabelle definiert, die mit den gleichen Attributen wie im Fall der überschrift für saubere
Abstände zum Rand sorgt. So werden auch alle Inhalte ordentlich auf einer Höhe ausgerichtet. Mit valign="top" wird in beiden Zellen dafür gesorgt, dass die Inhalte, die ja unterschiedlich groß sein können, auf jeden Fall obenbündig ausgerichtet
werden. Die Verweise in der linken Spalte heben sich durch ihre dateiweit definierten Farben gut von dem violetten Hintergrund ab. Normaler Text, der in der linken Spalte steht, also überschriften, Textabsätze usw., erhalten im Beispiel wieder mit Hilfe
von **CSS** eine weiße Textfarbe, um sich vom Hintergrund abzuheben.
Info Sie können Stylesheets noch wesentlich konsequenter einsetzen, als es im obigen Beispiel der Fall ist. Maßgeblich sind im hier beschriebenen Zusammenhang z.B. folgende CSS-Eigenschaften:
Schriftformatierung, Ausrichtung und Absatzkontrolle,
Außenrand und Abstand, Innenabstand, Rahmen, Hintergrundfarben und -bilder, Positionierung und

Anzeige von Elementen.

Zellen verbinden

Eine weitere Möglichkeit für **Tabellen** in HTML ist das Verbinden von Zellen. So lassen sich neben- oder übereinander liegende Zellen zu einer Zelle zusammenschließen und über mehrere Spalten oder Zeilen ausdehnen.

Um nebeneinander liegende Zellen miteinander verbinden zu können, wird das Attribut `colspan` (engl. column span, dt. Spaltenspanne) verwendet. Als Parameter wird dem Attribut die Anzahl der Spalten übergeben, über die sich die Zelle erstrecken
soll. Gehen Sie einmal davon aus, dass Sie eine Tabelle mit zwei Zeilen und vier Spalten haben. Die Anzahl der Zellen in der ersten Zeile soll gleich bleiben (insgesamt vier), aber in der zweiten Zeile sollen nur zwei Spalten zu sehen sein. Als Skizze
würde das so aussehen: Zuerst müssen Sie die Tabelle einleiten und die erste Zeile mit vier Zellen definieren.

```
<table border="1">
<tr>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
</tr>
```

Nun folgt die zweite Zeile, die ganz normal eingeleitet wird. Da Sie nur zwei Spalten in der zweiten Zeile haben wollen, müssen Sie sich zuerst überlegen, welche der Zellen Sie miteinander verbinden möchten. Laut Abbildung 1 sollen die Zellen 1 und 2
und die Zellen 3 und 4 miteinander verbunden werden. Sie definieren nun die erste Zelle und weisen ihr mittels `colspan="2"` eine Spannweite von zwei Zellen zu.

```
<tr>
<td colspan="2">Zelle 1 und 2</td>
```

Bildlich vorgestellt haben Sie eben zwei Zellen erstellt. Das ist wichtig, da Sie die zweite Zelle sozusagen überspringen. Nun folgt die nächste (dritte) Zelle, die sich über zwei Spalten erstrecken soll. Danach beenden Sie die Zeilen- und **Tabellendefinition**.

```
<td colspan="2">Zelle 3 und 4</td>
</tr>
</table>
```

Sie behalten also die standardmäßige Vorgehensweise beim Erzeugen von Tabellen bei (von links nach rechts und von oben nach unten). Prinzipiell wäre es auch möglich, alle vier Zellen der zweiten Zeile zu verbinden. Sie müssten lediglich die Anzahl der
Spalten, über die sich die erste Zelle erstrecken soll, auf vier erhöhen (`colspan="4"`). Genauso gut ist es möglich, die zweite Zelle auf eine Spannweite von drei zu erhöhen. Sie definieren dann zuerst ganz normal die erste Zelle und weisen
bei der Definition der zweiten Zelle eine Spannweite von drei zu (`colspan="3"`). Alle drei Varianten werden im nachfolgenden Code berücksichtigt und dargestellt.

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<!-- Zellenverbund //-->
<html>
<head>
<title>Zellenverbund</title>
</head>
<body>
<table border="1" width="50%">
<tr>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td colspan="2">Zelle 1 und 2</td>
<td colspan="2">Zelle 3 und 4</td>
</tr>
</table>
<br>
<table border="1" width="50%">
<tr>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td colspan="4">Zelle 1, 2, 3 und 4</td>
</tr>
</table>
<br>
<table border="1" width="50%">
<tr>
<td>1</td>
<td>2</td>
```

```
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>Zelle 1</td>
<td colspan="3">Zelle 2, 3 und 4</td>
</tr>
</table>
</body>
</html>
```

Um Zellen, die übereinander liegen, verbinden zu können, verwenden Sie das `rowspan`-Attribut (engl. row spanning, dt. Zeilenspanne). Es verhält sich äquivalent zum `colspan`-Attribut mit dem Unterschied, dass es einen Zellenverbund
über mehrere Zeilen erzeugt. Die Anzahl der zu verbindenden Zellen bzw. Zeilen wird dem Attribut `rowspan` als Parameter übergeben. Als Beispiel verwenden Sie eine Tabelle, die über jeweils vier Zeilen und Spalten verfügt. Wenn Sie nun die
Zellen 5 und 9 (siehe Abbildung 3) miteinander verbinden möchten, müssen Sie zuerst den **Tabellenbeginn** einleiten und die erste Zeile definieren:

```
<table border="1" width="100%">
<tr>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
</tr>
```

Als Nächstes folgt die zweite Zeile. Deren erste Zelle ist die Zelle 5. Da diese mit der Zelle 9 verbunden werden soll, weisen Sie ihr das Attribut `rowspan` mit dem Parameter 2 zu (die verbundene Zelle erstreckt sich über zwei Zeilen). Die
Zellen 6, 7 und 8 werden ganz normal definiert.

```
<tr>
<td rowspan="2">5 und 9</td>
<td>6</td>
<td>7</td>
<td>8</td>
</tr>
```

In der dritten Zeile müsste es rein theoretisch vier Zellen zu definieren geben. Die Zelle 9 ist jedoch schon durch das `rowspan`-Attribut der Zelle 5 definiert worden, sodass lediglich die restlichen drei Zellen fehlen.

Sie beginnen mit der
Zellendefinition der Zeile 3, also bei Zelle 10. Die vierte Zeile notieren Sie ganz normal und schließen danach die Tabellendefinition ab.

```
<tr>
<td>10</td>
<td>11</td>
<td>12</td>
</tr>
<tr>
<td>13</td>
<td>14</td>
<td>15</td>
<td>16</td>
</tr>
</table>
```

Info Sie gehen also wieder streng nach dem Schema „von links nach rechts, von oben nach unten“ vor. Natürlich ist es auch durchaus möglich, mehrere Zellen übereinander zu verbinden. Dazu werden Sie nun die Tabelle aus Abbildung 4 in **HTML**umsetzen. Die Zellen, die miteinander verbunden werden müssen, sind 1-5-9, 11-15 und 4-8-12-16. Wie üblich leiten Sie die Tabellen- und anschließend die Zeilendefinition ein. Da die Zelle 1 mit Zelle 5 und 9 verbunden werden soll, müssen Sie ihr als Parameter
für rowspan den Wert 3 übergeben. Danach notieren Sie jeweils die Zellen 2 und 3. Die Zelle 4 soll mit 8, 12 und 16 verbunden werden, also den Parameter 4 für das `rowspan`-Attribut notieren.

```
<table border="1" width="100%">
<tr>
<td rowspan="3">1, 5, 9</td>
<td>2</td>
<td>3</td>
<td rowspan="4">4, 8, 12, 16</td>
</tr>
```

Es folgt die zweite Zeile. Die Zellen 5 und 8 sind bereits durch die erste Zeile definiert.

```
<tr>
<td>6</td>
<td>7</td>
</tr>
```

In der dritten Zeile müssen Sie lediglich die Zellen 10 und 11 definieren. Die Zellen 9 und 12 wurden

ebenfalls in der ersten Zeile definiert.

```
<tr>
<td>10</td>
<td rowspan="2">11, 15</td>
</tr>
```

Zum Schluss müssen Sie noch die vierte Zeile und die Zellen 13 und 14 notieren, da 15 und 16 bereits in der dritten bzw. ersten Zeile notiert worden sind. Danach schließen Sie die Tabellendefinition ab.

```
<tr>
<td>13</td>
<td>14</td>
</tr>
</table>
```

Die beiden Beispiele würden in einem HTML-Dokument wie folgt aussehen. Abbildung 5 zeigt die Darstellung im Internet Explorer 6.0.

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<!-- Zellenverbund -->
<html>
<head>
<title>Zellenverbund</title>
</head>
<body>
<h1>Tabelle aus Abbildung 3 mit Verbund von 5 und 9</h1>
<table border="1" width="100%">
<tr>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td rowspan="2">5 und 9</td>
<td>6</td>
<td>7</td>
<td>8</td>
</tr>
<tr>
<td>10</td>
```

```
<td>11</td>
<td>12</td>
</tr>
<tr>
<td>13</td>
<td>14</td>
<td>15</td>
<td>16</td>
</tr>
</table>
<h1>Tabelle aus Abbildung 4</h1>
<table border="1" width="100%">
<tr>
<td rowspan="3">1, 5, 9</td>
<td>2</td>
<td>3</td>
<td rowspan="4">4, 8, 12, 16</td>
</tr>
<tr>
<td>6</td>
<td>7</td>
</tr>
<tr>
<td>10</td>
<td rowspan="2">11, 15</td>
</tr>
<tr>
<td>13</td>
<td>14</td>
</tr>
</table>
</body>
</html>
```

Über- und nebeneinander liegende Zellen verbinden

Das Verbinden von Zellen über mehrere Zeilen und Spalten hinweg erfolgt nur unwesentlich anders als mit den bisher behandelten Methoden. Sie weisen einer Zelle einfach beide Attribute (`colspan` und `rowspan`) zu. Abbildung 6 zeigt
das Schema einer Tabelle mit vier Spalten und vier Zeilen. Aufgrund dieses Schemas sollen nun die Zellen 6, 7, 10, 11, 14 und 15 miteinander verbunden werden. Wie auch bisher leiten Sie die **Tabellendefinition** ein und definieren die erste
Zeile mit den Zellen 1, 2, 3 und 4. Anschließend folgt die zweite Reihe. Hier definieren Sie ganz normal die Zelle 5. Bei der Definition der Zelle 6 übergeben Sie die Attribute und Parameter `colspan="2"` und `rowspan="3"`. Dies
bedeutet, dass sich die Zelle über zwei Spalten und drei Zeilen erstrecken soll. Anschließend folgt die Zelle

8. In der Reihe 3 definieren Sie lediglich die Zellen 9 und 12 und in Reihe 4 die Zellen 13 und 16. Dann beenden Sie die Tabellendefinition.
Das vollständige Beispiel-Code und die Abbildung sehen folgendermaßen aus:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<!-- Zellenverbund -->
<html>
<head>
<title>Zellenverbund</title>
</head>
<body>
<h1>Zellenverbund über mehrere Reihen und Spalten</h1>
<table border="1" width="100%">
<tr>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>5</td>
<td colspan="2" rowspan="3">6, 7, 10, 11, 14 und 5</td>
<td>8</td>
</tr>
<tr>
<td>9</td>
<td>12</td>
</tr>
<tr>
<td>13</td>
<td>16</td>
</tr>
</table>
</body>
</html>
```

Info Aufgrund der Beispiele in diesem Artikel sollten Sie also bei komplexen **Tabellenstrukturen** vorher immer eine Zeichnung anlegen oder sich zumindest ganz genau überlegen, wie die Tabelle später aussehen soll. Dadurch können
Sie eine Menge Zeit sparen.