

LF2 - Webseiten erstellen

2 CSS

2.1 CSS-Grundlagen und Box-Modell

Stundenempfehlung

FIAE	FISI	ITSK	IFK	ITSE
5	5	5	5	5

[2.1.1 Überblick über dieses Kapitel](#)

[2.1.2 Einführung in CSS](#)

[2.1.3 Syntax eines Styles](#)

[2.1.4 Stylesheets in HTML einbinden](#)

[2.1.5 Maßeinheiten und Farbangaben](#)

[2.1.6 Inline- und Block-Elemente, <div>- und -Tag](#)

[2.1.7 Box-Modell](#)

[2.1.8 CSS-Klassen und CSS-IDs](#)

[2.1.9 Mehrfach-Formatdefinitionen und Formatdefinitionen verschachtelter HTML](#)

http://www.daa-mws-virtuell.de/content/7/html3/it/it06_2011/IT.VI.a/02/ueb_it0602a_01.html

[2.1.9 Elemente](#)

[2.1.10 Kapitelschwerpunkte](#)

2.1.1 Überblick über dieses Kapitel

CSS bedeutet Cascading Stylesheets (wörtlich etwa kaskadierende Stilblätter). CSS ermöglicht die Definition von Formatierungen für HTML-Elemente (HTML-Tags).

Der wesentliche Vorteil der Formatierung von HTML-Seiten durch CSS gegenüber der Formatierung mit HTML-Attributen besteht darin, dass die Formatierung vom Inhalt unabhängig ist und zentral festgelegt werden kann. Dazu kommt, dass CSS wesentlich mehr Formatierungsmöglichkeiten bietet als HTML.

In diesem Kapitel lernen Sie den Aufbau (die Syntax) von CSS-Formatierungen kennen. Sie werden erste Formatierungen in CSS erstellen, die Grundkonzepte für HTML-Block- und -Inline-Elemente sowie das Box-Modell kennenlernen.

Weiterhin wird gezeigt, wie CSS-Formatierungen direkt einem HTML-Tag sowie indirekt über eine CSS-Klasse oder einer CSS-ID zugewiesen werden kann.

Weiterführende Informationen: [SelfHTML -> Stylesheets und HTML](#)

CSS

Cascading Stylesheets (wörtlich etwa kaskadierende Stilblätter)

Sprache zur Definition von Formateigenschaften einzelner HTML-Elemente (Quelle: [SelfHTML](#))

Vorteil der Formatierung der HTML-Seiten durch CSS

Viele HTML-Formatierungen gelten als "missbilligt". Sie sollen nicht mehr eingesetzt werden.

Praktisch wird angestrebt, dass HTML nur noch zur Beschreibung der Inhalte genutzt wird. Formatierungen sollen nicht mit Hilfe von HTML-Elementen durchgeführt werden.

Die gesamte Formatierung (das Design, Layout) soll durch CSS erfolgen.

Damit wird der Inhalt (HTML) von der Form (CSS) getrennt. Das führt dazu, dass die selbe HTML-Seite nur durch unterschiedle CSS-Formatierungen völlig unterschiedlich dargestellt werden kann.

Die Website <http://www.mezzoblue.com/zengarden/alldesigns/> liefert Beispiele, die dieselbe HTML-Seite im unterschiedlichsten Design darstellen.

CSS-Sprachversionen

CSS-Version	
1.0	Elemente zur Formatierung
2.0	Elemente zur Positionierung
2.1	
3.0	<p>In der Entwicklung, bringt weitere CSS-Elemente und unterstützt neue HTML5Elemente</p> <p>Einen kleinen Ausblick auf CSS 3 zeigt http://www.webmasterpro.de/coding/article/css3.html. Viele der neuen Elemente werden aber noch nicht durch die Browser unterstützt.</p>

Browser-Abhängigkeit der Stylesheet-Verarbeitung

Sie finden auf SelfHTML einen Abschnitt über die [Browserabhängigkeit](#). Der wurde aber 2007 geschrieben. Mittlerweile ist der Stand der, dass alle gängigen Browser in der aktuellen Version CSS 1 und CSS 2 gut unterstützen.

Im Rahmen dieses CSS-Moduls können nicht alle Feinheiten und Unterschiede der verschiedenen Browser behandeln werden. Sie haben aber die Möglichkeit, ihre Webseiten in verschiedenen Browser und auf unterschiedlichen Betriebssystemen aufzurufen und damit Übereinstimmungen sowie Unterschiede in der Darstellung zu erkennen.

Hinweis

Zur Prüfung, ob ein CSS-Code fehlerfrei ist, kann der [CSS-Validator des W3C](#) verwendet werden.

2.1.3 Syntax eines Styles

Nach dieser kurzen Einführung soll es jetzt konkret werden. In diesem Abschnitt wird die Syntax eines Styles behandelt.

Weiterführende Informationen: [SelfHTML -> Stylesheets und HTML -> CSS-Formate](#)

Syntax einer Style-Definition

```
Selektor { Eigenschaft:Wert; ... }
```

Begriffe

Selektor Legt die HTML-Elemente fest, auf die der Style angewendet werden soll.
Das kann z.B. ein **HTML-Element** wie <p>, <td> oder sein. Mit Hilfe der CSS-Klassen bzw. CSS-IDs können einzelne oder mehrere HTMLTags ausgewählt werden.

Deklaration Als Deklaration wird ein Eigenschaft: Wert-Paar bezeichnet.

Eigenschaft Formatierungseigenschaft, die festgelegt werden soll.

Beispiele sind:

```
color font-size
background-color
```

Wert Wert, der der betreffenden Eigenschaft zugewiesen werden soll.

Beispiele sind:

```
color:#FF0000; font-size: 14px;
background-color:#FFFFFF;
```

Beispiele für vollständige Styles:

```
p {font-size: 14px; color: #000000; background-color: #FFFFFF; } h1
{font-size: 20px; color: #FF0000; background-color: #FFFFFF; }
```

Der erste Styles formatiert alle <p>-Tags mit der Schriftgröße 14 Pixel, der Zeichenfarbe Schwarz und der Hintergrundfarbe des Absatzes Weiß.

Der zweite Styles formatiert alle <h1>-Tags.

Kommentare innerhalb der Stylesheet-Definitionen

Kommentare können eingesetzt werden, um zu den Styles Kommentare mit ergänzenden Informationen zu erstellen oder um einzelne Styles bzw. einzelne Deklarationen unwirksam zu machen, ohne sie zu löschen.

Syntax eines CSS-Kommentars

```
/* CSS-Kommentar */
```

Beispiel

```
/* Absatz-Formate */ p {font-size: 14px; color:
#000000; /*background-color:
#FFFFFF;*/ } h1 {font-size: 20px; color: #FF0000; background-
color: #FFFFFF; }
```

Der Kommentar über den Styles erläutert die Art der darauf folgenden Styles, hier "Absatz-Formate". Der Kommentar innerhalb des oberen Styles des <p>-Tag kommentiert die Deklaration für die Hintergrundfarbe in dem Style aus.

2.1.4 Stylesheets in HTML einbinden

Nachdem Sie den Aufbau der Stylesheets kennen gelernt haben, ist als nächstes zu klären, welche Möglichkeiten es gibt, die Stylesheets mit den HTML-Seiten bzw. den HTML-Elementen zu verbinden.

(1.1) Möglichkeiten des Einbindens der Stylesheets

Aufgabe

In diesem Abschnitt werden Beispiele angegeben, die nachvollzogen werden können. Testen Sie die Beispiele in HTML-Dateien.

Weiterführende Informationen: [SelfHTML -> Stylesheets einbinden](#)

Dazu bestehen drei Möglichkeiten

- Einbinden eines Styles als Attribut eines HTML-Element
- Einbinden von Styles im Header einer HTML-Datei
- Auslagerung der Stylesheets in eine CSS-Datei und Inkludieren dieser Datei in eine oder mehrere HTML-Dateien

Einbinden eines Styles als Attribut eines HTML-Element

```
<p style="font-style:italic; font-weight:bold">Formatierter Absatz</p>
```

Das CSS-Stylesheet wird als Attribut in den betreffenden Tag eingetragen. Es beginnt mit dem Schlüsselwort `style`. In den Anführungszeichen für den Wert des Attributs werden die Deklarationen (ohne geschweifte Klammern) eingetragen.

In dem Beispiel wird der Absatz kursiv (`font-style:italic`) und fett (`fontweight:bold`) formatiert.

Einbinden von Styles in den Header einer HTML-Datei

```
<!DOCTYPE ...>
<html>
  <head>
    <title>Das HTML-&lt;stlye&gt;-Element</title>
    <meta http-equiv="Content-Type" content="text/html;
charset=iso8859-1">
    <style type="text/css">      p      {font-size: 14px; color:
#000000; background-color:
#FFFFFF; }      h1      {font-size: 20px; color: #FF0000;
background-color: #FFFFFF; }
    </style>
  </head>
  <body>
    <h1>Überschrift</h1>
    <p>Absatz</p>
  </body>
</html>
```

Die Stylesheets für die HTML-Datei werden im Header in `<style>`-Tags eingetragen. Sie gelten nur für diese Datei. Falls dieselben Formatierungen auch in anderen HTMLDateien verwendet werden sollen, müssen sie dort ebenfalls eingetragen werden.

Auslagern der Stylsheets in eine separate CSS-Datei

CSS-Datei "standard.css"

In dieser Datei werden die Styles definiert.

```
p      {font-size: 14px; color: #000000; background-color: #FFFFFF; }
h1     {font-size: 20px; color: #FF0000; background-color: #FFFFFF; }
```

Inkludieren der CSS-Datei in die HTML-Datei

1. Mit dem einzelnen <link>-Tag

```
<!DOCTYPE ...>
<html>
  <head>
    <title>CSS-Datei mit link-Tag einbinden</title>
    <meta http-equiv="Content-Type" content="text/html;
charset=iso8859-1">
    <link rel="stylesheet" type="text/css" href="02_standard.css">
  </head>
  <body>
    <h1>Überschrift</h1>
    <p>Absatz</p>
  </body>
</html>
```

Zur Referenzierung einer CSS-Datei wird der `<link>`-Tag verwendet. Das Attribut `rel="stylesheet"` definiert den Bezug, dass eine Stylesheet-Datei geladen wird. Das Attribut `type="text/css"` gibt den MIME-Type 'CSS-Datei' an. Das href-Attribut kennen Sie schon, es wurde für Hyperlinks verwendet. Hier wird es zur Referenzierung der zu inkludierenden CSS-Datei verwendet.

2. Mit dem <style>-Tag-Paar

Die Referenzierung kann auch mit dem `<style>`-Tag-Paar erfolgen.

```
<!DOCTYPE ...>
<html>
  <head>
    <title>CSS-Datei mit style-Tag einbinden</title>
    <meta http-equiv="Content-Type" content="text/html;
charset=iso8859-1">
    <style type="text/css">
      @import url(02_standard.css)
    </style>
  </head>
  <body>
    <h1>Überschrift</h1>
    <p>Absatz</p>
  </body>
</html>
```

Der Unterschied zwischen beiden Inkludierungsvarianten besteht darin, dass in der ersten Variante ein HTML-Tag verwendet wird in der zweiten Variante das CSSSprachelement `url()`.

(1.2) Kombination der Möglichkeiten zum Einbinden der CSSStyles

Die verschiedenen Möglichkeiten der Einbindung von CSS-Styles können kombiniert werden. So können in einer zentralen CSS-Datei Styles definiert werden. Diese können im Style- Bereich des Headers überschrieben und dann noch einmal direkt in dem HTML-Element überschrieben werden. Der letzte Wert, der definiert wurde, wird verwendet.

Kaskadieren

Diese Möglichkeit wird als Kaskadieren bezeichnet und daher wird auch der Name "Cascading Stylesheets" verwendet. Der folgende Quellcode zeigt ein Beispiel.

```
<!DOCTYPE ...>
<html>
  <head>
    <title>CSS-Styles kaskadieren</title>
    <meta http-equiv="Content-Type" content="text/html;
charset=iso8859-1">
    <link rel="stylesheet" type="text/css" href="02_standard.css">
    <style type="text/css">      p      {color: #894F4F;
background-color: #FAF5A9; }    </style>

  </head>
  <body>
    <h1>Überschrift</h1>
    <p>Absatz</p>

    <p style="font-size: 10px;font-style:italic; font-weight:bold">
Formatierter Absatz</p>

  </body>
</html>
```


In diesem Beispiel wird zuerst die oben definierte CSS-Datei "standard.css" inkludiert. Diese enthält Styles für `<h1>`- und `<p>`-Tags.

Danach werden in dem `<style>`-Tag im Header der HTML-Datei für den `<p>`-Tag die Zeichenfarbe (`color`) und die Hintergrundfarbe (`background-color`) definiert. Da diese Definition hinter der Inkludierung der CSS-Datei steht, wird sie verwendet.

Im zweiten `<p>`-Tag der HTML-Datei befindet sich ein `style`-Attribut, dass für diesen Tag die Schriftgröße neu festlegt und weitere Eigenschaften hinzufügt.

Bearbeitungshinweis

Testen Sie die fünf Beispiele in separaten HTML-Dateien. Ändern Sie dabei auch die Werte in den Deklarationen.

2.1.5 Maßeinheiten und Farbangaben

Die CSS-Deklarationen bestehen aus **Eigenschaften: Wert**-Paaren. Für viele der Eigenschaften müssen Werte angegeben werden. Die Werte können in unterschiedlichen Maßeinheiten angegeben werden.

Weiterführende Informationen: [SelfHTML - > Maßeinheiten und Farbangaben](#)

Die folgende Tabelle gibt einen Überblick über die Maßeinheiten:

Maß- einheit	Erläuterung
Absolute Längenangaben	
<code>pt</code>	Punkt, 1 pt = 1/72 inches
<code>pc</code>	Pica, 1 pc = 12 pt
<code>in</code>	Inch (Zoll), 1 inch = 2,54 cm
<code>mm</code>	Milimeter
<code>cm</code>	Zentimeter
<code>px</code>	Pixel, bezogen auf die Ausflösung px/inch des Ausgabemediums

Relative Längenangaben	
em	<p>Bezieht sich auf die Schriftgröße des Elements. D.h. die Schriftgröße (<code>font-size</code>) selbst sollte nicht in <code>em</code> angegeben werden. <code>1em</code> entspricht der Schriftgröße.</p> <p>Wenn die Schriftgröße des Elements aber doch in <code>em</code> gesetzt wird, wird der Bezug auf die Schriftgröße des Elternelements geändert.</p>
ex	<p>Bezieht sich auf die Schriftgröße des Kleinbuchstaben <code>x</code>. Auch hier gilt wieder, dass die Schriftgröße des Elements nicht in <code>ex</code> angegeben werden sollte.</p> <p>Wenn die Schriftgröße des Elements aber doch in <code>ex</code> gesetzt wurde, wird der Bezug auf die Schriftgröße des Elternelements geändert.</p>
%	<p>Prozent, Problem ist hier, worauf sich die Prozentangabe bezieht.</p> <p>Die %-Angabe kann gut für Breitenangaben verwendet werden. Wenn z.B. die Breite (<code>width</code>) in Prozent angegeben wird, dann kann sich die Angabe auf das Elternelement (z.B. eines umschließenden Tabellenrahmens) oder wenn keins existiert, auf die Browserfensterbreite beziehen.</p>
Farbangaben	
#RRGGBB #RGB	<p>Hexadezimale Angabe: z.B. <code>#FF0000</code> oder <code>#F00</code> (verkürzt, entspricht auch <code>#FF0000</code>)</p>
rgb(R, G, B)	<p>Angabe mit Dezimalwerten von 0 bis 255, z.B. <code>rgb(255, 0, 0)</code></p>
rgb(%, %, %)	<p>Angabe in Prozent bezogen auf die Skala von 0 bis 255, z.B. <code>rgb(100%, 0%, 0%)</code></p>
red	<p>Farbname, die Namen sind vorgegeben, eine Tabelle finden Sie in -> SelfHTML</p>

2.1.6 Inline- und Block-Elemente, <div>- und -Tag

HTML unterscheidet Inline- und Block-Elemente.

Weiterführende Informationen: [CSS4You -> Workshop: Elemente](#)

Block-Elemente

Ein Block-Element erzeugt bezogen auf den Text einen eigenen Absatz. Es ist bei Texten mit den Absätzen in Word vergleichbar.

Zu den Block-Elementen gehören die folgenden Tags: <p>, <h1> .. <h6>, <table>, , , <form>, <div>

Inline-Elemente

Ein Inline-Element bleibt im Textfluss in der Zeile. Es ist mit der Zeichenformatierung in Word vergleichbar.

Zu den Inline-Elementen gehören die HTML-Tags zur Zeichenformatierung , <i> und <u>, der Zeilenumbruch
 sowie die Tags , <a>, <input> und .

Inline-Elemente sollen innerhalb von Block-Elementen stehen und dürfen i.allg. selbst keine Block-Elemente enthalten. Sie können Texte und teilweise andere Block-Elemente enthalten.

Z. B. kann ein Hyperlink-Tag .. neben Text auch Formatierungen wie .. und/oder ein Bild enthalten.

<div>- und -Tag

<div>- und -Tags sind spezielle HTML-Elemente für die CSS-Formatierung.

Der <div>-Tag ist ein Block-Element und der -Tag ist ein Inline-Element. Sie haben keine weiteren Eigenschaften und dienen dazu, mit Hilfe von CSS formatiert zu werden.

Die Anwendung dieser beiden Elemente wird im weiteren Verlauf dieses Moduls erläutert.

Weiterführende Informationen: [SelfHTML => Allgemeine Elemente für Textbereiche](#)

(1.4) Das Box-Modell

Weiterführende Informationen: [SelfHTML -> Box-Modell](#)

Box

Jedes HTML-Block-Element wird als ein rechteckiger Bereich angesehen und beansprucht damit einen bestimmten Platz. Dieser Bereich wird als Box bezeichnet.

Eine Box enthält die folgenden Bestandteile:

- Bereich für den Inhalt
- Innenabstand (padding) zwischen Inhaltsabstand und Rahmen
- Rahmen (border)
- Außenabstand (margin) zwischen äußerer Begrenzung der Box und dem Rahmen Das folgende Bild veranschaulicht den Aufbau einer Box:

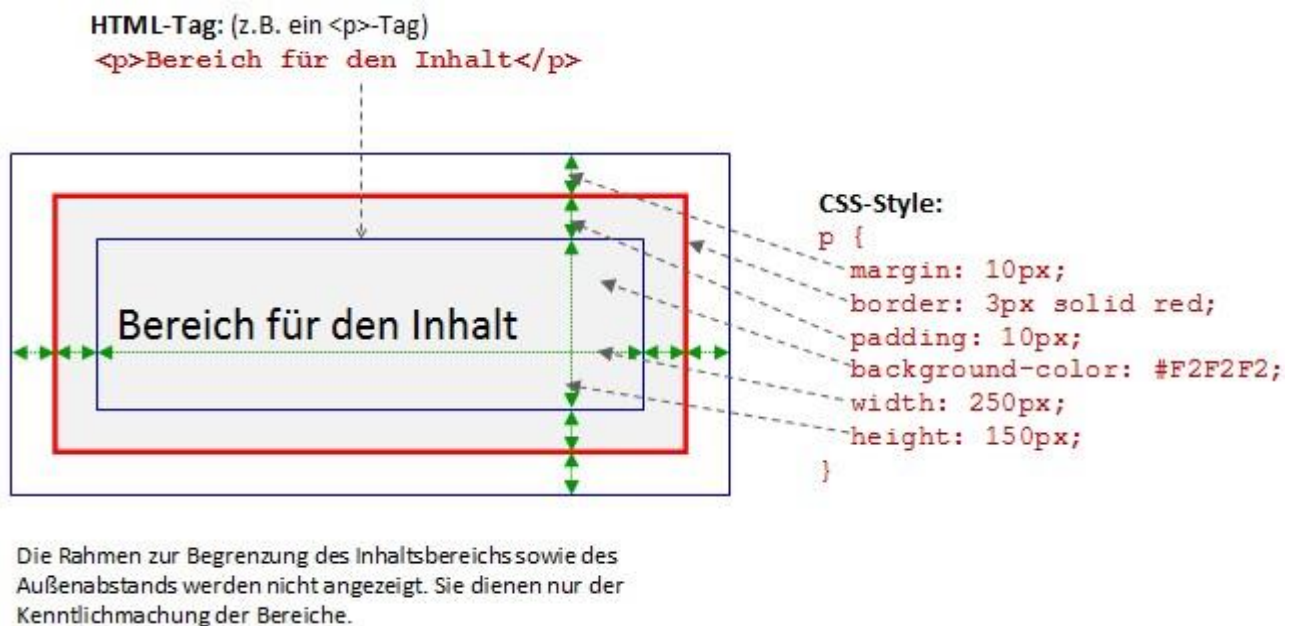


Bild 2.1: Aufbau einer Box

Wenn der Box eine Hintergrundfarbe zugewiesen wird, dann erstreckt sie sich im inneren Bereich bis zum Rahmen. Der Außenabstand ist immer transparent.

CSS-Deklarationen einer Box

Die Eigenschaften einer Box werden durch die folgenden CSS-Deklarationen definiert:

```
p { margin: 10px; border:
3px solid red; padding:
10px; background-color:
#F2F2F2; width: 250px;
height: 150px;
```

```
}
```

Die Breite des Außenrandes wird durch `margin` definiert. Der Rahmen (Breite, Linienart, Linienfarbe) wird durch `border` definiert. Die Breite des Innenabstandes wird durch `padding` definiert. Die Breite des Inhaltsbereichs wird durch `width` und die Höhe durch `height` definiert. Die Hintergrundfarbe wird durch `background-color` definiert.

Die Breite der Randbereiche können für jede Seite einzeln angegeben werden.

Gesamtgröße einer Box

Da durch `width` und `height` die Größe des Inhaltsbereichs definiert werden, müssen zur Berechnung der Gesamtgröße für die Breite und die Höhe die Stärken des Innenabstands, des Rahmens und des Außenabstands jeweils auf beiden Seiten hinzugerechnet werden.

```
Gesamtbreite = Margin-Links + Border-Links + Padding-Links
               + Width
               + Margin-Rechts + Border-Rechts + Padding-Rechts
```

```
Gesamthöhe   = Margin-Oben + Border-Oben + Padding-Oben
               + Height
               + Margin-Unten + Border-Unten + Padding-Unten
```

Beispiel

```
Gesamtbreite = 10px + 3px + 10px + 250px + 10px + 3px + 10px = 296px
Gesamthöhe   = 10px + 3px + 10px + 150px + 10px + 3px + 10px = 196px
```

Das folgende Bild zeigt die durch den angegebenen `<p>`-Tag in Verbindung mit dem CSS-Style erzeugte Browseransicht.

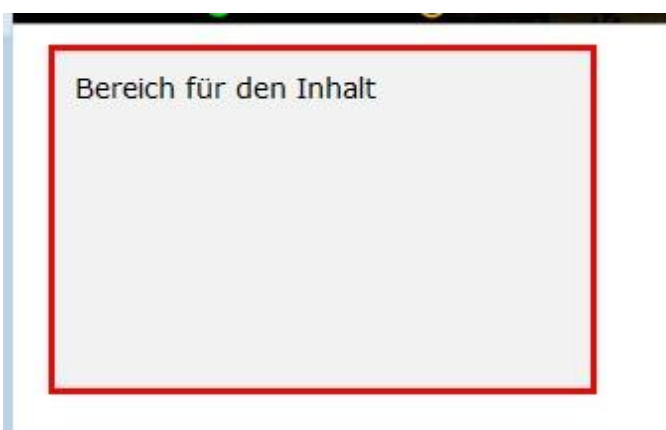


Bild 2.2: Browseransicht der Box mit dem `<p>`-Tag

Aufgabe

Erstellen Sie eine HTML-Datei mit dem in Bild 2.1 angegebenen `<p>`-Tag und dem CSSStyle. Experimentieren Sie mit dem Inhaltstext. Geben Sie so viel Text ein, damit der Text mehr Platz benötigt als der Inhaltsbereich groß ist.

Musterlösung:

- 02_CSS/01_css-grundlagen/07-box-style.html
- 02_CSS/01_css-grundlagen/07-box-style_2.html

2.1.8 CSS-Klassen und CSS-IDs

Styles, die in externen Dateien oder im `<style>`-Tag des Headers der HTML-Seite definiert werden, werden als zentrale Formate bezeichnet. Sie wurden bisher immer einem HTML-Tag zugewiesen. Diese Möglichkeit reicht aber nicht aus, da dann z. B. alle `<p>`-Tags dasselbe Format haben würden.

Aus diesem Grund gibt es noch zwei weitere Möglichkeiten, einem HTML-Tag einen Style zuzuordnen. Es besteht die Möglichkeit, CSS-Klassen und auch Individualformate (IDs) zu definieren.

Weiterführende Informationen: [SelfHTML -> Zentrale Formate definieren](#)

(1.5) Klassen für CSS-Styles

Eine CSS-Klasse bietet die Möglichkeit, einen Style zu definieren, der mit Hilfe des class-Attributs allen gewünschten Tags zugeordnet werden kann.

Syntax der Definition einer CSS-Klasse

```
.class1 {color: #F00; font-weight: bold; }
```

Der Name der Klasse ist ein beliebiger Name. Um zu kennzeichnen, dass es sich um einen Klassen-Namen handelt, wird davor ein Punkt geschrieben.

Die Definition der Eigenschaften:Wert-Pare (der Deklarationen) erfolgt wie bei den Styles für HTML-Tags.

Zuordnung einer Klasse zu einem HTML-Tag

Die CSS-Klassen können einzelnen HTML-Tags mit Hilfe des class-Attributes zugeordnet werden.

```
<p class="class1">Absatz mit class1 formatiert</p>
```

Beispiel

```
<!DOCTYPE ...>
<html>
  <head>
    <title>Verwendung von CSS-Klassen</title>
    <meta http-equiv="Content-Type" content="text/html;
charset=iso8859-1">
    <style type="text/css">      p    {font-size: 14px; color:
#000000; background-color: #FFFFFF; }
      .class1 {color: #F00; font-weight: bold; }
      .class2 {color: #0FF; font-style: italic; }
    </style>
  </head>
  <body>
    <h1>Überschrift</h1>
    <p>Absatz</p>
    <p class="class1">Absatz mit class1 formatiert</p>
    <p class="class2">Absatz mit class2 formatiert</p>
  </body>
</html>
```

Aufgabe

Testen Sie die Verwendung von CSS-Klassen zur Formatierung von HTML-Elementen.

Musterlösung: 02_CSS/01_css-grundlagen/05_css-klassen.html

(1.6) IDs zur Definition von Individualformaten für HTML-Tags

Eine CSS-Klasse kann beliebig vielen HTML-Tags zugeordnet werden. Die IDs sind dafür vorgesehen, genau einem HTML-Tag einen Style zuzuordnen.

Syntax der Definition einer ID

```
#id1 {color: #F00; font-weight: bold; }
```

Der Name der ID ist ein beliebiger Name. Um zu kennzeichnen, dass es sich um einen ID-Namen handelt, wird davor ein #-Zeichen geschrieben.

Die Definition der Eigenschaften: Wert-Pare (der Deklarationen) erfolgt wie bei den Styles für HTML-Tags.

Zuordnung einer ID zu einem HTML-Tag

Die ID kann einem einzelnen HTML-Tags mit Hilfe des id-Attributes zugeordnet werden.

```
<p id="id1">Absatz mit id1 formatiert</p>
```

Beispiel

```
<!DOCTYPE ...>
<html>
  <head>
    <title>Verwenden von Individualformaten</title>
    <meta http-equiv="Content-Type" content="text/html;
charset=iso8859-1">
    <style type="text/css">      p      {font-size: 14px; color:
#000000; background-color: #FFFFFF; }
      #id1 {color: #F00; font-weight: bold; }
      #id2 {color: #0FF; font-style: italic; }
    </style>
  </head>
  <body>
    <p>Absatz</p>
    <p id="id1">Absatz mit id1 formatiert</p>
    <p id="id2">Absatz mit id2 formatiert</p>
  </body>
</html>
```

Aufgabe

Testen Sie die Verwendung von IDs zur individuellen Formatierung einzelner HTMLElemente.

Musterlösung: 02_CSS/01_css-grundlagen/06_css-id.html

2.1.9 Mehrfach-Formatdefinitionen und Formatdefinitionen verschachtelter HTML-Elemente

Weiterführende Informationen: [SelfHTML -> Zentrale Formate definieren](#)

Formatdefinitionen für mehrere HTML-Elemente

Texte können beispielsweise in beispielsweise Absätzen (<p>-Tags), in Listenelementen (-Tags) und in Tabellenzellen (<td>-Tags) stehen. Wenn alle Texte dieselbe Formatierung erhalten sollen, dann ist es wünschenswert, dass die Formatierung nur einmal aufgeschrieben wird. Derartige Mehrfach-Formatdefinitionen sind in CSS möglich. Die entsprechenden HTML-Elemente, CSS-Klassen bzw. CSS-IDs werden durch Kommas getrennt zusammengefasst.

Mehrfach-Format-Definition

```
p, li, td {font-size: 14px; color: #000000;}
```

(1.7) Formatdefinitionen für verschachtelte HTML-Elemente

Die Verschachtelung von HTML-Elementen ist ein Grundprinzip von HTML-Dateien. Wenn wird das HTML-Grundgerüst betrachten, dann ist dort die hierarchische Struktur bereits zu erkennen. Sie beginnt bei dem <html>-Tag, dem Wurzel-Element einer HTML-Seite. Die erste Verschachtelungsebene sind der <head>- und der <body>-Tag.

Im <body>-Tag befinden sich die Tags, die den Inhalt der Seite aufnehmen.

Listen, Tabellen und Formulare werden auch durch hierarchische Tag-Strukturen definiert.

Beispiel

Es wird die folgende HTML-Struktur betrachtet:

```
<p>Text</p>

<div class="div1">
  <p>Text mit <span>Zeichenformatierung</span></p>
</div>

<div class="div2">
  <p>Text mit <span>Zeichenformatierung</span></p>
</div>
```

In diesem Beispiel soll nur die Textfarbe verändert werden.

Zentrale CSS-Formate

```
/* Nicht verschachtelter p-Tag */ p
{color: #000; }

/* p Tag verschachtelt im ersten div-Tag */


```

Text

Text mit Zeichenformatierung

Text mit Zeichenformatierung

Bild 1.1: Browseransicht der Musterlösung

Bild 1.1 zeigt, dass einem HTML-Tag durch verschachtelte Formatdefinitionen unterschiedliche Formatierungen zugewiesen werden können. Der zugehörige HTMLCode ist gleichzeitig ein Beispiel für die Verwendung von `<div>`- und ``-Tags.

Aufgabe

Erstellen Sie eine HTML-Datei und testen Sie mit deren Hilfe die in diesem Abschnitt beschriebenen Formatdefinitionen.

Musterlösung: 02_CSS/01_css-grundlagen/08_verschachtelt.html

2.1.10 Kapitelschwerpunkte

Syntax eines Styles

Selektor { Eigenschaft:Wert; ... }

Beispiele für vollständige Styles:

```
p {font-size: 14px; color: #000000; background-color: #FFFFFF; } h1
{font-size: 20px; color: #FF0000; background-color: #FFFFFF; }
```

Syntax einer CSS-Kommentars

```
/* CSS-Kommentar */
```

Möglichkeiten des Einbindes der Stylesheets

Style einem HTML-Tag als Attribut zuweisen

```
<p style="font-style:italic; font-weight:bold">Formatierter
Absatz</p>
```

style-Tag im Header einer HTML-Datei

```
<style type="text/css"> p {font-size: 14px; color:
#000000; background-color:
#FFFFFF; } h1 {font-size: 20px; color: #FF0000;
background-color: #FFFFFF; }
</style>
```

Inkludieren einer externen CSS-Datei

```
<link rel="stylesheet" type="text/css" href="02_standard.css">
```

oder

```
<style type="text/css"> @import url(02_standard.css) </style>
```

Block- und Inline-Elemente

Ein Block-Element erzeugt bezogen auf den Text einen eigenen Absatz, d.h. einen rechteckigen Bereich.

Ein Inline-Element bleibt im Textfluss in der Zeile.

Das Box-Modell

Bestandteile einer Box:

Eine Box enthält die folgenden Bestandteile:

- Bereich für den Inhalt
- Innenabstand (padding) zwischen Inhaltsabstand und Rahmen
- Rahmen (border)

- Außenabstand (margin) zwischen äußerer Begrenzung der Box und dem Rahmen

CSS-Style für eine Box

```
p { margin: 10px; border: 3px solid red;
padding: 10px; background-color: #F2F2F2; width:
250px; height: 150px;
}
```

Gesamtbreite und -höhe einer Box

```
Gesamtbreite = Margin-Links + Border-Links + Padding-Links
               + Width
               + Margin-Rechts + Border-Rechts + Padding-Rechts
Gesamthöhe   = Margin-Oben + Border-Oben + Padding-Oben
               + Height
               + Margin-Unten + Border-Unten + Padding-Unten
```

CSS-Klassen und IDs

Klasse

Syntax der Definition einer CSS-Klasse

```
.class1 {color: #F00; font-weight: bold; }
```

Zuordnung einer Klasse zu einem HTML-Tag

```
<p class="class1">Absatz mit class1 formatiert</p>
```

ID

Syntax der Definition einer ID

```
#idl {color: #F00; font-weight: bold; }
```

Zuordnung einer ID zu einem HTML-Tag

```
<p id="idl">Absatz mit idl formatiert</p>
```

Mehrfache und verschachtelte Formatdefinitionen

Mehrfach-Format-Definition

```
p, li, td {font-size: 14px; color: #000000;}
```

Formatdefinition für den verschachtelten -Tag

```
/* span-Tag verschachtelt im ersten div- und p-Tag */
.div1 p span {color: #808; }
```

2.2 Schrift- u. Hyperlinkformatierung

Stundenempfehlung

FIAE	FISI	ITSK	IFK	ITSE
4	4	4	4	4

[2.2.1 Übersicht über das Kapitel](#)

[2.2.2 CSS-Eigenschaften zur Schriftformatierung](#)

[2.2.3 CSS-Eigenschaften zur Hyperlink-Formatierung](#)

[2.2.4 Hinweise zur Schriftformatierung einer HTML-Seite](#)

[2.2.5 Kapitelschwerpunkte](#)

2.2.1 Übersicht über das Kapitel

In diesem Kapitel werden die CSS-Eigenschaften zur Text- und zur Hyperlinkformatierung behandelt. Sie werden die Namen der betreffenden Eigenschaften in Verbindung mit deren Werten kennenlernen und an Beispielen erste Erfahrungen mit den Formatierungen machen können.

2.2.2 CSS-Eigenschaften zur Schriftformatierung

Weiterführende Informationen: [SelfHTML -> Schriftformatierung](#)

(2.1) Übersicht über die CSS-Eigenschaften zur Schriftformatierung

Eigenschaft	Zugelassene Werte	Beispiel
Zeichensatz: <code>font-family</code>	Namen von Zeichensätzen	<pre>p { font-family: Verdana, Arial, Helvetica, sans-serif; }</pre>

Schriftstil: font-style	normal (Standard) italic (kursive) oblique (schräggestellt)	.schraeg { font-style: italic; }
Schriftvariante: fontvariant	normal (Standard) small-caps (Kapitälchen)	.kapitaelchen { font-variant: smallcaps; }
Schriftgröße: font-size	xx-small, x-small, small, medium (Standard), large, x- large, xx-large Zugelassene Längenangaben: z.B. px (Pixel), mm (Millimeter), % (in Prozent)	.schriftgroesse { font-size: 12px }
Schriftstärke: font-weight	lighter (dünner) normal (Standard) bold (fett) bolder (extrafett) 100 (extra dünn) 500 (normal) 900 (extrafett)	.fetter { font-weight: 600; }
Schrift gesamt font	Erlaubte Eigenschaften: font-family, -style, -variant, -weight size/line-height (siehe Absatzkontrolle), Pflichtangaben sind size und family, alle anderen Angaben sind optional. Die Reihenfolge ist siehe Beispiel.	.schrift1 { font: italic smallcaps bold 20px Verdana ; }
Wortabstand wordspacing	normal (Standard) Zugelassene Längenangaben: z.B. px (Pixel), mm (Millimeter), % (in Prozent)	.wortabstand { word-spacing: 16px; }
Zeichenabstand letterspacing	normal (Standard) Zugelassene Längenangaben: z.B. px (Pixel), mm (Millimeter), % (in Prozent)	.zeichenabstand { letter-spacing: 4px; }
Ausschmückung textdecoration	none (Standard) underline (unterstrichen) overline (überstrichen) line- through (durchgestrichen) blink (blinkend)	.durchgestrichen { text-decoration: line-through; }

Groß-/Kleinbuchstaben <code>texttransform</code>	<code>none</code> (Standard) <code>capitalize</code> (Wortanfang groß) <code>uppercase</code> (Großbuchstaben) <code>lowercase</code> (Kleinbuchstaben)	<pre>.grossbuchstaben { text-transform: uppercase; }</pre>
Textfarbe <code>color</code>	Zulässige Farbangabe, z. B. red, #F00, rgb(255,0,0)	<pre>.farbe { color: #FF0000; }</pre>
Text-Schatten <code>text-shadow</code>	Angeben werden: horizontaler, vertikaler Versatz, Unschärfe, Farbe des Schattens Versatz und Unschärfe in einer Längeneinheit, Farbe in einer Farbangabe.	<pre>.schatten { text-shadow: 4px 2px 3px blue }</pre>
Zeichenbreite <code>font-</code>	<code>normal</code> (Standard) <code>wider</code> (weiter)	<pre>.geweitet { font-stretch:</pre>
<code>stretch</code>	<code>narrower</code> (enger) <code>condensed</code> (gedrängt) <code>semi-condensed</code> (halb gedrängt) <code>extra-condensed</code> (stark gedrängt) <code>ultra-condensed</code> (extrastark gedrängt) <code>expanded</code> (geweitet) <code>semi-expanded</code> (halb geweitet) <code>extra-expanded</code> (stark geweitet) <code>ultra-expanded</code> (extrastark geweitet)	<pre>expanded; }</pre>

Generische Schriften

Die Eigenschaft `font-family` erfordert die Angabe von Namen von Schriften. Das setzt voraus, dass die angegebene Schrift auf dem Rechner auch unter dem Namen installiert sein muss. Sonst wird die Standardschriftart des Browsers verwendet.

Eine Möglichkeit, die Wahrscheinlichkeit zu erhöhen, dass die Schriftart verfügbar ist, besteht darin, mehrere durch Kommas getrennte Schriftarten anzugeben. Damit können Schriftnamen der verschiedenen Betriebssysteme angegeben werden.

Auch dann kann der Fall eintreten, dass keiner der angegebenen Schriftnamen auf dem Rechner gefunden wird. Für diesen Fall können generische Schriften angegeben werden. Diese haben Standardnamen, denen der Browser automatisch eine verfügbare Schrift zuordnet.

Folgende generische Schriften existieren:

serif	Schriften mit Abschlussstrichen (Serifen) oder andere Verzierungen Times New Roman
--------------	---

sans-serif	Schriften ohne Abschlusstrichen (Serifen) und Verzierungen Verdana, Arial, Helvetica
cursive	Kursive Schrift
fantasy	Dekorative Schrift
monospace	Schrift, bei der alle Zeichen die gleiche Breite haben Courier, Courier New

Aufgabe

Informieren Sie sich über die Schriftformatierungen in SelfHTML. Verwenden Sie dazu den oben angegebenen Hyperlink.

(2.2) Testen der Eigenschaften zur Schriftformatierung

Machen Sie sich jetzt selbst mit den Eigenschaften zur Schriftformatierung vertraut.

Aufgabe

Erstellen Sie dazu eine HTML-Seite und testen Sie alle Schriftformatierungen. Experimentieren Sie auch mit den angegebenen Werten.

Schräger Text TEXT MIT KAPITÄLCHEN

Fetter Text

Text mit vergrößertem Wortabstand.

Text mit vergrößertem Zeichenabstand

~~Durchgestrichener Text~~

TEXT NUR IN GROSSEN BUCHSTABEN

Text in roter Farbe mit Schatten

Text mit geweiteten Buchstaben

Bild 2.3: Ansicht der Musterlösung

Das Bild zeigt, dass der Mozilla Firefox die Eigenschaft `font-strech` (noch) nicht unterstützt.

Bearbeitungshinweise

Als Hilfestellung wird ein Absatz aus der Musterlösung dargestellt:

```
<p class="schriftgroesse"><span class="farbe">  
Text in roter Farbe  
<span class="schatten">mit Schatten</span></span></p>
```

Musterlösung: 02_CSS/02_schriftformatierung/01_schrift_test.html

2.2.3 CSS-Eigenschaften zur Hyperlink-Formatierung

Zustände von Hyperlinks

Hyperlinks können unterschiedliche Zustände annehmen. Diese Hängen davon ab, ob der Hyperlink angeklickt wurde bzw. ob er ausgewählt wurde.

Um die unterschiedlichen Hyperlink-Zustände kenntlich zu machen, stellt CSS sogenannte Pseudoklassen bereit.

(2.3) Übersicht über die Pseudoklassen zur HyperlinkFormatierung

Beispiel für einen Hyperlink

Der folgende Hyperlink soll formatiert werden:

```
<a href="seite1.html" target="_blank">Seite1</a>
```

Dieser Hyperlink öffnet die Seite in einem neuen Fenster.

Element: Pseudoklasse	Erläuterung	Beispiel
<code>a</code>	HTML-Tag zur Definition eines Hyperlinks	<pre>a { font-size: 12px; }</pre>
<code>a:link</code>	Hyperlink, der noch nicht besucht und auch nicht ausgewählt wurde	<pre>a:link { text-decoration: none; color: #0000A0; }</pre>
<code>a:hover</code>	Hyperlink, auf den mit der Maus gezeigt wird	<pre>a:hover { text-decoration: underline; color: #FF0000; }</pre>
<code>a:active</code>	Letzter angeklickter Hyperlink	<pre>a:active { text-decoration: blink; color: #404040; }</pre>
<code>a:visited</code>	Hyperlink, der besucht wurde (deren Seite sich im Browser-Cache befindet)	<pre>a:visited { text-decoration: linethrough; color: #000040; }</pre>
<code>a:focus</code>	Hyperlink, auf dem der Fokus liegt. Der Fokus kann z. B. mit der Tab-Taste oder durch anklicken festgelegt werden.	<pre>a:focus { text-decoration: blink; color: #000040; }</pre>

Aufgabe 1

Informieren Sie sich über die Hyperlink-Formatierungen in SelfHTML. Verwenden Sie dazu den oben angegebenen Hyperlink.

Aufgabe 2

Erstellen Sie sich eine HTML-Seite, in der Sie Hyperlinks auf Webseiten aus dem Internet definieren. Fügen Sie dieser Seite die CSS-Styles für Hyperlinks hinzu. Machen Sie sich dabei mit der Anzeige der formatierten Hyperlinks vertraut.

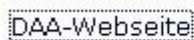
A screenshot of a web browser window. The address bar shows 'http://www.daa.de/'. Below the address bar, the text 'DAA-Webseite' is displayed. A dotted rectangular border is drawn around the text 'DAA-Webseite', indicating it is the focus of the image.A screenshot of a web browser window showing the Google search engine. The word 'Google' is visible in its characteristic font.

Bild 2.4: Ansicht der Musterlösung mit Fokus auf dem Hyperlink zur DAA-Webseite (gepunkteter Rahmen)

Musterlösung: 02_schriftformatierung/02_hyperlink-formatierung.html

2.2.4 Hinweise zur Schriftformatierung einer HTML-Seite

In diesem Kapitel möchte ich Ihnen einige allgemeine Aspekte vermitteln, wie die Formatierung der Schrift geplant und umgesetzt werden kann.

1. Die Formatierung beginnt mit den allgemeinen Eigenschaften, die für alle Texte und für die Hyperlinks gelten sollen.
Dazu gehören i. allg. die Schriftart, die Schriftgröße, die Schriftfarbe und natürlich die Farben der verschiedenen Hyperlink-Zustände.
Diese Eigenschaften werden dem <html>- und dem <body>-Tag zugeordnet.
2. Danach werden für die Tags, die Texte enthalten, die CSS-Styles definiert.
3. Danach folgen die benötigten Klassen und die IDs.

(2.4) Beispiel

Im Folgenden zeige ich Ihnen ein kleines Beispiel.

CSS-Style-Definitionen

```
<style type="text/css"> /*
Standardeinstellungen */
html, body {
    /* Textformatierung */    font-family: Verdana,
Arial, Helvetica, sans-serif;    font-size: 11pt;
color: #000033;
} h1, h2 {    color:
#00247D;    font-
weight: bold;
} h1 {    font-
size: 16pt;
    background-color: #FFDB82;
} h2 {    font-
size: 14pt;
color: #FFE784;
    background-color: #00187B;
} p span {    font-
style: italic;
}
.schatten {
    text-shadow: 6px 6px 0px #97925C;    font-
weight: bold;
}
</style>
```

Als allgemeine Schriftformatierungen wurden die Schriftart, die Schriftgröße und die Schriftfarbe definiert.

Danach wurde für <h1> und <h2> erst gemeinsame Formatierungen definiert und danach für jede Überschrift noch individuelle Formate.

Danach folgen weitere Format-Definitionen.

HTML-Quelltext, auf den diese Formate angewendet wurden

```
<h1>&Uuml;berschrift 1</h1>
<h2>&Uuml;berschrift 2</h2>
<p>Absatz mit <span>span-Tag</span></p>
<p class="schatten">Absatz mit CSS-Klasse formatiert</p>
```

Aufgabe

Erstellen Sie nach der Beispiel-Vorlage eine HTML-Seite und formatieren Sie die Schrift mit CSS-Styles.

Musterlösung: 02_CSS/02_schriftformatierung/03_seitenformatierung.html

2.2.5 Kapitelschwerpunkte

CSS-Eigenschaften zur Schriftformatierung

=> [Tabelle](#)

CSS-Eigenschaften zur Hyperlink-Formatierung

=> [Tabelle](#)

2.3 Abstände, Rahmen und Hintergrund

Stundenempfehlung

FIAE	FISI	ITSK	IFK	ITSE
4	4	4	4	4

[2.3.1 Überblick über dieses Kapitel](#)

[2.3.2 CSS-Eigenschaften für Innen- und Außenabstände](#)

[2.3.3 CSS-Eigenschaften für Rahmenlinien](#)

[2.3.4 CSS-Eigenschaften für Hintergrundfarben und -bilder](#)

[2.3.5 Hinweise zur Formatierung von Boxen](#)

[2.3.6 Kapitelschwerpunkte](#)

2.3.1 Überblick über dieses Kapitel

Die Texte gehören zum möglichen Inhalt der Boxen. In diesem Kapitel werden die CSSEigenschaften behandelt, mit denen die Innen- und die Außenabstände sowie die die Rahmen der Boxen formatiert werden können.

Damit werden die Möglichkeiten, HTML zu formatieren, deutlich erweitert.

2.3.2 CSS-Eigenschaften für Innen- und Außenabstände

Weiterführende Informationen:

- [SelfHTML -> Außenabstand \(Margin\)](#)
- [SelfHTML -> Innenabstand \(Padding\)](#)

(3.1) Übersicht über die CSS-Eigenschaften für Außen- und Innenabstand

Eigenschaft	Zugelassene Werte	Beispiel
Äußerer Abstand: <code>margin</code>	Zugelassene Längenangaben: z.B. <code>px</code> (Pixel), <code>mm</code> (Millimeter), <code>%</code> (in Prozent) Angegeben werden können ein, zwei, drei oder vier Werte ⁽¹⁾ .	<pre>.aussen1 { margin: 5px 10px; }</pre>
Äußerer Abstand einer Seite: <code>margin-top</code> oben <code>margin-right</code> rechts <code>margin-bottom</code> unten	Zugelassene Längenangaben: z.B. <code>px</code> (Pixel), <code>mm</code> (Millimeter), <code>%</code> (in	<pre>.aussen2 { margin-top: 10; margin-right: 5px; margin-bottom: 15px;</pre>
<code>margin-left</code> links	Prozent)	<pre>margin-left: 0px; }</pre>

Innerer Abstand: padding	Zugelassene Längenangaben: z.B. px (Pixel), mm (Millimeter), % (in Prozent). Angegeben werden können ein, zwei, drei oder vier Werte ⁽¹⁾ .	.innen1 { padding: 5px 10px 15px 20px; }
Innerer Abstand einer Seite: padding-top oben padding-right rechts padding-bottom unten padding-left links	Zugelassene Längenangaben: z.B. px (Pixel), mm (Millimeter), % (in Prozent) Angegeben werden können ein, zwei, drei oder vier Werte ⁽¹⁾ .	.innen2 { padding- top: 0; padding- right: 20px; padding- bottom: 0px; padding-left: 30px; }
(1) Zuordnung der angegebenen Werte in Abhängigkeit von der Anzahl: Ein Wert: gilt für alle vier Seiten Zwei Werte: erster Wert - oben und unten, zweiter Wert - rechts und links Drei Werte: erster Wert - oben, zweiter Wert - rechts und links, dritter Wert: unten Vier Werte: in der Reihenfolge oben, rechts, unten, links		

Aufgabe

Informieren Sie sich über die Formatierung der Außen- und Innenabstände von Blockelementen in SelfHTML. Verwenden Sie dazu die oben angegebenen Hyperlinks.

2.3.3 CSS-Eigenschaften für Rahmenlinien

Weiterführende Informationen: [SelfHTML -> Rahmen \(Border\)](#)

(3.2) Übersicht über die CSS-Eigenschaften für Rahmenlinien

Eigenschaft	Zugelassene Werte	Beispiel
Rahmenstärke: border-width border-top-width border-right-width border-bottom-width border-left-width	Zugelassene Längenangaben: z.B. px (Pixel), mm (Millimeter), % (in Prozent)	<pre>.rahmen-breite { border-width: 5px; }</pre>
Rahmenfarbe: border-color border-top-color border-right-color border-bottom-color border-left-color	Zugelassene Farbangabe	<pre>.rahmen-farbe { border-color: #F0F; }</pre>
Rahmentyp: border-style border-top-style border-right-style border-bottom-style border-left-style	none keiner (Standard) hidden unsichtbar dotted gepunktet dashed gestrichelt solid durchgezogen groove 3D-Effekt ridge 3D-Effekt inset 3D-Effekt outset 3D-Effekt	<pre>.rahmen-typ { border-style: groove; }</pre>
Rahmen - komplett: border border-top border-right border-bottom border-left	Anzugeben sind: Breite Farbe Typ	<pre>.rahmen { border: 2px #000 dashed; }</pre>

Aufgabe

Informieren Sie sich über die Formatierung der Rahmenlinien von Blockelementen in SelfHTML. Verwenden Sie dazu den oben angegebenen Hyperlink.

2.3.4 CSS-Eigenschaften für Hintergrundfarben und -bilder

Weiterführende Informationen: [SelfHTML -> Hintergrund](#)

(3.3) Übersicht über die CSS-Eigenschaften für den Hintergrund

Eigenschaft	Zugelassene Werte	Beispiel
Hintergrundfarbe: <code>background-color</code>	<code>transparent</code> ohne (Standard) oder zugelassene Farbangabe	<pre>.hintergrundfarbe { background-color: #FFF; }</pre>
Hintergrundbild: <code>background-image</code>	<code>none</code> ohne (Standard) <code>url (name)</code> Adresse Wenn Bild und Farbe angegeben werden, liegt das Bild über der Farbe.	<pre>.hintergrundbild { background-image: url (img/bild.jpg); }</pre>
Typ der Bildwiederholung: <code>background-repeat</code>	<code>repeat</code> Wiederholun g (Standard) <code>repeat-x</code> Wiederh. nur in x-Ri. <code>repeat-y</code> Wiederh. nur in y-Ri. <code>no-repeat</code> keine Wiederholung	<pre>.rahmen-typ { background-repeat: groove; }</pre>
Scroll-Eigenschaften des Hintergrundbildes: <code>backgroundattachment</code>	<code>scroll</code> Bild scrollt mit <code>fixed</code> Bild steht beim Scrollen	<pre>.bild-fest { background- attachment: fixed; }</pre>

Hintergrundposition: <code>backgroundposition</code>	Definiert die Position der linken und der oberen Kanten in der Reihenfolge links - oben: Horizontale Ausrichtung <code>left</code> , <code>center</code> , <code>right</code> Vertikale Ausrichtung <code>top</code> , <code>center</code> , <code>bottom</code> Die Position kann auch in den zugelassenen Längenangaben angegeben werden.	<pre>.hintergrundposition { background-position: 30px center; }</pre>
Hintergrund - komplett: <code>background</code>	Anzugeben sind: Hintergrundfarbe, Hintergrundbild, Hintergrundwiederholungstyp, Scroll-Eigenschaft, Hintergrundposition (x und y)	<pre>.hintergrund { background: black :url(img/bild.jpg) :repeat-y scroll :center top; }</pre>

Aufgabe

Informieren Sie sich über die Formatierung des Hintergrunds von Blockelementen in SelfHTML. Verwenden Sie dazu den oben angegebenen Hyperlink.

2.3.5 Hinweise zur Formatierung von Boxen

Die in diesem Abschnitt enthält Beispiele und Übungen, die Verständnis im Umgang mit verschachtelten Blockelementen aufbauen sollen.

Bei diesen Übungen wird bei den Blockelementen nur die Breite gesetzt. Die Höhe soll sich aus dem Platzbedarf des Inhalts der Box ergeben.

Wiederholung

Wiederholen Sie bitte noch einmal das Box-Model für Blockelemente. Sie sollten die einzelnen Bereiche sicher zuordnen können sowie die Gesamtbreite und die Gesamthöhe eines Blockelements berechnen können.

(3.4) Übung: Sichtbarmachen der inneren Abstände eines Blockelements

Um die inneren Abstände eines Blockelements sichtbar zu machen, wird in das Blockelement ein weiteres verschachtelt. Bei dem äußeren Element werden die Innenabstände gesetzt. Bei dem inneren Element wird der Außenabstand auf 0px gesetzt. Beide Elemente erhalten einen Rahmen, aus dem die Größen und damit auch die Innenabstände des äußeren Blockelements erkennbar sind.

Innenabstand der äußeren Box sichtbar machen

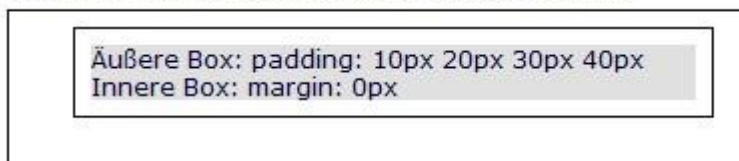


Bild 3.1: Anzeige des Innenabstandes des äußeren Blockelements

Der von den <p>-Tags belegte Platz wird durch die graue Hintergrundfarbe sichtbar gemacht.

HTML-Code

Das Bild 3.1 wurde durch den folgenden HTML-Code erzeugt:

```
<p>Innenabstand der äußeren Box sichtbar machen</p>
<div class="d1">
  <div class="d2">
    <p>Äußere Box: padding: 10px 20px 30px 40px<br />
    Innere Box: margin: 0px</p>
  </div>
</div>
```

CSS-Code

Die Formatierung wurde durch den folgenden CSS-Code erzeugt:

```
<style type="text/css"> html,
body {
  /* Textformatierung */
  font-family: Verdana, Arial, Helvetica, sans-serif;
  font-size: 11pt; color: #000033;
} p { margin:
10px;
  background-color: #E0E0E0;
}

.d1 { margin: 5px;
border: 1px solid #000;
padding-top: 10px;
padding-right: 20px;
padding-bottom: 30px;
padding-left: 40px;
width: 400px;
} .d2 { margin: 0px;
border: 1px solid #000;
}
</style>
```

Erläuterungen

Der Abstand zwischen dem Rahmen des äußeren und des inneren Blockelements wird durch den Innenabstand des äußeren Blockelements erzeugt. Das wird durch die folgenden Maßnahmen erreicht.

Das innere Element hat keinen Außenabstand (margin: 0px).

Dem inneren Blockelement wurde keine Breiten zugewiesen. In dem Fall füllt es den verfügbaren Platz in der Breite aus, der von dem äußeren Blockelement bereitgestellt wird. Der Innenabstand des äußeren Elements ist der freie Platz. Er beträgt links 40px und der rechts 20px.

Bei der Höhe verhält es sich anders. Die Höhe des inneren Elements wird durch deren Inhalt festgelegt. Da dem äußere Element wie auch dem inneren Element keine Höhe zugewiesen wurde, umschließt das äußere Element in der Vertikalen das innere Element. Der Abstand wird daher wieder durch die Innenabstände des äußeren Elements festgelegt. Er beträgt oben 10px und der unten 30px.

Aufgabe

Vollziehen Sie diese Übung nach. Experimentieren Sie mit den Abständen, der Breite und der Höhe des äußeren Blockelements.

Musterlösung: 02_CSS/03_abstand_rahmen/01_padding.html

(3.5) Übung: Sichtbarmachen der äußeren Abstände eines Blockelements

In dieser Übung sollen die äußeren Abstände des inneren Elements sichtbar gemacht werden. Dazu wird der gleiche HTML-Code verwendet. Er wird nur anders mit CSS formatiert. In diesem Fall wird der Innenabstand des äußeren Blockelements auf 0px gesetzt.

Dadurch zeigt der Abstand der Rahmen zwischen äußere und innerem Element die Breite des äußeren Abstands des inneren Blockelements an.

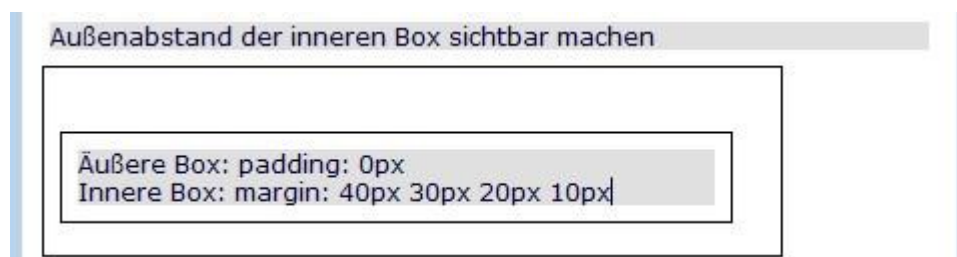


Bild 3.2: Anzeige des Außenabstands des inneren Blockelements

Die blauen Ränder Rechts und Links sind die Ränder des Browserfensters.

HTML-Code

Der HTML-Code ist mit dem aus Übung (3.4) identisch.

CSS-Code

```
<style type="text/css"> html,
body {
  /* Textformatierung */  font-family: Verdana,
  Arial, Helvetica, sans-serif;  font-size: 11pt;
  color: #000033;
} p {  margin: 10px;
background-color: #E0E0E0;
}
.d1 {  margin: 5px;
border: 1px solid #000;
padding: 0px;  width:
460px;
} .d2
{
  margin-top: 40px;  margin-
right: 30px;  margin-
bottom: 20px;  margin-left:
10px;  border: 1px solid
#000;
}
</style>
```

Um den Außenabstand des inneren Blockelements sichtbar zu machen, wurde der Innenabstand des äußeren Blockelements auf 0px gesetzt. Dadurch zeigen die Rahmenlinien der beiden Elemente den äußeren Abstand des inneren Elements an.

Aufgabe 1

Vollziehen Sie diese Übung nach. Experimentieren Sie auch hier wieder mit den Abständen, der Breite und der Höhe des äußeren Blockelements.

Musterlösung: 02_CSS/03_abstand_rahmen/02_margin.html

Aufgabe 2

Die Gesamtbreite der äußeren Blockelemente ist in beiden Beispiel gleich. Im ersten Beispiel (3.4) wurde die Breite des äußeren Elements auf 400px und im zweiten Beispiel (3.5) auf 460px gesetzt.

Erläutern Sie, warum die Breite in den beiden Beispielen unterschiedlich sein muss, um die gleiche Gesamtbreite zu erzeugen.

=> Lösung

Die Breite eines Blockelements ergibt sich aus der folgenden Formel:

```
Gesamtbreite = Margin-Links + Border-Links + Padding-Links
               + Width
               + Margin-Rechts + Border-Rechts + Padding-Rechts
```

Da das innere Blockelement sich in die Breits an das äußere anpasst, braucht für die Breitenberechnung nur das äußere Blockelement betrachtet zu werden.

Beispiel (3.4)

```
Gesamtbreite = 5 + 1 + 40 + 400 + 20 + 1 + 5
               = 472
```

Beispiel (3.4)

```
Gesamtbreite = 5 + 1 + 0 + 460 + 0 + 1 + 5
               = 472
```

Damit beide Boxen gleich breit sind, musste die Breite (width) im zweiten Beispiel um die Innenabstände links und rechts gegenüber dem 1. Beispiel vergrößert werden:

```
40 + 400 + 20 = 0 + 460 + 0
```

(3.6) Breite des inneren Block-Elements auf 100% setzen

Wenn die Breite des inneren Blockelements auf 100% gesetzt wird, dann ergibt sich die Breite des inneren Blocks aus der Breite des äußeren Elements.

Aufgabe

Wie breit ist der innere Block im Beispiel (3.5) bei width: 100%?

Testen Sie das Ergebnis und begründen Sie es danach.

=> Lösung

Außenabstand der inneren Box sichtbar machen

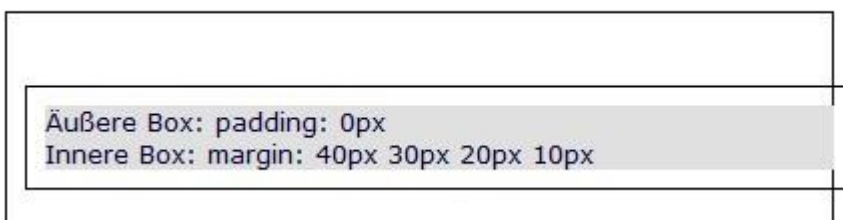


Bild: Breite des inneren Block: width: 100%

Da beim äußeren Block der Innenabstand links und rechts 0px beträgt, wurde die Breite äußeren Blocks auf 460px gesetzt. Diese Breite hat auch der innere Block durch die Eigenschaft "width: 100%". Die Breite des inneren Blocks hat also den gleichen wert wie die Breite des äußeren Blocks.

Hinweis

Beim inneren Block wurden keine Innenabstände gesetzt, sie beträgt daher 0px (Standardwert).

2.3.6 Kapitelschwerpunkte

CSS-Eigenschaften für Innen- und Außenabstand

=> [Tabelle](#)

CSS-Eigenschaften für Rahmenlinien

=> [Tabelle](#)

CSS-Eigenschaften für den Hintergrund

=> [Tabelle](#)

2.4 Ausrichtung und Absatzkontrolle

Stundenempfehlung

FIAE	FISI	ITSK	IFK	ITSE
3	3	3	3	3

[2.4.1 Übersicht über dieses Kapitel](#)

[2.4.2 CSS-Eigenschaften zur Ausrichtung und Absatzkontrolle](#)

[2.4.3 Kapitelschwerpunkte](#)

[2.4.4 Übung](#)

2.4.1 Übersicht über dieses Kapitel

Texte können links, zentriert oder rechts ausgerichtet sein. Dazu kommt noch die Möglichkeit, Texte in Blocksatz anzuzeigen. Neben diesen Ausrichtungen enthält CSS noch weitere Eigenschaften, um die Darstellung von Absätzen zu steuern.

2.4.2 CSS-Eigenschaften zur Ausrichtung und Absatzkontrolle

Weiterführende Informationen: [SelfHTML -> Ausrichtung und Absatzkontrolle](#)

(4.1) Übersicht über die Eigenschaften

Eigenschaft	Zugelassene Werte	Beispiel
Texteintrückung für die erste Zeile: <code>text-indent</code>	Zugelassene Längenangaben: z.B. <code>px</code> (Pixel), <code>mm</code> (Millimeter), <code>%</code> (in Prozent)	<code>.einrueck { text-indent: 20px }</code>
Zeilenabstand: <code>line-height</code>	<code>normal</code> kein zusätzlicher Abstand (Standard) Verwendet werden können alle zugelassenen Längenangaben Relative Längenangaben beziehen sich auf die Schriftgröße.	<code>.zeilenanstand { line-height: 1.2em }</code>
Vertikale Ausrichtung/ Versatz nach oben / unten: <code>vertical-align</code>	Vertikale Ausrichtung an: <code>baseline</code> Schriftlinie (Standard)	<code>v-ausrichtung { vertical-align: 14px; }</code>
	<code>text-top</code> Oberlänge <code>text-bottom</code> Unterlänge <code>top</code> oben bündig <code>middle</code> mittig <code>bottom</code> unten <code>sub</code> tiefer <code>super</code> höher <code>stellen</code> stellen	

	<p>Versatz bezogen auf vertikale Ausrichtung in:</p> <p>%, em, px und andere relative und absolute Längenangaben. Relative Längenangaben beziehen sich auf die Boxhöhe.</p>	<pre>v-versatz { vertical-align: top; }</pre>
<p>Horizontale Ausrichtung <code>text-align</code></p>	<p>left linksbündig right rechtsbündig center zentriert justify Blocksatz</p>	<pre>.zentriert { text-align: center; }</pre>
<p>Textumbruch <code>white-space</code></p>	<p>normal Automatisch (Standard) pre Wie im HTML-Code nowrap Kein Umbruch</p>	<pre>.kein_umbruch { white-space: nowrap }</pre>

Aufgabe

Informieren Sie sich über die CSS-Formatierungen zur Ausrichtung und Absatzkontrolle in SelfHTML. Verwenden Sie dazu den oben angegebenen Hyperlink.

(4.2) Übung: Testen der Eigenschaften zur Ausrichtung eines Absatzes

Um diese Eigenschaften testen zu können, werden Absätze (<p>-Tags) benötigt. Die drei Absätze sollen in einem <div>-Tag platziert werden, damit die Breite und Höhe des verfügbaren Platzes festlegbar sind.



Bild 4.1: Ansicht der Musterlösung mit drei <p>-Tags in einem <div>-Tag

Aufgabe 1

Erstellen Sie den HTML-Code ohne Formatierung.

Der obere Text "Text im div-Tag" soll direkt in den <div>-Tag eingetragen werden. Die Wörter "Links", "Zentriert" und "Rechts" sollen in <p>-Tags stehen.

Der <div>-Tag soll eine CSS-Klasse (class="...") erhalten.

=> Lösung

HTML-Code

```
<div class="d1">
  Text im div-Tag
  <p style="text-align: left; width: 200px">Links</p>
  <p style="text-align: center; width: 200px">Zentriert</p>
  <p style="text-align: right; width: 200px">Rechts</p>
</div>
```

Aufgabe 2

Erstellen Sie jetzt den CSS-Code.

Der <div>-Tag soll eine Breite von 300px haben und einen Rahmen. Texte sollen in diesem Block zentriert werden.

Die <p>-Tags sollen eine Breite von 200px haben. Die Texte sollen entsprechend dem Bild ausgerichtet werden. Sie sollen einen Rahmen und eine Hintergrundfarbe erhalten.

=> Lösung

CSS-Code

```
<style type="text/css">
body {
  /* Textformatierung */ font-family: Verdana,
  Arial, Helvetica, sans-serif; font-size: 14pt;
  color: #000033;
}
p
{
  background-color: #E0E0E0; margin: 5px; border:
  1px solid #000; padding: 0px;
} .d1 { width: 300px;
border: 1px solid #000;
text-align: center;
}
</style>
```

In Bild 4.1 ist zu erkennen, dass die Eigenschaft `text-align` nur auf Texte wirkt und nicht auf Block-Elemente. Das ist daran zu erkennen, dass die `<p>`-Tags im `<div>`-Tag nicht zentriert wurden.

Musterlösung: 02_CSS/04_ausrichtung/01_absatz_ausrichtung.html

(4.3) Übung: Testen der Zeilen- und Absatzabstände

Auf nicht durch CSS formatierten HTML-Seiten haben die Absätze große Abstände. Thema dieser Übung ist die Formatierung des Zeilenabstands innerhalb eines Absatzes und des Abstandes zwischen den Absätzen.

Das ist ein mehrzeiliger Text. Das ist ein mehrzeiliger Text. Das ist ein mehrzeiliger Text.

Das ist ein mehrzeiliger Text. Das ist ein mehrzeiliger Text. Das ist ein mehrzeiliger Text.

Das ist ein mehrzeiliger Text. Das ist ein mehrzeiliger Text. Das ist ein mehrzeiliger Text.

Bild 4.2: Ansicht der Musterlösung mit drei Absätzen (`<p>`-Tags) in einem `<div>`-Tag

Aufgabe

Erstellen Sie eine HTML-Seite, in der ein `<div>`-Tag mit mindestens drei `<p>`-Tags erstellt werden. Geben Sie in die `<p>`-Tags Texte ein, die über mehrere Zeilen gehen.

Weisen Sie dem `<div>`-Tag eine Breite zu, z.B. 300px. Weisen Sie dem `<p>`-Tag eine Zeilenhöhe für den Zeilenabstand innerhalb des Absatzes und einen Rand oben und unten für den Abstand der Absätze zu.

Experimentieren Sie mit den Werten für die Zeilenhöhe und dem Außenrand.

=> Lösung

HTML-Code

```
<div>
<p>Das ist ein mehrzeiliger Text.
Das ist ein mehrzeiliger Text.
Das ist ein mehrzeiliger Text.</p>
<p>Das ist ein mehrzeiliger Text.
Das ist ein mehrzeiliger Text.
Das ist ein mehrzeiliger Text.</p>
<p>Das ist ein mehrzeiliger Text.
Das ist ein mehrzeiliger Text.
Das ist ein mehrzeiliger Text.</p>
</div>
```

CSS-Code

```
<style type="text/css"> body
{
  /* Textformatierung */ font-family: Verdana,
  Arial, Helvetica, sans-serif; font-size: 10pt;
  color: #000033;
} p { margin: 0em
3px; line-height:
1.2em;
}
div {
  width: 300px;
}
</style>
```

Musterlösung: 02_CSS/04_ausrichtung/02_zeilenabstand.html

(4.4) Übung: Testen der Schriftposition in der Zeile

Für die vertikale Ausrichtung verfügt CSS über acht vordefinierte Ausrichtungswerte innerhalb der Zeile. Dazu kommt noch die Möglichkeit, den vertikalen Versatz manuell zu definieren.

In dieser Übung sollen die Darstellung des Textes unter Verwendung der vordefinierten Ausrichtungswerte untersucht werden.

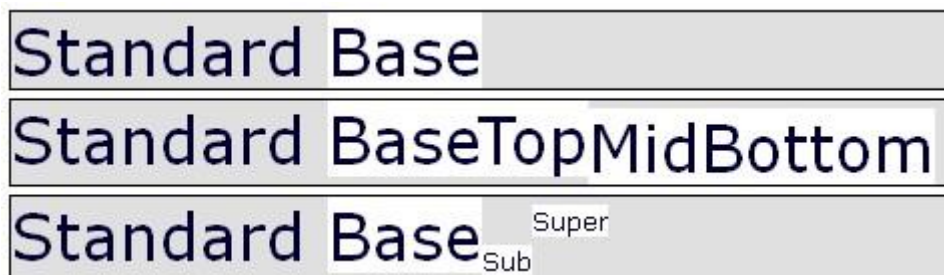


Bild 4.3: Ansicht der Musterlösung

Für die obere Zeile wird der HTML- und der CSS-Code angegeben. Der `<p>`-Tag und die ``-Tags sollen unterschiedliche Hintergrundfarben erhalten, damit die vertikale Ausdehnung erkennbar ist.

HTML-Code

```
<div class="d1">
  <p>Standard <span id="p1">Base</span></p>
</div>
```

CSS-Code

```
<style type="text/css"> body
{
  /* Textformatierung */  font-family: Verdana,
Arial, Helvetica, sans-serif;  font-size: 30pt;
color: #000033;
} p {  background-color: #E0E0E0;  margin: 5px;
border: 1px solid #000; padding: 0px;
} .dl
{
  width: 600px;
} #p1{  vertical-align:
baseline;  background-
color: #FFF;
}
```

Aufgabe

Erstellen Sie eine HTML-Datei und übernehmen Sie den Quellcode.

Ergänzen Sie danach die beiden fehlenden Absätze nach dem angegebenen Muster.
Jede vertikale Ausrichtung benötigt einen eigenen -Tag.

Musterlösung: 02_CSS/03_ausrichtung/03_schrift_position.html

2.4.3 Kapitelschwerpunkte

CSS-Eigenschaften zur Ausrichtung und Absatzkontrolle

=> [Tabelle](#)

2.5 Listenformatierung

Stundenempfehlung

FIAE	FISI	ITSK	IFK	ITSE
3	3	3	3	3

[2.5.1 Überblick über dieses Kapitel](#)

[2.5.2 CSS-Eigenschaften zur Listenformatierung](#)

[2.5.3 Hinweise zur Formatierung einer Liste](#)

[2.5.4 Kapitelschwerpunkte](#)

[2.5.5 Übung](#)

2.5.1 Überblick über dieses Kapitel

In diesem Kapitel werden die speziellen CSS-Eigenschaften behandelt, die zur Formatierung von Listen und Aufzählungen verfügbar sind.

Im Kapitel 2.7. wird aufbauend zu diesem Kapitel gezeigt, wie Listen zur Erstellung von Navigationsleisten genutzt werden können und wie sie dafür formatiert werden können.

2.5.2 CSS-Eigenschaften zur Listenformatierung

Weiterführende Informationen: [SelfHTML -> Listenformatierung](#)

(5.1) Übersicht über die CSS-Eigenschaften

Eigenschaft	Zugelassene Werte	Beispiel
Art des Auszeichnungspunktes: <code>list-style-type</code>	Auszeichnungspunkte für Listen: <code>disc</code> gefüllter Kreis (Standard) <code>circle</code> leerer Kreis <code>square</code> Rechteck	<pre>.liste { list-style-type: circle; }</pre>
	Auszeichnungspunkte für Nummerierungen: <code>decimal</code> 1. 2. 3 (Standard) <code>lower-roman</code> i. ii. iii. iv. <code>upper-roman</code> I. II. III. IV. <code>lower-latin</code> a. b. c. <code>upper-latin</code> A. B. C.	<pre>.nummerierung { list-style-type: upper-latin; }</pre>
	<code>lower-greek</code> α., β., γ., δ. u.a.	
	Kein Auszeichnungspunkt: <code>none</code>	
Listeneinrückung: <code>list-style-position</code>	<code>outside</code> ausgerückt (Standard) <code>inside</code> eingerückt	<pre>.position { list-style-position: inside; }</pre>
Eigene Grafik für den Auszeichnungspunkt: <code>list-style-image</code>	<code>none</code> keine (Standard) <code>url()</code> Eigenes Bild	<pre>.grafik-liste { list-style-image: url(img/bullet.gif); }</pre>

<p>Listendarstellung - komplett:</p> <p><code>list-style</code></p>	<p>Angegeben werden können:</p> <ul style="list-style-type: none"> - Art des Auszeichnungspunktes, - URL der eigenen Grafik, - Position des Auszeichnungspunkts 	<pre>ul { list-style: square inside; }</pre>
---	--	--

Aufgabe

Informieren Sie sich über die CSS-Formatierungen zur Formatierung von Listen in SelfHTML. Verwenden Sie dazu den oben angegebenen Hyperlink.

2.5.3 Hinweise zur Formatierung einer Liste

Die Formatierungsmöglichkeiten einer Liste sind:

- Auszeichnungspunkt festlegen oder Grafik verwenden
- Einrückungsart der Liste

Weitere Eigenschaften zur Formatierung der Listenelemente

Diese CSS-Formatierungen werden den ``- bzw. ``-Tags zugeordnet. Auf die ``-Tags für die Listenelemente können zur Formatierung sämtliche Eigenschaften angewendet werden, die auch bei den `<p>`-Tags für Absätze zur Anwendung kamen.

(5.2) Übung: Formatierung einer einfachen Liste

Eine einfache Liste hat das folgende Aussehen:

- Element 1
- Element 2
- Element 3

In dieser Übung sollen die Möglichkeiten zur Formatierung einfacher Listen untersucht werden.

HTML-Code: Verwendet werden die folgenden beiden Listen

```
<ul class="ul1">
  <li class="li1">Element 1<br />
mit 2. Zeile</li>
  <li class="li1">Element 2<br />
mit 2. Zeile</li>
  <li class="li1">Element 3<br />
mit 2. Zeile</li>
</ul>

<ul class="ul2">
  <li class="li2">Element 1<br />
mit 2. Zeile</li>
  <li class="li2">Element 2<br />
mit 2. Zeile</li>
  <li class="li2">Element 3<br />
mit 2. Zeile</li>
</ul>
```

Erzeugt werden sollen die folgenden Ansichten der Listen:

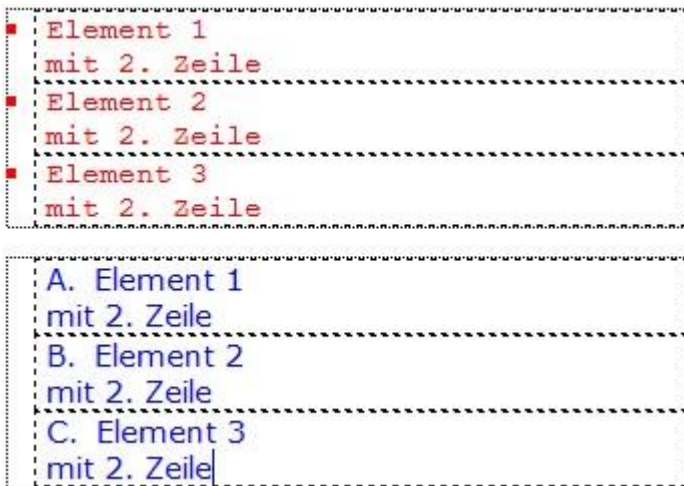


Bild 5.1: Ansichten der Musterlösung

CSS-Code der oberen Listen (rote Schrift)

```
body {
  /* Textformatierung */
  font-family: Verdana, Arial, Helvetica, sans-serif;
  font-size: 10pt; color: #000033;
}

.ul1 { width: 300px; list-
style-type: square; list-
style-position: outside;
border: 1px dotted #000;

margin-left: 0px; padding-
left: 12px; font-family:
monospace;
}

.li1 { list-style-type:
square; list-style-position:
outside; color: red; border:
1px dashed #000; padding-
left: 5px; margin-left: 0px;
}
```

Erläuterungen zur Formatierung

Rahmen für die Tags

Der ``- und der ``-Tag haben einen Rahmen erhalten. Damit soll die Ausdehnung der betreffenden Blöcke sichtbar gemacht werden.

Diese Rahmen haben mit der eigentlichen Formatierung der Listen nichts zu tun.

Schriftfarbe

Die Schriftfarbe des ``-Tags gilt für den Text des Listenelements und gleichzeitig für den Auszeichnungspunkt.

Verwendete Listen-Deklarationen (im ``-Tag)

```
list-style-type: square; list-style-position:
outside;
```

Mit diesen Listen-Deklarationen wurde die Liste formatiert.

Einstellen des Abstandes des Auszeichnungspunktes vom linken Rand und des Textabstandes zum Auszeichnungspunkt. Diese beiden Abstände können durch die Eigenschaft `padding-left` für den ``- und den ``-Tag eingestellt werden. In der Musterlösung beträgt `padding-left` für den ``-Tag `12px` und für den ``-Tag `5px`. Der äußere Rand (`margin-left`) wurde jeweils auf `0px` gesetzt.

Aufgabe 1

Erstellen Sie eine HTML-Datei und vollziehen Sie dieses Beispiel nach.

Experimentieren Sie dann mit den Werten, insbesondere mit den Rändern und den Auszeichnungspunkten.

Aufgabe 2

Erstellen Sie die CSS-Styles für die in Bild 5.1 zweite dargestellte Liste (blaue Schrift).

=> Lösung

CSS-Style

```
.ul2 { width: 300px; list-style-
type: upper-alpha; list-style-
position: inside; color: blue;
border: 1px dotted #000; padding-
left: 12px;
}
.li2 { margin-
left: 0px; border: 1px
dashed #000; padding-
left: 5px; margin-left:
0px;
}
```

Musterlösung: 02_CSS/05_listen/01_listen.html

(5.3) Formatierung einer verschachtelten Liste

Eine verschachtelte Liste ist eine Liste mit mehreren Ebenen mit Listenelementen.

Aufgabe 1

Wiederholen Sie den HTML-Code zur Erzeugung einer verschachtelten Liste.

=> [HTML-Modul -> 1.4 Listen und Aufzählungen](#)

Vorlage für die Liste

Erstellen Sie eine HTML-Seite und definieren Sie die folgende Liste.

Element 1 ○

Element 1.1

1. Element

1.1.1 2. Element

1.1.2 ○ Element

1.2 Element 2

Aufgabe 2

Diese Liste soll mit CSS Formatiert werden. Die Formatierung soll so erfolgen, dass die Liste wie im folgenden Bild dargestellt aussieht.

- Element 1
 - Element 1.1
 - Element 1.1.1
 - Element 1.1.2
 - Element 1.2
- Element 2

Bild 5.2: Verschachtelte Liste, mit CSS formatiert

Als Auszeichnungspunkte sollen alle Listenelemente eine Punkt erhalten. Die Abstände der Einrückungen können mit `padding-left` festgelegt werden.

=> Lösung

CSS-Styles

```
.div1 {
width: 300px;
}

.div1 ol, .div1 ul { /* Textformatierung */ font-
family: Verdana, Arial, Helvetica, sans-serif;
font-size: 10pt; color: #000033; list-style-type:
square; list-style-position: outside; /* border:
1px dotted #000; */ margin-left: 0px; padding-
left: 14px; font-family: monospace;
}

.div1 li { list-style-type:
disc; list-style-position:
outside; /* border: 1px
dashed #000; */ padding: 1px
0px 1px 5px; margin-left:
0px;
}
```

Musterlösung: 05_listen/02_listen_verschachtelt.html

Aufgabe 3

Erweitern Sie die Formatierung, so dass die erste und die dritte Ebene wieder Dezimalzahlen als Auszeichnungspunkte ausgeben.

1. Element 1
 - Element 1.1
 1. Element 1.1.1
 2. Element 1.1.2
 - Element 1.2
2. Element 2

Bild 5.3: Vorlage für die Formatierung der Liste

=> Lösung

Zusätzlich erforderliche CSS-Styles

```
.div1 li { list-style-
type: decimal;
}

.div1 li li{
list-style-type: disc;
}

.div1 li li li {
list-style-type: decimal;
}
```

Musterlösung: 02_CSS/05_listen/02a_listen_verschachtelt.html

2.5.4 Kapitelschwerpunkte

Spezielle CSS-Eigenschaften zur Listenformatierung

=> [Tabelle](#)

Weitere CSS-Eigenschaften, die zur Formatierung von Listen genutzt werden können

- Eigenschaften zur Textformatierung
- Eigenschaften für Ränder, Außen- und Innenabständen

2.5.5 Übung

(5.4) Erstellen und Formatieren einer verschachtelten Liste

Aufgabe

Erstellen Sie eine mindestens dreifach verschachtelte Liste und formatieren Sie sie mit CSS. Orientieren Sie sich an dem folgenden Bild.

- Element 1
 - Element 1.1
 - Element 1.1.1
 - Element 1.1.2
 - Element 1.2
- Element 2

Bild 5.4: Ansicht der Musterlösung

Musterlösung: [02_CSS/05_listen/03_listen_uebung.html](#)

2.6 Tabellenformatierung

Stundenempfehlung

FIAE	FISI	ITSK	IFK	ITSE
3	3	3	3	3

[2.6.1 Überblick über dieses Kapitel](#)

[2.6.2 CSS-Eigenschaften zur Tabellenformatierung](#)

[2.6.3 Hinweise zur Formatierung einer Tabelle](#)

[2.6.5 Kapitelschwerpunkte](#)

2.6.1 Überblick über dieses Kapitel

Tabellen sind und bleiben ein bedeutendes Mittel zu Darstellung von Daten. Dementsprechend verfügt auch CSS über Eigenschaften, um Tabellen formatieren zu können.

Diese Eigenschaften werden in diesem Kapitel behandelt.

2.6.2 CSS-Eigenschaften zur Tabellenformatierung

Weiterführende Informationen: [SelfHTML -> Tabellen](#)

(6.1) Übersicht über die CSS-Eigenschaften

Hinweis: Mit Ausnahme des `caption-side`-Styles gelten alle anderen Tabellen-Styles für den `<table>`-Tag.

Eigenschaft	Zugelassene Werte	Beispiel
Ausrichtung der Tabellenüberschrift: <code>caption-side</code>	Anzeige der Überschrift (<code><caption></code> -Tag) <code>top</code> oberhalb der Tabelle (Standard) <code>bottom</code> unterhalb <code>left</code> links <code>right</code> rechts	<pre>.table_header { caption-side: left; }</pre>

Breite-Inhalt-	Vorrang für die Zellbreite hat	<code>.zellbreite {</code>
Vorrang: <code>table-layout</code>	<code>auto</code> der Inhalt (Standard) <code>fixed</code> die Breitenabgabe	<code>table-layout: fixed; }</code>
Zusammenfallen der Rahmen: <code>border-collapse</code>	Die Rahmen fallen <code>separate</code> nicht zusammen (Standard) <code>collapse</code> zusammen	<code>.eine-rahmenlinie { border-collapse: collapse; }</code>
Rahmenabstand: <code>border-spacing</code>	Festgelegt wird der Abstand zwischen Zellen/Außenrahmen Zugelassene Längenangaben: z.B. <code>px</code> (Pixel), <code>mm</code> (Millimeter)	<code>.rahmen-abstand { border-spacing: 5px; }</code>
Anzeige leerer Zellen: <code>empty-cells</code>	Leerer Zellen werden <code>hide</code> nicht angezeigt (Standard) <code>show</code> angezeigt	<code>zeige-leere-zelle { empty-cells: show; }</code>
Sprachausgabe: <code>speak-header</code>	Der Kopfzelleninhalt wird <code>once</code> nur einmal vorgelesen (Standard) <code>always</code> bei jeder Zelle wiederholt	<code>speak-kopfzellinhalt { speak-header: always; }</code>

Aufgabe

Informieren Sie sich über die CSS-Formatierungen zur Formatierung von Tabellen in SelfHTML. Verwenden Sie dazu den oben angegebenen Hyperlink.

2.6.3 Hinweise zur Formatierung einer Tabelle

Boxen-Modell einer Tabelle

Eine Tabelle ist hierarchisch aufgebaut. Begrenzt wird sie durch das `<table>`-Tag-Paar. Innerhalb des `<table>`-Tag-Paars werden die Zeilen (row) durch `<tr>`-Tag-Paare definiert. Die Tabellenzellen werden in den `<tr>`-Tags durch `<td>`-Tags (data) definiert. Jede Zeile soll die gleiche Anzahl Zellen erhalten. Der folgende HTML-Code zeigt ein Beispiel.

HTML-Code einer Tabelle

```
<div>
<table>
  <tr>
    <td><p>abcdefg</p></td>
    <td><p>abcdefg</p></td>
    <td><p>abcdefg</p></td>
  </tr>

  <tr>
    <td><p>abcdefg</p></td>
    <td><p>abcdefg</p></td>
    <td><p>abcdefg</p></td>
  </tr>
</table>
</div>
```

In diesem Beispiel wurden die Daten innerhalb der Zellen noch in `<p>`-Tags gesetzt. Damit besteht die Möglichkeit, den Innenrand der Zellen sichtbar zu machen.

Der die Tabelle umschließende `<div>`-Tag soll den äußeren Rand des `<table>`-Tags sichtbar machen.

(6.2) Übung: Ergründen des Verhalten der Ränder-Einstellungen

Die `<table>`-, `<tr>`- und `<td>`-Tags sind Block-Elemente. Sie verfügen aber über spezielle Eigenschaften, die für Tabellen benötigt werden.

Diese speziellen Eigenschaften beziehen sich auf die Außen- und Innenabstände.

Ziel der Übung

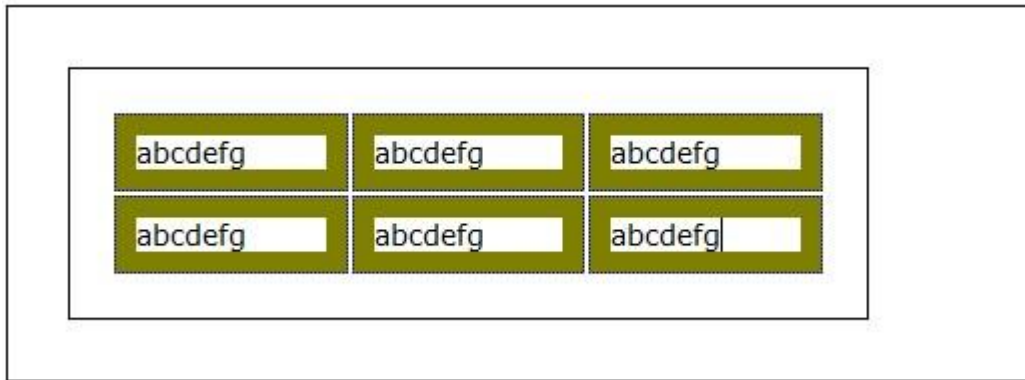
Das Ziel dieser Übung besteht darin, die Möglichkeiten und Grenzen für die Festlegung der Außen- und Innenabstände über die allgemeinen CSS-Eigenschaften `padding` und `margin` kennenzulernen.

CSS-Styles der Musterlösung

```
<style type="text/css">
div { margin: 0px;
padding: 0px;
border: 1px solid #000;
} table { margin:
30px; padding: 20px;
border: 1px solid #000;
width: 400px;
/* border-collapse: collapse; */
} tr {
margin: 0px;
padding: 0px;
} td { border: 1px
dotted #00f;

background-color: #808000;
padding: 10px; margin:
20px;
} p { margin: 0px;
padding: 0px; /* width:
100%; height: 100%; */
border: 0px solid #0f0;
background-color: #fff;
}
</style>
```

Dieser CSS-Code erzeugt die folgende Formatierung der Tabelle:



abcdefg	abcdefg	abcdefg
abcdefg	abcdefg	abcdefg

Bild 6.1: Ansicht der Tabelle mit dem angegebenen CSS-Code

Erläuterungen zur CSS-Formatierung

Der äußere Rahmen wird durch den `<div>`-Tag erzeugt.

Der darin befindliche Rahmen wird durch die `margin`-Eigenschaft des `<table>`-Tags erzeugt. Der Abstand zu den Zellen wird durch die `padding`-Eigenschaft des `<table>`-Tags erzeugt.

Der Abstand der Schrift in den Zellen zum Rahmen wird durch die `padding`-Eigenschaft der `<td>`-Tags erzeugt.

Aufgabe

Testen Sie unterschiedliche Werte für `margin` und `padding` und auch für `border` für die drei Tags-Arten der Tabelle und ermitteln Sie die Wirkung auf die Formatierung der Tabelle.

Fassen Sie Ihre Beobachtungen schriftlich zusammen.

=> Lösung

`<tr>`-Tag

Die Formatierung der Tabelle wird durch die `margin`- und `padding`-Eigenschaften des `<tr>`-Tag nicht beeinflusst.

`<td>`-Tag

Beim `<td>`-Tag hat nur die `padding`-Eigenschaft Einfluss auf die Formatierung. Die `margin`-Eigenschaft beeinflusst die Formatierung nicht.

Hinweis

Die Abstände zwischen den Zellen können durch die `border-spacing`- und die `border-collapse`-Eigenschaft beeinflusst werden.

<table>-Tag

Beim `<table>`-Tag beeinflussen die `margin`- und die `padding`-Eigenschaft die Formatierung der Tabelle.

Aufgabe

Erstellen Sie eine Tabelle mit mindestens drei Zeilen und 3 Spalten. Formatieren Sie die Tabelle mit CSS wie im folgenden Bild dargestellt.

1	2	3
4	5	6
7	8	9

Bild 6.2: Gewünschte Ansicht der Tabelle

=> Lösung

CSS-Code der Musterlösung

```
<style type="text/css">
table {
  border-collapse:
collapse; empty-cells:
show; width: 300px;
width: 300px;
}
tr
{
  border: 1px solid #000;
} td { border-
width: 0px;;
```

```
}
</style>
```

(6.3) Weitere HTML-Tags für Tabellen

Die Verfügbarkeit verschiedener HTML-Tags erleichtert die Formatierung einer HTMLSeite mit Hilfe von CSS, da dadurch bei der Formatierung die Anzahl u. U. der CSSKlassen reduziert werden kann.

Zellen für Überschriften der Spalten

Neben den bisher verwendeten Tabellen-Tags `<table>`, `<tr>` und `<td>` gibt es für die Zellen der Spaltenüberschriften noch den `<th>`-Tag.

Spaltenbreiten zentral vordefinieren

Die Spaltenbreite kann für eine Tabelle mit Hilfe der Tags `<colgroup>` und `<col>` zentral für die Tabelle festgelegt werden.

Tabelle in Kopf, Körper und Fuß unterteilen

Weiterhin kann eine Tabelle in Kopf, Körper und Fuß unterteilt werden. Dazu stehen die Tags `<thead>`, `<tbody>` und `<tfoot>` zur Verfügung.

Aufgabe 1

Weiterführende Informationen: [SelfHTML -> Tabellen definieren](#)

Informieren Sie sich mit Hilfe des angegebenen Links über die Verwendung der zusätzlichen Tags für Tabellen.

Aufgabe 2

Erstellen Sie eine Tabelle nach dem folgenden Muster. Die erste Zeile soll der Tabellenkopf, die letzte der Tabellenfuß und die mittleren Zeilen der Tabellenkörper zugeordnet werden.

Ü1	Ü2	Ü3
1	2	3
1	2	3
F1	F2	F3

Bild 6.3: Ansicht der Musterlösung

=> Lösung

HTML-Code

```
<table border="1">
  <thead>
    <tr>
      <th>&Uuml;1</th>
      <th>&Uuml;2</th>
      <th>&Uuml;3</th>
    </tr>
  </thead>
  <tfoot>
    <tr>
      <td>F1</td>
      <td>F2</td>
      <td>F3</td>
    </tr>
  </tfoot>
  <tbody>
    <tr>
      <td>1</td>
      <td>2</td>
      <td>3</td>
    </tr>
    <tr>
      <td>1</td>
      <td>2</td>
      <td>3</td>
    </tr>
  </tbody>
</table>
```

Aufgabe 3

Formatieren Sie die Tabelle mit CSS ohne Verwendung von CSS-Klassen oder CSS-IDs.
Formatieren Sie die Zellen der drei Bereiche unterschiedlich.

2.6.5 Kapitelschwerpunkte

Spezielle CSS-Eigenschaften für die Formatierung von Tabellen

=> [Tabelle](#)

Weitere HTML-Tags für Tabellen

[SelfHTML -> Tabellen definieren](#)

2.7 Positionierung, Layouts und Navigationsleisten

Stundenempfehlung

FIAE	FISI	ITSK	IFK	ITSE
7	7	7	7	7

[2.7.1 Überblick über dieses Kapitel](#)

[2.7.2 CSS-Styles für die Positionierung und Anzeige](#)

[2.7.3 Einführung in die Positionierung von HTML-Elementen](#)

[2.7.4 Navigationsleisten aus Listen erzeugen](#)

[2.7.5 Vorlagen für mehrspaltige CSS-Layouts](#)

[2.7.6 Kapitelschwerpunkte](#)

[2.7.7 Übungen](#)

2.7.1 Überblick über dieses Kapitel

Hinter der Positionierung verbirgt sich die Erstellung von Layouts. In diesem Kapitel werden die CSS-Eigenschaften zur Positionierung behandelt. Danach folgen erste kleine Beispiele, wie Block-Elemente absolut und relativ positioniert werden können.

Danach wird ein Einfaches, aber funktionstüchtiges zweispaltige Layout erstellt und gezeigt, wie Listen zur Erstellung von Navigationsbereichen genutzt werden können.

An zwei Beispielen können Sie dann selbst ausprobieren, wie das Layout nur durch Änderungen in den CSS-Formatierungen geändert werden kann.

2.7.2 CSS-Styles für die Positionierung und Anzeige

Weiterführende Informationen: [SelfHTML -> Positionierung und Anzeige von Elementen](#)

(7.1) Übersicht über die CSS-Styles

Eigenschaft	Zugelassene Werte	Beispiel
Art der Positionierung einer Box: <code>position</code>	<code>static</code> normaler Elementfluss, wie er ohne Positionierung ist (Standard) <code>relative</code> Positionierung bezieht sich auf die Normalposition der positionierten Box <code>absolute</code> Positionierung bezieht sich auf die linke obere Ecke des Fensters/der übergeordneten Box <code>fixed</code> wie absolut, jedoch bleibt die Box beim Scrollen stehen	<pre>.positionsart { position: absolute; }</pre>
Startposition einer	<code>auto</code> kein Versatz (Standard)	<pre>.startposition {</pre>
Box-Kante: <code>top</code> obere Kante <code>bottom</code> untere Kante <code>left</code> rechte Kante <code>right</code> linke Kante	Zugelassene Längenangaben: z.B. <code>px</code> (Pixel), <code>mm</code> (Millimeter), <code>%</code> (in Prozent)	<pre>top: 100px; left: 0px }</pre>
Breite/Höhe des Inhaltsbereichs: <code>width</code> <code>height</code>	<code>auto</code> Richtet sich nach dem Inhalt (Standard) Zugelassene Längenangaben: z.B. <code>px</code> (Pixel), <code>mm</code> (Millimeter), <code>%</code> (in Prozent) Relative Angaben beziehen sich auf die übergeordnete Box.	<pre>.groesse { width: 200px; height: 200px }</pre>
Minimale/maximale Breite/Höhe einer Box: <code>min-width</code> <code>min-height</code> <code>max-width</code> <code>max-height</code>	Zugelassene Längenangaben: z.B. <code>px</code> (Pixel), <code>mm</code> (Millimeter), <code>%</code> (in Prozent)	<pre>.groessen-bereich { min-width: 200px; maxwidth: 300px; min-height: 300px; max-hight: 300px; }</pre>

<p>Art der Darstellung bei zu geringer Breite/Höhe der Box:</p> <p><code>overflow</code></p>	<p><code>visible</code> Inhalt wird vollständig dargestellt (Standard)</p> <p><code>auto</code> Anwendungsprogramm entscheidet</p> <p><code>hidden</code> Alles, was nicht einpasst, wird abgeschnitten</p> <p><code>scroll</code> Die Box erhält Scrollleisten</p>	<pre><code>.overflow-art { overflow: scroll; }</code></pre>
<p>Richtung des Schriftflusses:</p> <p><code>direction</code></p>	<p><code>ltr</code> von links nach rechts (Standard)</p> <p><code>rtl</code> von rechts nach links</p>	<pre><code>.direction-art { direction: ltr; }</code></pre>
<p>Textumfluss um eine Box:</p> <p><code>float</code></p>	<p><code>none</code> kein Umfluss</p> <p><code>left</code> Element links, Umfluss rechts</p> <p><code>right</code> Element rechts, Umfluss links</p>	<pre><code>.float-art { float: left; }</code></pre>
<p>Aufheben des durch float gesetzten Textumflusses:</p> <p><code>clear</code></p>	<p><code>none</code> keine Aufhebung</p> <p><code>left</code> Aufhebung von left</p> <p><code>right</code> Aufhebung von right</p> <p><code>both</code> Aufhebung von left u. right</p>	<pre><code>.clear-art { clear: both; }</code></pre>
<p>Definieren von Ebenen-Hirarchien für Boxen (oben und unten):</p> <p><code>z-index</code></p>	<p><code>auto</code> Gemäß der Reihenfolge Definition der Boxen im HTML-Code (Standard)</p> <p><code>zahl</code> Ganze Zahl für die Ebene, höherer Wert ist oben</p>	<pre><code>.z-index-nr { z-index: 20; }</code></pre>
<p>Art der Anzeige einer Box ohne Platzhalter:</p> <p><code>display</code></p>	<p><code>inline</code> Element erzeugt keinen Umbruch (Standard)</p> <p><code>block</code> Element erzeugt Umbruch</p> <p><code>list-item</code> Erzwingt Listendarstellung</p> <p><code>none</code> Keine Anzeige ohne Platz zu beanspruchen (Platzhalter) u. a.</p>	<pre><code>.display-art { display: none; }</code></pre>

Art der Anzeige einer Box mit Platzhalter: <code>visibility</code>	<code>visible</code> Element erzeugt keinen Umbruch (Standard) <code>block</code> Element erzeugt Umbruch	<code>visibility-art { visibility: visible; }</code>
Beschneidungsbereich für Grafik: <code>clip</code>	<code>auto</code> Anzeige ohne Beschnitt (Standard) <code>rect(o,r,u,l)</code> Beschneidungsbereich	<code>clip-art { clip: auto; }</code>

Aufgabe

Informieren Sie sich über die CSS-Formatierungen zur Positionierung und Anzeige von HTML-Elementen in SelfHTML. Verwenden Sie dazu den oben angegebenen Hyperlink.

2.7.3 Einführung in die Positionierung von HTML-Elementen

Die Positionierung kann absolut und relativ erfolgen.

Absolute Positionierung

Bei der absoluten Positionierung werden die Koordinaten der linken oberen Ecke einer Box festgelegt. Auf diese Art erhält jede Box seine Position.

Relative Positionierung

Bei der relativen Positionierung ergibt sich die Position einer Box aus den Positionen der Boxen, die vor ihr definiert wurden in Verbindung mit den Werten der CSSEigenschaften `float` und `clear`.

(7.2) Vorbereiten der HTML-Seite

Benötigt werden mehrere Boxen, die dann positioniert werden sollen. Das folgende Bild zeigt die Musterlösung.

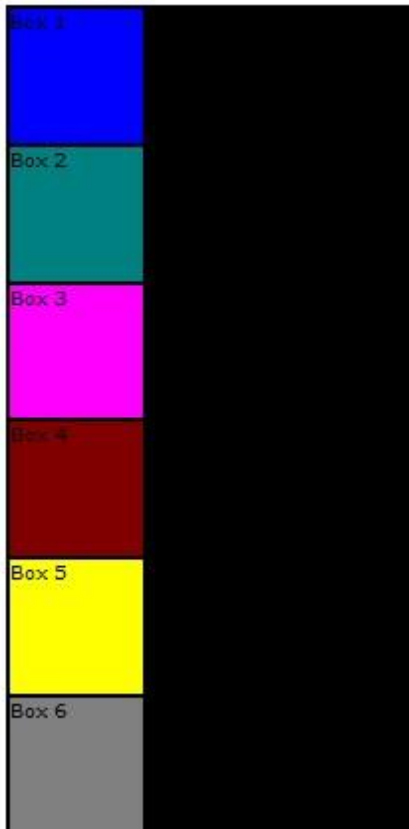


Bild 7.1: Ansicht der Musterlösung mit sechs Boxen in einer Box

HTML-Code der Boxen

```
<div id="d0">
  <div id="d1">Box 1</div>
  <div id="d2">Box 2</div>
  <div id="d3">Box 3</div>
  <div id="d4">Box 4</div>
  <div id="d5">Box 5</div>
  <div id="d6">Box 6</div>
</div>
```

Aufgabe

Formatieren Sie die Boxen. Sie sollen alle einen Rahmen erhalten und eine Hintergrundfarbe. Die inneren Boxen sollen eine Breite und eine Höhe erhalten.

Die äußere Box soll nur eine Breite erhalten, so dass drei Boxen in der Breite rein passen. Ich habe für die äußere Box die Größe auf 350 x 400 Pixel festgelegt und für die inneren Boxen auf 100 x 100 Pixel.

Experimentieren Sie auch mit negativen Werten.

Speichern Sie diese Datei als separate Datei ab, so dass Sie sie später wieder verwenden können.

Musterlösung: 02_CSS/07_layouts/01_positionierung.html

(7.3) Übung: Erste Schritte mit der absoluten Positionierung

Positionierung mit absoluten Koordinaten

Zur Positionierung mit absoluten Koordinaten gibt es zwei Möglichkeiten.

Positionierung mit Eckpunkt und Breite/Höhe

Eine Möglichkeit besteht darin, die Position eines Eckpunktes festzulegen und dazu die Breite. In diesen Übungen soll auch die Höhe festgelegt werden.

Die Positionsart muss auf `position: absolute;` festgelegt werden.

Mit `top`, `right`, `bottom` und `left` kann eine der vier Ecken der betreffenden Box festgelegt werden. Die Abstände beziehen sich bei der äußeren Box auf die betreffenden Seiten des Browserfensters und bei Boxen, die in einer Box definiert wurden, auf die betreffenden Seiten der übergeordneten Box. Die Größe der Box kann mit `width` und `height` festgelegt werden.

Beispiel: CSS-Code für die äußere Box

```
#d0 { border: 1px solid #000; background-color:
#000000; width: 350px; height: 400px; position:
absolute; left: 100px; top: 100px;
}
```

In diesem Beispiel wird die Position der linken oberen Ecke festgelegt. Mit `left` wird der Abstand von der linken Seite und mit `top` der Abstand zur oberen Seite festgelegt. Die Größe der Box wurde durch `width: 350px; height: 400px;` festgelegt.

Positionierung durch festlegen der Abstände zu allen vier Seiten der äußeren Box

Eine andere Möglichkeit besteht darin die Position der inneren Box durch Festlegen der Abstände zu allen vier Seiten mit den Eigenschaften `top`, `right`, `bottom` und `left` festzulegen. In diesem Fall dürfen Breite und Höhe nicht festgelegt werden.

Beispiel: CSS-Code für die Box 6

```
#d6 {  
  border: 1px solid #000; background-color: #808080;  
  /*width: 100px; height: 100px;*/  
  position: absolute;  
  
  top: 150px;  
  left: 120px;  
  right: 50px;  
  bottom: 50px;  
}
```

Durch die Festlegung der Abstände zu den vier Außenseiten der äußeren Box wird die Größe zwar absolut, aber relativ zur Größe der äußeren Box festgelegt.

Hinweis

Wenn Sie für die äußere Box keine Breite und Höhe angeben, sondern dafür die Abstände zu den vier Seiten des Browserfensters, dann ändert sich die Größe der Box in Abhängigkeit von der Größe des Browserfensters.

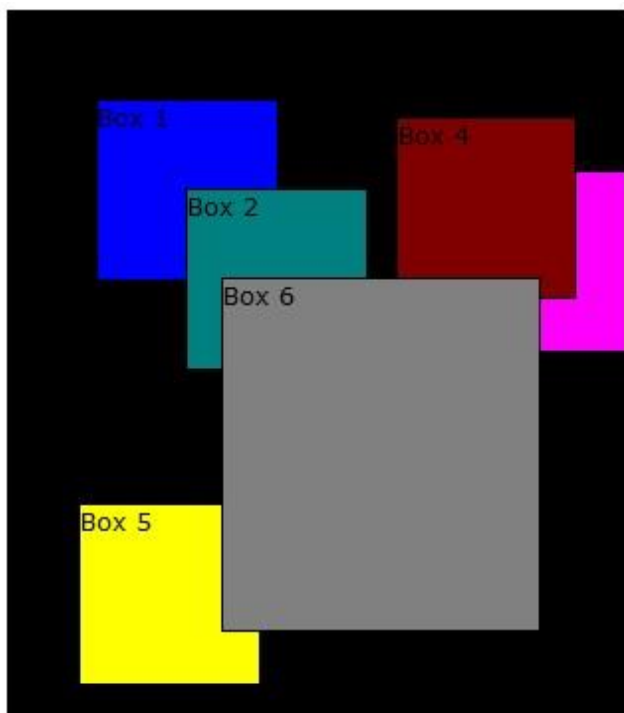


Bild 7.2: Ansicht der Musterlösung mit den positionierten Boxen

Aufgabe

Experimentieren Sie mit den angegebenen Möglichkeiten der Positionierung. Testen sie auch Kombinationen wie linker und rechter Abstand, oberer Abstand und Höhe.

Musterlösung: 02_CSS/07_layouts/02_positionierung_abs.html

Reihenfolge überlappender Boxen festlegen

Wenn sich Boxen überlappen, dann ist die erste definierte Box unten und die letzte definierte Box oben.

Die Reihenfolge der Boxen lässt sich mit der CSS-Eigenschaft `z-index` ändern. Als Werte werden ganzen Zahlen angegeben. Jeder Box kann ein `z-index`-Wert zugewiesen werden. Je größer der `z-index`-Wert ist, um so weiter oben wird sie angezeigt.

Beispiel `z-index`:

4;

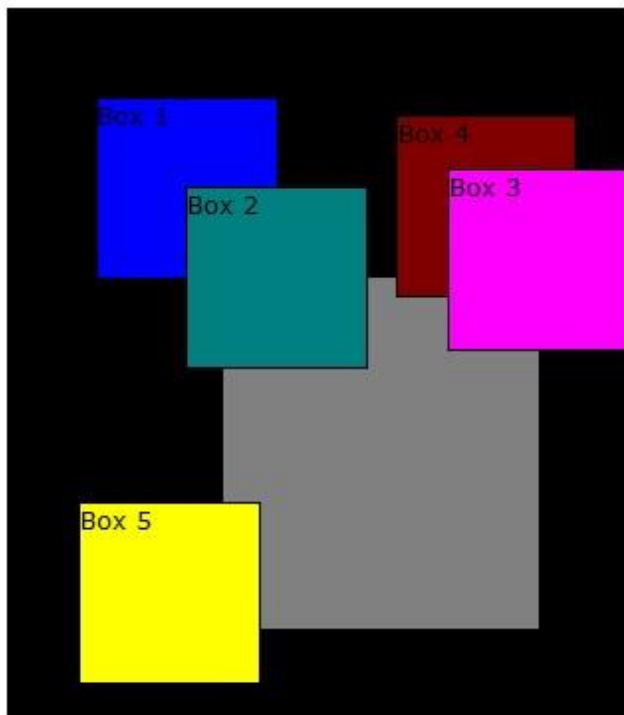


Bild 7.3: Änderung der Reihenfolge der Boxen mit der `z-index`-Eigenschaft

Musterlösung: 02_CSS/07_layouts/03_positionierung_z-index.html

(7.4) Übung: Erste Schritte zur relativen Positionierung

Die absolute Positionierung ist sinnvoll, wenn die Größen der Boxen fest sein können. Dieser Fall tritt aber nur sehr selten auf. Im Normalfall sollen sich die Höhen der Boxen dem Inhalt anpassen.

Für diesen Zweck stellt CSS auch Eigenschaften bereit, die eine relative Positionierung der Boxen ermöglichen.

Voraussetzung ist, dass die Boxen die Positionierungsart `position: relative;` für die betreffende Box, die relativ positioniert werden soll.

Mit `float` und `clear` kann gesteuert werden, ob die Boxen nebeneinander oder untereinander platziert werden sollen. Mit `float: left` wird einer Box zugewiesen, dass die ihr folgende Box neben ihr platziert werden soll (falls der Platz ausreicht). Mit `clear: left` wird einer Box zugewiesen, dass sie nicht neben der vorherigen Box angeordnet werden soll.

Die folgenden Aufgaben sollen Ihnen helfen, ein Gefühl für die Wirkung der beiden CSSEigenschaften `float` und `clear` zu entwickeln.

Anordnung der Boxen durch 'float: left'

Aufgabe

Verwenden Sie die in Übung (7.2) vorbereitete HTML-Datei als Vorlage für diese Aufgabe.

Versehen Sie sämtliche Boxen einheitlich mit den folgenden Positionierungsanweisungen:

```
position: relative; float:
left;
```

CSS-Code-Beispiel für die erste (obere) Box

```
#d1 { border: 1px solid #000; background-color:
#0000FF; width: 100px; height: 100px; position:
relative; float: left;
}
```

Testen Sie die Ausgabe. Verbreitern Sie auch die umschließende Box, so dass alle inneren Boxen in eine Reihe passen..

Musterlösung: 02_CSS/07_layouts/04_a_positionierung-relativ.html

Eine Box mit 'clear: left' nach unten links holen

Wenn die dritte Box eine neue Reihe beginnen soll, dann muss sie die CSS-Eigenschaft `clear: left` erhalten. Diese kann sie zusätzlich zu `float: left` erhalten.

Wählen Sie zwei Boxen aus und weisen Sie ihnen diese Eigenschaft zu und betrachten Sie dann die Browserausgabe. Die beiden ausgewählten Boxen sollten jetzt jeweils eine neue Reihe beginnen.

Musterlösung: 02_CSS/07_layouts/04_b_positionierung-relativ.html

Anordnung der Boxen mit 'float: right'

Die CSS-Eigenschaft `float: left` bewirkt, dass sie selbst links bleibt und die nachfolgende Box rechts neben ihr angeordnet wird.

Testen Sie jetzt auch noch, wie die Boxen angeordnet werden, wenn ihnen `float: right` zugewiesen wird. Das folgende Bild zeigt eine Anordnung, bei der den Boxen 2 und 6 `float: right` zugewiesen wurde.

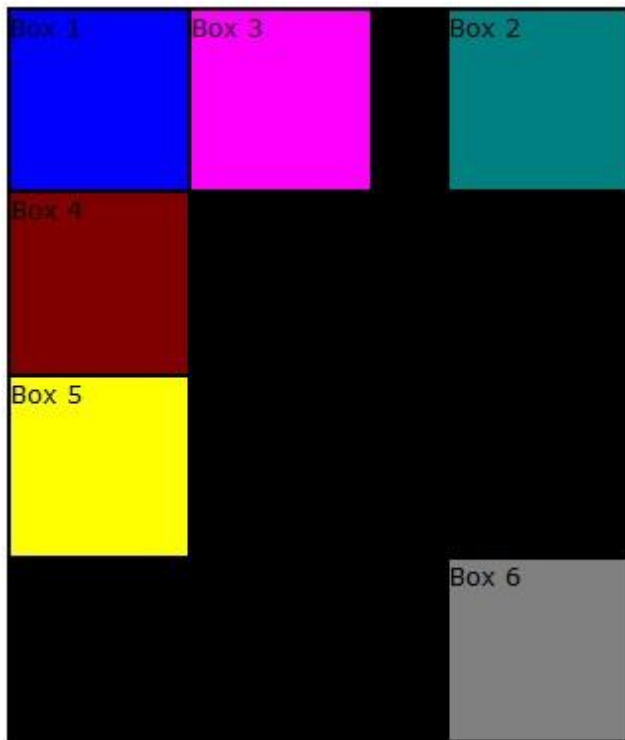


Bild 7.4: Ansicht der Musterlösung

Musterlösung: 02_CSS/07_layouts/04_c_positionierung-relativ.html

(7.5) Workshop: Ein einfaches Layout erstellen

Die bisherigen einführenden Übungen zur Positionierung werden höchstwahrscheinlich noch keine realen Vorstellungen aufgebaut haben, wie mit diesen CSS-Eigenschaften Layouts für Webseiten erstellt werden können.

Eine erste Vorstellung soll dieser Abschnitt ihnen entwickeln. Erzeugt werden soll ein zweispaltiges Layout mit Kopf- und Fußbereich. Das folgende Bild zeigt das angestrebte Ergebnis.

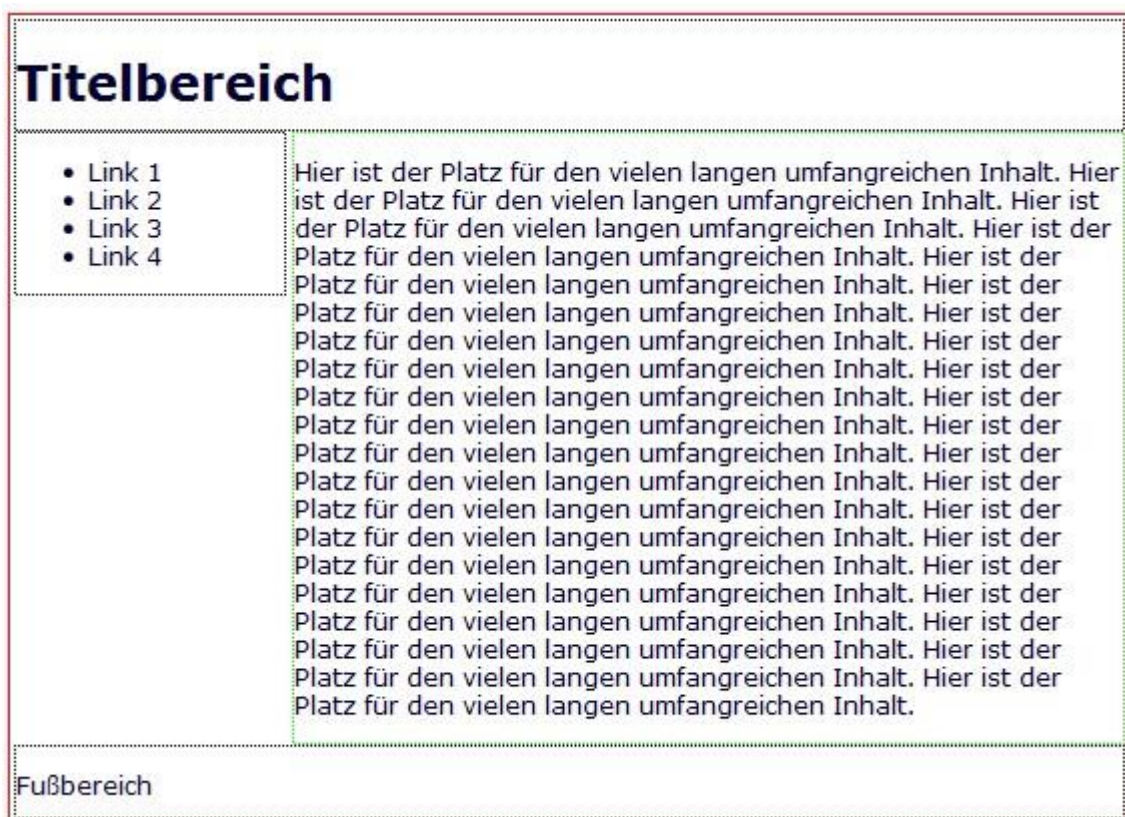


Bild 7.5: Ansicht des gewünschten Layouts

Das Layout soll feste Breiten haben, aber beliebige Höhen im Inhaltsbereich ermöglichen.

Die Rahmen wurden in dem Layout dargestellt, um die Positionen der einzelnen Boxen zu erkennen.

Das Layout wird im folgenden schrittweise erstellt.

1. Boxen erzeugen

Für das Layout werden die folgenden div-Bereiche (Boxen) benötigt:

- `container`: umschließende Box (im Bild mit Roter Rand)
- `header`: Box für den Kopfbereich
- `dir`: Box für den Navigationsbereich □ `content`: Box für den eigentlichen Inhalt □ `footer`: Box für den Fußbereich.

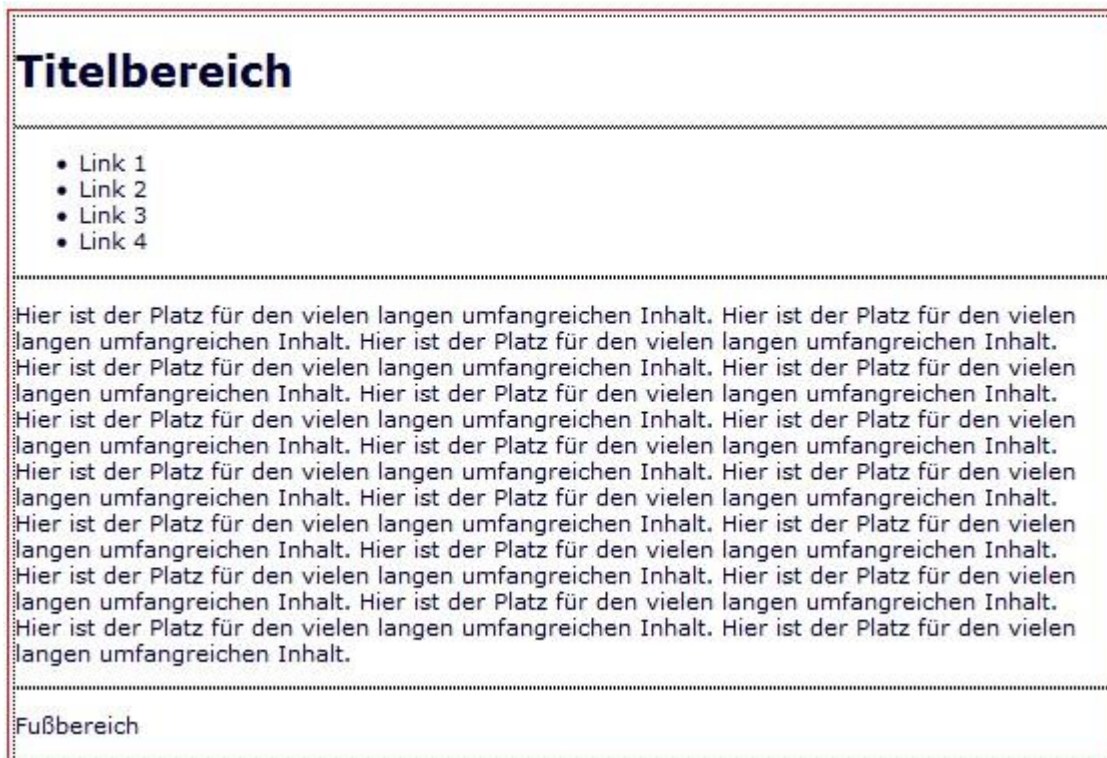


Bild 7.6: Ansicht der noch nicht positionierten Boxen mit Rahmen und Inhalt

Hinweis zum Navigationsbereich

Die zukünftigen Navigationselemente (Hyperlinks) wurden in eine Liste eingetragen. Im nächsten Abschnitt wird behandelt, wie unter Verwendung von CSS aus der Liste ein vertikales oder horizontales Menü erstellt werden kann.

Aufgabe

Erstellen Sie HTML-Datei mit den beschriebenen Boxen. Geben Sie der `container`-Box eine feste Breite, die ausreichend groß ist.

Alle Boxen sollen einen Rahmen erhalten und Inhalt. Bei dem Inhalt können Sie sich an der Musterlösung orientieren.

Musterlösung: 02_CSS/07_layouts/05_layout_01.html

2. Die inneren Boxen positionieren

Wie die inneren Boxen angeordnet werden sollen, können Sie Bild 7.5 entnehmen.

Die Positionierung der inneren Boxen erfordert nur wenig zusätzlichen Aufwand. Sie erfolgt relativ, damit der Navigations- und der Inhaltsbereich in der Höhe nicht beschränkt werden.

Box	CSS	Bemerkungen
header	<code>position: relative; height: 60px; clear: both;</code>	Der Kopfbereich soll eine feste Höhe erhalten. Er soll die volle Breite erhalten.
dir	<code>position: relative; width: 150px; float: left;</code>	Neben der dir -Box soll content -Box platziert werden.
content	<code>position: relative;</code>	Rechts neben dieser Box soll keine weitere Box platziert werden. Daher benötigt sie keine Eigenschaften zur Positionierung.
footer	<code>position: relative; clear: both;</code>	Der Fußbereich soll wieder eine eigene Zeile erhalten.

Aufgabe

Fügen Sie die angegebenen CSS-Eigenschaften in die Style-Definitionen der betreffenden Klassen ein.

Mit diesen Eigenschaften wird die folgende Anordnung erzeugt:

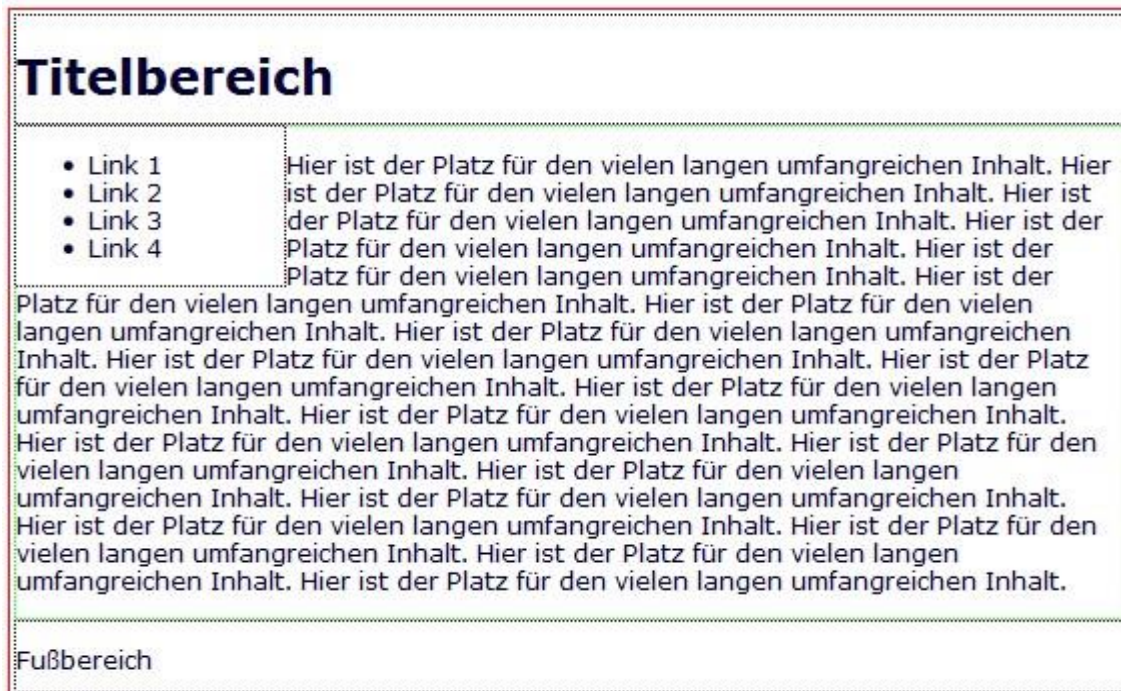


Bild 7.7: Anordnung der Boxen nach dem Einfügen der Positionierungsanweisungen

Diese Ansicht hat noch einen Schönheitsfehler. Der Text des Content-Bereichs umfließt den Navigationsbereich. Gewünscht ist, dass er nicht bis an den linken Rand geht.

Wenn Sie sich den grünen Rahmen der Content-Box ansehen, dann wird Ihnen auffallen, dass sie unten bis an den linken Rand geht.

3. Ausdehnung des Content-Bereichs nach links begrenzen

Dass die Content-Box bis an den linken Rand geht, kann nicht geändert werden. Dafür ist CSS verantwortlich. Die Lösung besteht darin, den linken Rand der Content-Box so breit zu machen, dass in diesen Rand die Navigationsbox rein passt.

Festgelegt werden soll der rechte Rand durch die CSS-Eigenschaft `margin-left`.

Aufgabe

Fügen Sie die folgende Eigenschaft in die Style-Definition der Content-Box ein. `margin: 0 0 0 155px;`

Damit ist die Erstellung des Layout abgeschlossen. Sie sollten jetzt eine Darstellung wie in Bild 7.5 erhalten.

Ein abschließender Test soll noch zeigen, ob das Layout auch bei wenig Text funktioniert:

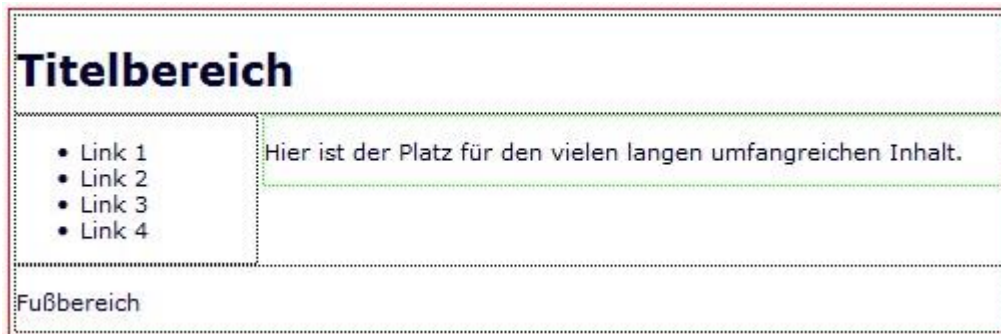


Bild 7.8: Ansicht des Layouts, wenn die dir-Box eine größere Höhe als die content-Box hat

Musterlösung: 02_CSS/07_layouts/07_a_layout_03.html

2.7.4 Navigationsleisten aus Listen erzeugen

Listen als Ausgangspunkt für Navigationsbereiche

Im zweispaltigen Layout, dass im vorherigen Abschnitt erstellt wurde, wurde als Navigationsbereich eine Liste verwendet.

Gründe für die Verwendung von Listen als Navigationsbereich

Listen werden häufig als Navigationsbereiche verwendet. Das hat die folgenden Gründe:

Barrierefreiheit

Listen werden ohne CSS-Formatierung so dargestellt, dass sie dennoch benutzbar sind. Das ist ein wichtiger Grund aus der Sicht der Barrierefreiheit.

Bereitstellung der gewünschten Boxen-Struktur

Listen bestehen aus den umschließenden `-/`-Tags. In ihnen befinden sich Hyperlinks in Listenelemente in ``-Tags. Damit stellen Listen genau die Boxenstruktur bereit, die für einen Navigationsbereich benötigt wird.

Darüber hinaus können Listen hierarchisch aufgebaut werden. Damit können auch hierarchische Navigationsstrukturen realisiert werden.

(7.6) Formatieren eines vertikalen Navigationsbereichs

Weiterführende Informationen: [SelfHTML -> CSS-basierte Navigationsleisten](#)

1. HTML-Code der Liste für den Navigationsbereich

Für den Navigationsbereich soll der folgende HTML-Code verwendet werden.

HTML-Code

```
<ul>
  <li><a href="?1">Link 1</a></li>
  <li><a href="?2">Link 2</a></li>
  <li><a href="?3">Link 3</a></li>
  <li><a href="?4">Link 4</a></li>
</ul>
```

Die Hyperlinks wurden bereits eingetragen. Sie verweisen auf die selbe Seite und geben einen Querystring mit einer Zahl zurück.

- [Link 1](#)
- [Link 2](#)
- [Link 3](#)
- [Link 4](#)

Bild 7.9: Browseransicht der Liste

Aufgabe

Erstellen Sie eine HTML-Seite und fügen Sie eine Liste nach dem Muster des dargestellten HTML-Codes ein.

Musterlösung: 02_CSS/07_layouts/08_b_navigationsbereich.html?4

2. Aufzählungszeichen aus der Liste entfernen

Die Aufzählungszeichen sollen als erstes entfernt werden. Dazu kann die CSSEigenschaft `list-style: none;` des ``-Tags genutzt werden.

[Link 1](#)
[Link 2](#)
[Link 3](#)
[Link 4](#)

Bild 7.10: Liste ohne Aufzählungspunkte

Aufgabe

Fügen Sie den Style ein, der die Aufzählungspunkte ausblendet.

3. Gestalten des Navigationsbereichs

Zur Gestaltung der Liste können die CSS-Eigenschaften für Boxen, Text und Hyperlinks verwenden.



Bild 7.11: Ansicht der Musterlösung der vertikalen Navigationsbereichs

Das obere Listenelement zeigt die Formatierung im "hover"-Zustand (Mauszeiger über dem Element) an.

Aufgabe

Formatieren Sie Ihren vertikalen Navigationsbereich.

Musterlösung: 02_CSS/07_layouts/08_c_navigationsbreich.html

(7.7) Formatieren eines horizontalen Navigationsbereichs

Um einen horizontalen Navigationsbereich zu erstellen, wird ebenfalls eine Liste verwendet. Der Unterschied zum vertikalen Navigationsbereich besteht darin, dass die ``-Tags nebeneinander angeordnet sein sollen. Die Listenelemente können durch die CSS-Eigenschaft `float: left` nebeneinander angeordnet werden.



Bild 7.12: Vertikaler Navigationsbereich auf Basis einer Liste

(Bild 7.12) zeigt den horizontalen Navigationsbereich, der aus dem vertikalen Navigationsbereich (Bild 7.11) erstellt wurde.

Aufgabe

Ändern Sie den Navigationsbereich aus der letzten Aufgabe so ab, dass daraus ein horizontaler Navigationsbereich wird.

Musterlösung: 02_CSS/07_layouts/09_navigationsbreich.html

2.7.5 Vorlagen für mehrspaltige CSS-Layouts

(7.8) Verbessertes zweispaltiges Layout

In Abschnitt [2.7.3](#) dieses Kapitels wurde ein einfaches, aber funktionsfähiges zweispaltiges Layout erstellt.

Dieses Layout besitzt keine speziellen Anpassungen für Besonderheiten älterer Browser. Ich möchte Ihnen in diesem Kapitel ein verbessertes zweispaltiges Layout vorstellen, die die Abweisungen des fehlerhaften Boxenmodells älterer Internetexplorer korrigiert.

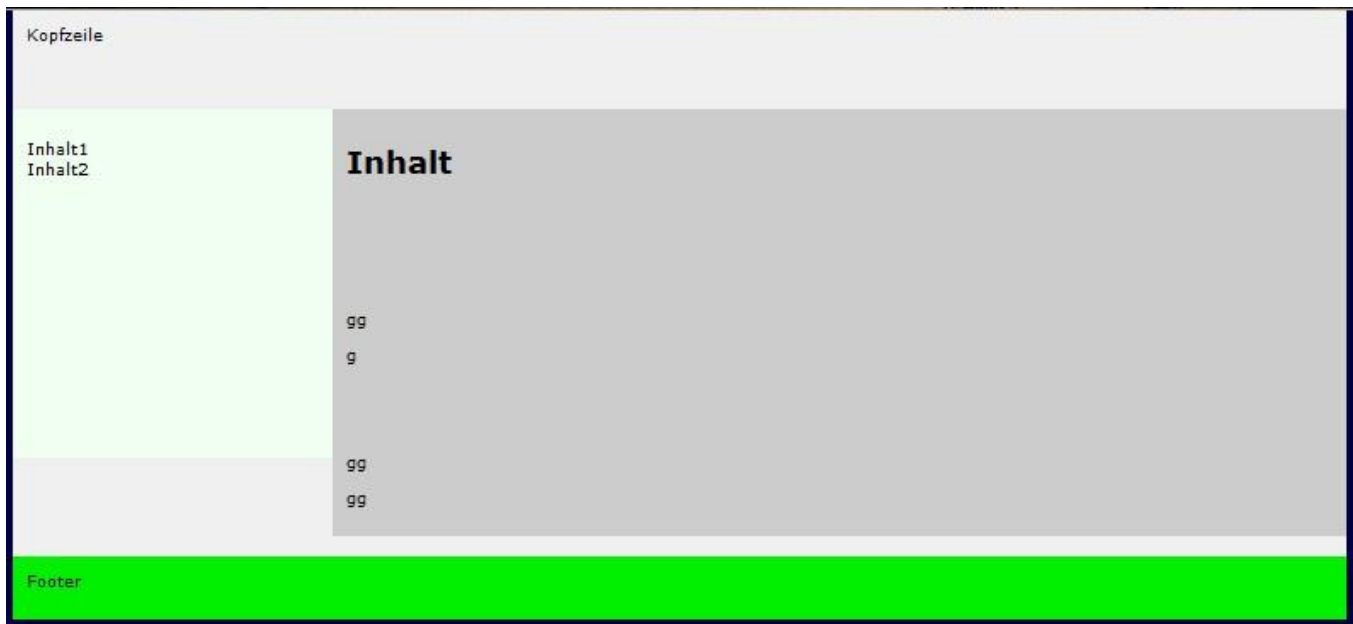


Bild 7.13: Ansicht des verbesserten zweispaltigen Layouts

Das Layout wird in Bild 7.13 dargestellt. Damit die Darstellung der einzelnen Boxen nachvollzogen werden kann, haben alle Boxen unterschiedliche Hintergrundfarben erhalten.

Download des Layouts

Sie können sich das Layout [hier herunterladen](#).

Aufgabe 1

Laden Sie das Layout herunter: Wandeln Sie die herunter geladene Textdatei in eine HTML-Datei um, indem Sie die Namenserweiterung ändern.

Machen Sie sich mit der Boxenstruktur des Layouts vertraut und sowie mit den CSSStyles, die in die Datei integriert sind,

Testen Sie das Layout. Fügen Sie in die Navigationsleiste eine Liste für Hyperlinks ein und ändern Sie die Hintergrundfarben der Boxen.

Aufgabe 2

Lagern Sie die CSS-Styles in eine externe CSS-Datei aus.

Beschaffung weiterer Layouts

Falls Sie weitere Layouts, insbesondere auch ein dreispaltiges Layout, benötigen, empfehle ich Ihnen, sich den HTML-Editor Dreamweaver von Adobe herunterzuladen. Adobe bietet auf seiner Website den Download einer 30-Tage-Testversion an, die voll funktionsfähig ist.

Dreamweaver stellt viele unterschiedliche Layouts bereit, bei denen zu den CSS-Styles umfangreiche Erläuterungen und Hinweise angegeben werden.

2.7.6 Kapitelschwerpunkte

2.7.7 Übungen

Die folgenden beiden Übungen zielen darauf ein, zu demonstrieren, wie die Ansicht desselben HTML-Codes nur durch unterschiedliche CSS-Formatierungen geändert werden kann.

Diese beiden Aufgaben würde ich eher in die Kategorie "Quickies", d. h. kleine Aufgaben, einordnen. Die Änderungen in den CS-Styles sind minimal. Wichtig ist, dass Sie sich überlegen, welche Änderungen erforderlich sind.

(7.9) Übung: Einspaltiges Layout mit horizontalem Navigationsbereich

Ändern Sie das fertig gestellte Layout aus dem Workshop 7.5 (Musterlösung: 02_CSS/07_layouts/07_layout_03.html) so, dass der Navigationsbereich als vertikales Menü zwischen Kopf- und Inhalt-Bereich angezeigt wird. Der Inhaltsbereich soll die gesamte verfügbare Breite ausnutzen.



Bild 7.14: Ansicht der Musterlösung

Musterlösung: [02_CSS/07_layouts/10_layout_horizontal.html](#)

(7.10) Übung: Zweispaltiges Layout mit vertikalem Navigationsbereich auf der rechten Seite

Verwenden Sie auch für die Übung die Musterlösung [02_CSS/07_layouts/07_a_layout_03.html](#) aus dem Workshop 7.5. In dieser Übung soll die Navigationsleiste auf der rechten Seite angezeigt werden.



Bild 7.15: Ansicht der Musterlösung

Musterlösung: 02_CSS/07_layouts/11_layout_rechts.html