1 CSS Grundlagen

1.1 Vorbemerkungen

1.1.1 Abgrenzung

In diesem Kapitel werden Grundlagen zu CSS aus den Standards CSS1 und CSS2 behandelt.

1.1.2 Syntaxdefinition

CSS-Eigenschaften und CSS-Stilregeln werden mit einer an die [EBNF] angelehnten Notation definiert. Diese Notation enthält folgende Bestandteile:

- das zu beschreibende Element steht links von der Zeichenfolge `::=`
- die Bestandtile des zu beschreibenden Elements stehen rechts von der Zeichenfolge `::=`
- rechts können auftreten:
  - Reihungen (die Bestandteile sind durch Leerzeichen getrennt und werden von links nach rechts aufgeführt)
  - Alternativen, die in eckigen Klammern aufgeführt werden und untereinander durch das Zeichen | getrennt werden; wenn eine Alternative nur einen Eintrag enthält, ist es ein Option (d.h. der Eintrag kann, muss aber nicht auftreten)
  - Wiederholungen, die in geschweiften Klammern aufgeführt werden
  - Zeichenketten, die als solche auftreten (z.B. Schlüsselworte), werden in Anführungszeichen (") gesetzt.

1.2 Einleitung

Ein Grundprinzip der Software-Technik lautet : Trennung von Inhalt und Benutzungsoberfläche. Inhalte sollen sich nicht ändern, wenn sich die Art der Präsentation ändert.

Unterschiedliche Präsentationsformen können z.B. durch unterschiedliche Ausgabemedien - Anzeige einer Webseite im Webbrowser auf dem Bildschirm des Arbeitsplatzrechners oder Ausdruck der Webseite - oder durch unterschiedliche Gestaltungswünsche - z.B. unterschiedliche Schriftarten oder Farben - erforderlich sein.

In früheren Versionen von HTML weisen viele Elemente Eigenschaften auf, die nicht oder nicht nur die Struktur des HTML-Dokuments festlegen, sondern sich auch auf die Darstellung beziehen. Inhalt und und Darstellung sind miteinander verzahnt, gegen das oben genannte Prinzip wird also verstoßen. Mit XHTML und HTML5 wird versucht, auf Elemente zu verzichten, die sich in erster Linie auf die Präsentation beziehen. Vollständig erreicht wird das aber nicht.

Das W3-Konsortium W3C hat 1996 die Sprache CSS eingeführt, mit der festgelegt wird, wie ein HTML-Dokument dargestellt wird. Die Referenz zu CSS finden Sie hier : W3C-CSS-Standards.

Einfache Beispiele zur Bedeutung der Trennung von Inhalt und Präsentation werden in Struktur und Präsentation trennen erläutert.

CSS verwendet eine spezielle Beschreibungsform für die Festlegung der Präsentation von HTML-Elementen. Eine erste Einführung dazu finden Sie in CSS-Stilregeln. Dort wird auch erläutert, wie bei einem HTML-Dokument festgelegt wird, welche CSS-Regeln angewendet werden.

Wie man mit Hilfe von CSS die Verwendung von Schriften und Farben steuert, den Hintergrund gestaltet und Texte formatiert, erfahren Sie in CSS-Stilregeln für Schrift, Farbe, Hintergrund und Textformatierung.

CSS bietet vielfältige Möglichkeiten zu spezifizieren, wie die CSS-Regeln auf HTML-Elemente angewendet werden. Eine erste Vertiefung der Angaben aus CSS-Stilregeln finden Sie in CSS-Stilregeln für Schrift, Farbe, Hintergrund und Textformatierung.

Die Anordnung von Texten und Bildern kann ebenfalls durch CSS-Angaben gesteuert werden. Ausgangspunkt ist das sog. Box-Modell, das in Das Box-Modell eingeführt wird, woran sich die Erklärung der Möglichkeiten zur Positionierung von HTML-Elementen in HTML-Elemente mit CSS positionieren anschließt.

1.3 Struktur und Präsentation trennen

Die Auszeichnungssprache HTML wurde ursprünglich zur Beschreibung der Struktur von Dokumenten geschaffen. Außerdem ermöglicht sie die Referenzierung von Dokumenten durch Hyperlinks.

Mit HTML-Elementen wie `h1` und `p` kann ein Text strukturiert und den einzelnen Textbestandteilen eine Bedeutung innerhalb des HTML-Dokuments zugewiesen werden.

Beispiel

```
<h1>Kapitel 1</h1>
   <h2>Erster Abschnitt</h2>
      <p>Der Inhalt des ersten Abschnitts (Kapitel 1).</p>
   <h2>Zweiter Abschnitt</h2>
      <p>Der erste Absatz des zweiten Abschnitts (Kapitel 1).</p>
      <p>Der zweite Absatz des zweiten Abschnitts (Kapitel 1).</p>
<h1>Kapitel 2</h1>
   <h2>Erster Abschnitt</h2>
      <p>Der Inhalt des ersten Abschnitts (Kapitel 2).</p>
   <h2>Zweiter Abschnitt</h2>
      <p>Der Inhalt des zweiten Abschnitts (Kapitel 2).</p>
```

Zur Vereinfachung sind die weiteren Bestandteile des HTML-Dokuments weggelassen worden.

Die im Beispiel angegebenen Auszeichnungen (HTML-Elemente) definieren die Hierarchie des Dokuments, die durch Einrückung der HTML-Elemente auch optisch verdeutlicht wird.

Ohne weitere Angaben verwendet ein Webbrowser spezielle Voreinstellungen zur Darstellung der einzelnen Elemente. So werden `h1`-Überschriften in der Regel in einer größeren Schriftart dargestellt als `h2`-Überschriften.

Häufig besteht die Notwendigkeit, in einem Textabschnitt (HTML-Element `p`) einen Textbestandteil aufgrund seiner **Bedeutung** hervorzuheben. Dazu gibt es die HTML-Elemente `em` und `strong` zur Betonung oder deutlichen Kennzeichnung.

Als Relikt aus den Anfangszeiten von HTML ohne zusätzliche Möglichkeiten zur Beschreibung der Präsentation mit CSS gibt es in HTML5 auch noch die Elemente `b` (ursprünglich im Sinne von bold : Fettdruck einer Textpassage) und `i` (ursprünglich im Sinne von italic : Kursivdruck einer Textpassage). In der HTML5-Spezifikation wird der halbherzige Versuch unternommen, beiden Elementen eine neue Bedeutung zu geben. Beide Elemente können zwar wie `em` und `strong` zur Verdeutlichung der Bedeutung eines Textabschnittes verwendet werden, mit beiden ist aber auch die Erwartung an eine konkrete Darstellungsform, eben als fett oder kursiv gedruckter Text, verbunden. An dieser Stelle sind in HTML5 Dokumentstrukturierung und Präsentation miteinander verzahnt, das Prinzip der Trennung von Struktur und Präsentation wird verletzt.

Beispiel : Hervorhebungen

```
<!DOCTYPE html>
<html>
<head>
   <title>Hervorhebungen</title>
   <meta charset="UTF-8" />
</head>
<body>
```

```
8       <p>
9          Besonders <em>wichtige</em> Textbestandteile
10         sollten <strong>hervorgehoben</strong> werden.
11      </p>
12      <p>
13         Besonders <b>wichtige</b> Textbestandteile
14         sollten <i>hervorgehoben</i> werden.
15      </p>
16   </body>
17   </html>
```

em und strong statt b und i

An der Ausgabe im Webbrowser kann man gut die Probleme beim Einsatz z.B. des HTML-Elements b erkennen : wie soll man sich entscheiden ? Sollen Betonungen durch Fettschrift oder durch Kursivschrift verdeutlicht werden ? Wenn zu einem späteren Zeitpunkt eine andere Vereinbarung getroffen wird, müssen bei der Verwendung der HTML-Elemente b und i **alle** HTML-Dokumente überarbeitet werden ! Mit den Mitteln von CSS geht man einen anderen Weg: man beschreibt für die HTML-Elemente em und strong die Art der Darstellung **unabhängig** von HTML-Dokumenten und kann daher diese Beschreibung jederzeit überarbeiten, ohne HTML-Dokumente verändern zu müssen.

Hervorhebungen: CSS verwenden

```
1    <!DOCTYPE html>
2    <html>
3    <head>
4       <title>Hervorhebungen</title>
5       <meta charset="UTF-8" />
6       <style type="text/css">
7          em     { font-style:normal;
8                   font-weight:bold;}
9          strong { font-style:italic;
10                  font-weight:normal;}
11      </style>
12   </head>
13   <body>
14      <p>
15         Besonders <em>wichtige</em> Textbestandteile
16         sollten <strong>hervorgehoben</strong> werden.
17      </p>
18      <p>
19         Besonders <b>wichtige</b> Textbestandteile
20         sollten <i>hervorgehoben</i> werden.
21      </p>
22   </body>
23   </html>
```

Im Abschnitt <style ...> ... </style> wird mit den Mitteln von CSS festgelegt, dass betonte Textteile (HTML-Element em) fett und hervorgehobene Textteile (HTML-Element strong) kursiv dargestellt werden, also die Voreinstellungen des Webbrowsers gerade umgekehrt werden.

Aufbau und Verwendung der Stilregeln wird in CSS-Stilregeln erläutert.

Ein weiteres Beispiel für ein HTML-Element, das in den meisten Fällen zur Beeinflussung der Darstellungsweise verwendet wird, ist das HTML-Element br. Dieses Element bewirkt eine Art Zeilenumbruch und wird daher oft zur Erzeugung zusätzlichen Leerraums z.B. unterhalb eines Textabschnitts verwendet.

Leerraum durch `br`

```
<!DOCTYPE html>
<html>
<head>
   <title>Leerraum durch br-Element</title>
   <meta charset="UTF-8" />
</head>
<body>
   <p>
      Zwei direkt aufeinander folgende Abs&auml;tze.
   </p>
   <p>
      Absatz 2.
   </p>
   <p>
      Zwei direkt aufeinander folgende Abs&auml;tze
      mit zus&auml;tzlicher Leerzeile.
   </p>
   <br/>
   <p>
      Absatz 2.
   </p>
</body>
</html>
```

Mit CSS schreib man stattdessen besser:

Leerraum mit CSS

```
<!DOCTYPE html>
<html>
<head>
   <title>Leerraum durch Margin</title>
   <meta charset="UTF-8" />
   <style type="text/css">
      #P3 { margin-bottom:3em; }
   </style>
</head>
<body>
   <p>
      Zwei direkt aufeinander folgende Abs&auml;tze.
   </p>
   <p>
      Absatz 2.
   </p>
   <p id='P3'>
      Zwei direkt aufeinander folgende Abs&auml;tze
      mit zus&auml;tzlichem Leerraum.
   </p>
   <p>
      Absatz 2.
   </p>
</body>
</html>
```

Im Abschnitt `<style ...> ... </style>` wird mit den Mitteln von CSS festgelegt, dass unterhalb des Absatzes mit der Identifikation `P3` ein Rand erzeugt wird. Der Rand ist durch die Angabe `1em` so groß wie die Schriftgröße des Absatzes. `em` ist hier also eine relative Größenangabe und **nicht** das HTML-Element `em` !

Aufbau und Verwendung der Stilregeln wird in CSS-Stilregeln erläutert.

Tipp

Durch den Einsatz von CSS ist es möglich, HTML-Elemente wie `b` und `i`, die vor allem der Steuerung der Präsentation dienen, zu vermeiden. Sehen Sie stattdessen CSS-Regeln vor. Vermeiden Sie möglichst auch den Einsatz des HTML-Elements `br` zur Beeinflussung der Darstellung.

Vorteile

Die Vorteile des Einsatzes von CSS sind bereits an diesen einfachen Beispielen klar erkennbar:

- die Dokumentstruktur wird ausschließlich mit HTML-Elementen erzeugt, die Art der Präsentation in einem Webbrowser ausschließlich durch die Mittel von CSS spezifiziert
- Präsentation und Struktur können unabhängig voneinander verändert werden. Insbesondere ist es möglich, verschiedene Darstellungen für identische Dokumentstrukturen zu verwenden.

1.4 CSS-Stilregeln

1.4.1 Aufbau von CSS-Stilregeln

Im einfachsten Fall besteht eine CSS-Stilregel aus :

- der Angabe, auf welches HTML-Element sich die CSS-Stilregel bezieht
- einer oder mehrerer CSS-Eigenschaften, die durch den Eigenschaftsnamen und den Eigenschaftswert definiert werden.

Die Zusammenfassung von mehreren CSS-Stilregeln nennt man eine CSS-Stilvorlage. In Anlehnung an eine Zusammenstellung von Layoutvorgaben auf einem Blatt Papier nennt man eine CSS-Stilvorlage auch ein *Stylesheet*.

Es ist zulässig, bei einer CSS-Stilregel **keine** CSS-Eigenschaft anzugeben.

1.4.2 Notation : CSS-Eigenschaften definieren

Syntax

Stile werden als CSS-Eigenschaften werden stets so notiert :

```
CSS-Eigenschaft ::= CSS-Eigenschaftsname ":" { CSS-Eigenschaftswert } ";"
```

Eigenschaftsname und Eigenschaftswerte werden durch einen Doppelpunkt getrennt, die Definition einer Eigenschaft wird durch ein Semikolon abgeschlossen. Je nach CSS-Eigenschaft können auch mehrere Eigenschaftswerte auftreten.

Es wird zwischen Klein- und Großschreibung unterschieden. Sie sollten daher CSS-Eigenschaftsnamen immer und CSS-Eigenschaftswerte nach Möglichkeit klein schreiben.

Es können wie in den Programmiersprachen `C++` oder `Java` Kommentare eingefügt werden, die mit der Zeichenfolge `/*` eingeleitet und mit der Zeichenfolge `*/` abgeschlossen werden und sich über mehrere Zeilen erstrecken dürfen. Der Inhalt der Kommentare wird durch den Webbrowser nicht ausgewertet.

Beispiele

```
1       /* legt die Schriftfarbe fest */
2       color            : white;
3    ..
4       /* legt die Hintergrundfarbe fest */
5       background-color : blue;
```

1.4.3 Selektoren : Bezug zu HTML-Elementen angeben

Der Bezug einer Stilregel zu einem HTML-Element wird durch den sog. **Selektor** hergestellt. Dabei sind mehrere Fälle zu unterscheiden:

- der Selektor ergibt sich aus dem Kontext, in dem die Stilregel notiert wird (siehe Stilregeln einbinden)
- als Wert des Selektors wird die Bezeichnung eines HTML-Elements angegeben, z.B. `p` (siehe Stilregeln einbinden); die Stilregel wird dann auf alle HTML-Elemente, die in einem HTML-Dokument auftreten und dem Selektor entsprechen, angewendet
- als Wert des Selektors wird die Identifikation eines HTML-Elements (HTML-Element Attribut `id`) im HTML-Dokument angegeben; da die Identifikation eindeutig im HTML-Dokument sein soll, wird die Stilregel ausschließlich auf das eine identifizierte HTML-Element angewendet
- als Wert des Selektors können auch Bezeichnungen verwendet werden, die einem oder mehreren HTML-Elementen durch das Attribut `class` zugewiesen werden; die Stilregel wird dann auf alle HTML-Elemente in einem HTML-Dokument angewendet, die denselben Attributwert bei `class` aufweisen
- der Selektor spezifiziert komplexere **Pfade**, anhand derer entschieden wird, auf welche Teile der Dokumentstruktur die Stilregel anzuwenden ist; weitere Einzelheiten dazu werden in einem anderen Dokument behandelt.

Erweiterte Notation

Wenn ein Selektor verwendet wird, wird die oben angegebene Notation für Stilregeln erweitert: auf den Selektor folgt eine Liste der CSS-Eigenschaften, die für diesen Selektor verwendet werden sollen; die Liste wird durch geschweifte Klammern begrenzt. Weitere Einzelheiten dazu werden in einem anderen Dokument behandelt.

Syntax

CSS-Stilregeln werden stets durch einen Selektor eingeleitet, gefolgt von einer Liste von CSS-Eigenschaften, die durch geschweifte Klammern begrenzt wird:

```
CSS-Stilregel ::= Selektor { CSS-Eigenschaft }
```

Schreibweise der Selektoren

Selektoren für gleichartige HTML-Elemente erhalten als Wert die Bezeichnung des HTML-Elements (z.B. `p`). Selektoren für identifizierte HTML-Elemente erkennt man an dem vorangestellten Zeichen #, auf das die Identifikation folgt. Selektoren für CSS-Klassen wird ein Punkt vorangestellt.

In den folgenden Beispielen werden zur Vereinfachung die Stilregeln in der erweiterten Notation, die entsprechenden Ausschnitte aus den HTML-Dateien und die daraus resultierenden Darstellungen im Webbrowser dargestellt.

Beispiel Stilregel für gleichartige HTML-Elemente

```
<!DOCTYPE html>
<html>
<head>
   <title>Stilregeln für gleichartige HTML-Elemente</title>
   <meta charset="UTF-8" />
   <style type="text/css">
      /* Stilregel */
      p { color : red; } /* in allen Absätzen rote Schrift */
   </style>
</head>
<body>
   <p>erster Absatz</p>
   <p>zweiter Absatz</p>
   <p>dritter Absatz</p>
</body>
</html>
```

Der Text wird in **allen** Absätzen in roter Schrift ausgegeben. Eine besondere Kennzeichnung der HTML-Elemente ist nicht erforderlich.

Wenn der erste Absatz dagegen in einer anderen Farbe dargestellt werden soll, kann mit Hilfe der eindeutigen Identifikation genau diesem Absatz eine eigene Stilregel zugewiesen werden.

Beispiel Stilregel für identifizierte HTML-Elemente

```
<!DOCTYPE html>
<html>
<head>
   <title>Stilregeln für gleichartige HTML-Elemente</title>
   <meta charset="UTF-8" />
   <!-- Ausschnitt HTML-Dokument : head -->
   <style type="text/css">
      /* Stilregel */
      /* in allen Absätzen rote Schrift */
      p { color : red; }
      /* nur in diesem Absatz grüne Schrift */
      #ErsterAbsatz { color : green; }
   </style>
</head>
<body>
   <!-- Ausschnitt HTML-Dokument : body -->
   <p id='ErsterAbsatz'> erster Absatz</p>
   <p>zweiter Absatz</p>
   <p>dritter Absatz</p>
</body>
</html>
```

In diesem Fall muss einerseits eine zusätzliche Stilregel verwendet und andererseits der Absatz durch eine eindeutige Identifikation gekennzeichnet werden.

Beispiel CSS-Klassen

```
<!DOCTYPE html>
<html>
<head>
```

```
4       <title>Stilregeln für gleichartige HTML-Elemente</title>
5       <meta charset="UTF-8" />
6       <!-- Ausschnitt HTML-Dokument : head -->
7       <style type="text/css">
8          /* Stilregel */
9          /* in allen Absätzen rote Schrift */
10         p { color : red; }
11         /* nur in diesen Absätzen grüne Schrift */
12         .Wichtig { color : green; }
13      </style>
14   </head>
15   <body>
16      <!-- Ausschnitt HTML-Dokument : body -->
17      <p class='Wichtig'> erster Absatz</p>
18      <p>zweiter Absatz</p>
19      <p class='Wichtig'>dritter Absatz</p>
20   </body>
21   </html>
```

Sie sehen, dass CSS-Klassen vor allem dazu geeignet sind, mehreren HTML-Elementen gemeinsame Eigenschaften zuzuweisen. Das Prinzip der Trennung von Struktur und Präsentation bleibt erhalten : im HTML-Dokument wird nur die Gemeinsamkeit der HTML-Elemente durch denselben Klassennamen (Attribut `class`) ausgedrückt, die Art der Darstellung wird ausschließlich durch die Stilregeln festgelegt. Würde man z.B. zur Hervorhebung der wichtigen Absätze statt einer anderen Farbe eine andere Schriftart verwenden wollen, würde nur die Stilregel geändert. Das HTML-Dokument bliebe unverändert.

1.4.4 Stilregeln einbinden

Es gibt verschiedene Möglichkeiten, CSS-Stilregeln in HTML-Dokumenten zu verwenden. Empfohlen wird die Zusammenfassung der CSS-Stilregeln einer Website in zentralen CSS-Dateien. Dadurch können Inhalt und Präsentation konsequent getrennt und die Wartung der CSS-Stilregeln vereinfacht werden.

CSS-Stilregeln müssen dem Webbrowser zur Anwendung auf ein HTML-Dokument zur Verfügung gestellt werden. Dazu gibt es unterschiedliche Möglichkeiten:

- Inline-Styles
- Interne Stylesheets
- Externe Stylesheets.

1.4.4.1 Inline-Styles

Inline-Styles sind CSS-Stileigenschaften, die unmittelbar bei HTML-Elementen notiert werden. Dazu wird das Attribut `style` benutzt, dessen Wert die Liste der CSS-Stileigenschaften ist. Die CSS-Stileigenschaften werden in der in Aufbau von CSS-Stilregeln erläuterten Notation angegeben.Da der Bezug zum HTML-Element direkt hergestellt wird, entfällt die Angabe des Selektors.

Beispiel

```
1       <p style="border:1px solid red;">
2          Absatz mit rotem Rand
3       </p>
```

Inline-Styles gelten nur für das HTML-Element, in dessen `style`-Attribut sie aufgeführt werden. Sollen mehrere HTML-Elemente mit identischen Layout-Eigenschaften versehen werden, müssen die Inline-Styles für jedes HTML-Element einzeln angeben werden. Das ist sehr umständlich und fehleranfällig und führt bei späteren Änderungen zu einem erheblichen Aufwand.

Inline-Styles vermeiden

Inline-Styles sollten weitgehend vermieden werden. Allenfalls zum Ausprobieren von CSS-Stileigenschaften während der Erstellung und des Tests von HTML-Dokumenten könnte ein Einsatz sinnvoll sein. Wenn Sie einem einzelnen HTML-Element spezielle CSS-Stileigenschaften zuweisen wollen, verwenden Sie stattdessen besser eine CSS-Stilregel mit der Identifikation des HTML-Elements als Selektor.

1.4.4.2 Interne Stylesheets

Interne Stylesheets erlauben die Zusammenfassung der CSS-Stilregeln für ein HTML-Dokument. Sie werden daher auch **eingebettete** Stylesheets genannt. Interne Stylesheets werden mit Hilfe des HTML-Elements `style` eingeführt, das stets im Kopfbereich eines HTML-Dokuments eingetragen werden muss.

Beispiel

```
1   <!DOCTYPE html>
2   <html>
3   <head>
4      <title>Interne Stylesheets</title>
5      <meta charset="UTF-8" />
6      <style type="text/css">
7         p { font-family: Courier, monospace; }
8      </style>
9   </head>
10  <body>
11     <p>Absatz</p>
12  </body>
13  </html>
```

Durch die CSS-Stilregel im HTML-Element `style` wird in allen Textabschnitten (HTML-Element `p`) des HTML-Dokuments die Schriftart Courier verwendet. Falls diese nicht verfügbar ist, wird eine Schriftart des Client-Systems verwendet, die der generischen Schriftart `monospace` soweit wie möglich entspricht.

Das HTML-Element `style` verfügt über verschiedene Eigenschaften:

- die Eigenschaft `type` ist zwingend erforderlich und gibt an, welcher Mime-Type für das Element vorliegt
  - damit erfährt der Webbrowser, wie er die Angaben verarbeiten muss
  - z.Zt. ist nur die Angabe `type="text/css"` sinnvoll, da es im praktischen Einsatz nur CSS-Stilregeln gibt.

Interne Stylesheets bieten den Vorteil einer zentralen Stelle zur Aufnahme der in einem HTML-Dokument gültigen Stilregeln. Sobald aber mehrere HTML-Dokumente einer Website nach einheitlichen Anforderungen gestaltet werden sollen, sind sie ebenfalls ein untaugliches Mittel, da bei Änderungen wiederum die HTML-Dokumente angepasst werden müssten. Der Einsatz von internen Stylesheets sollte daher auf den Sonderfall einzelner unabhängig verwendeter HTML-Dokumente beschränkt werden.

1.4.4.3 Externe Stylesheets

Externe Stylesheets sind eine Zusammenfassung von CSS-Stilregeln in eigenständigen Dateien, die mit speziellen Anweisungen den HTML-Dokumenten zugewiesen werden müssen. In diesen Dateien werden die CSS-Stilregeln genauso notiert wie im zuvor erläuterten Fall der Inline-Stylesheets, d.h. in der erweiterten Notation mit Selektoren.

Es gibt mehrere Möglichkeiten zur Einbindung externer CSS-Stylesheets in ein HTML-Dokument, die beiden wichtigsten Formen sind:

- Einbindung mit Hilfe des HTML-Elements `link`

- Einbindung mit Hilfe der speziellen CSS-Regel `@import`.

Einbinden mit `link`

Das HTML-Element `link` muss im Kopfbereich eines HTML-Dokuments angeordnet werden. Um ein CSS-Stylesheet einzubinden, müssen folgende Attribute verwendet werden:

- `href` enthält die URL der externen Stylesheet-Datei, in der Regel relativ zum HTML-Dokument
- `type` wie bei internen Stylesheets; in der Praxis z.Zt. immer `"text/css"`
- `rel` zur Spezifikation der Art des Bezugs durch das HTML-Element `link`, in diesem Fall der Wert `StyleSheet`.

Einbinden mit `@import`

Die Einbindung von Stilregeln mit Hilfe der CSS-Regel `@import` ist die neuere Technik, die von allen aktuellen Webbrowsern unterstützt wird. Die Regel muss innerhalb eines HTML-Elements `style` im Kopfbereich der HTML-Dokumente aufgeführt werden.

Die Syntax der CSS-Regel `@import` legt fest, dass auf das Schlüsselwort der URI als Argument der `url`-Funktion folgt:

```
CSSImport ::= "@import" "url" "(" URI ")" ";"
```

Mit beiden Varianten - HTML-Element `link` und CSS-Regel `import` können mehrere externe Stylesheets gleichzeitig in einem HTML-Dokument verwendet werden.

Man bevorzugt heutzutage (HTML5) das Einbinden per `link`, um die Beziehung zu einer externen Ressource deutlicher herauszustellen. Außerdem wird dann die externe CSS-Datei geladen, sobald der HTML5-Parser das `link`-Element erkennt. Die Anweisung `@import` wird als CSS-Anweisung erst durch den CSS-Prozessor ausgewertet.

Das nächste Beispiel zeigt die Einbindung mit Hilfe des HTML-Elements `link`.

Externe Stylesheets mit `link` einbinden

```
1   <!DOCTYPE html>
2   <html>
3   <head>
4      <title>Externe Stylesheets</title>
5      <meta charset="UTF-8" />
6      <link rel="StyleSheet" href="extern.css" type="text/css" />
7   </head>
8   <body>
9      <p class='Wichtig'>Absatz 1</p>
10     <p>Absatz 2</p>
11  </body>
12  </html>
```

Die Einbindung per CSS-Regel `@import` ist Gegenstand des folgenden Beispiels.

Externe Stylesheets mit `@import` einbinden

```
1   <!DOCTYPE html>
2   <html>
3   <head>
4      <title>Externe Stylesheets</title>
5      <meta charset="UTF-8" />
```

```
6       <style type="text/css">
7          @import url(extern.css);
8       </style>
9    </head>
10   <body>
11      <p class='Wichtig'>Absatz 1</p>
12      <p>Absatz 2</p>
13   </body>
14   </html>
```

Bei beiden Beispiel wird diese CSS-Datei mit Stilregeln verwendet:

Externe Stilregeln

```
1    /* Stilregel */
2    /* in allen Absätzen grüne Schrift */
3    p { color : green; }
4    /* nur in diesen Absätzen rote Schrift */
5    .Wichtig { color : red; }
```

Tipp

Es ist zulässig, beide Verfahren gleichzeitig zu verwenden. Eine klare Struktur der HTML-Dokumente und damit eine bessere Wartbarkeit erreichen Sie jedoch durch die Verwendung eines HTML-Elements `style` und darin eingebetteter `@import`-Regeln.

Tipp: `@import` voranstellen

Wenn Sie externe CSS-Stylesheets mit `@import` einbinden, müssen alle `@import`-Regeln **vor** allen anderen CSS-Stilregeln im HTML-Element `style` stehen.

1.4.4.4 Prioritätsregeln

In einem HTML-Dokument dürfen die beschriebenen Mechanismen zur Einbindung von CSS-Stilegeln gleichzeitig eingesetzt werden. Dann gelten folgende Prioriäten bei der Auswahl der anzuwendenden CSS-Stilregeln:

- höchste Priorität haben **Inline-Styles**, d.h. die bei einem HTML-Element notierten CSS-Stilregeln **überschreiben** entsprechende CSS-Stilregeln aus anderen Quellen
- eine geringere Priorität haben **Interne Stylesheets**, die in `style`-Abschnitten notiert werden; sie überschreiben aber die entsprechenden Angaben in externen Stylesheets
- CSS-Stilregeln in **Externen Stylesheets** sind nur dann wirksam, wenn es keine passenden CSS-Stilregeln in Internen Stylesheets oder Inline-Styles gibt.

Wenn es keine explizit notierte CSS-Stilregel für ein HTML-Element gibt, also weder inline, intern noch extern, werden die Voreinstellungen des Webbrowser wirksam. Im Prinzip handelt es sich dabei um ein spezielles CSS-Stylesheet, das als Standardvorgabe verwendet wird. In der CSS-Spezifikation des W3C wird ein solches CSS-Stylesheet vorgeschrieben: das sog. “user agent stylesheet”.

Diese Reihenfolge der Prioritäten wird auch **cascading order** genannt, woraus sich auch der Namensbestandteil **cascading** bei CSS ableitet.

1.4.4.5 Stylesheets organisieren

CSS-Stilregeln sollten in externen CSS-Stylesheets zusammengefasst werden. Bei einfachen Websites ist es ratsam, eine zentrale Datei als externes CSS-Stylesheet zu verwenden und dort alle CSS-Stilregeln zu notieren.

Bei komplexeren Websites ist diese Vorgehensweise ungeeignet. Dort ist es sinnvoller, die externen CSS-Stylesheets nach ihrer Bedeutung auf verschiedene Dateien aufzuteilen:

- CSS-Stilregeln, die in allen HTML-Dokumenten verwendet werden sollen, werden in einem Basis-Stylesheet zusammengefasst
- CSS-Stilregeln, die nur in einem Teil der HTML-Dokumente zum Einsatz kommen, werden so zusammengefasst, dass quasi einzelne CSS-Bausteine entstehen.

Die Aufteilung der CSS-Stylesheets erleichtert auch die Wiederverwendung in weiteren Websites.

1.5 CSS-Stilregeln für Schrift, Farbe, Hintergrund und Textformatierung

CSS-Stilregeln werden häufig eingesetzt, um Texte und Textabschnitte zu gestalten. Die wesentlichen Eigenschaften, die in CSS für Schrift, Farbe, Hintergrund und Textformatierung, verwendet werden können, sind bereits seit der ersten CSS-Spezifikation (CSS1) des W3C verfügbar.

1.5.1 Schrift

Regel `font` und Spezialisierungen

Schriften und Schrifteigenschaften werden durch die CSS-Eigenschaft `font` und Spezialisierungen dieser Regel, deren Bezeichnungen alle mit `font-` beginnen, eingestellt. Es stehen folgende Möglichkeiten zur Verfügung:

- `font-family`: **Schriftfamilie**, mit der die typographischen Grundeigenschaften einer Schrift angesprochen werden; dabei sind 2 Fälle zu unterscheiden:
  - Angabe einer generischen Schriftfamilie (`serif, sans-serif, cursive, fantasy, monospace`)
  - Angabe eines Schriftnamens (z.B. `Times`)
- `font-style`: **Schriftstil** mit den Varianten `normal, italic, oblique`
  - Voreinstellung ist `normal`
  - `italic` steht für kursiv, `oblique` **für schräg gestellt**

- `font-size`: Einstellung der Schriftgröße, die relativ oder absolut angegeben werden kann

- `font-variant`: mit `small-caps` können sog. **Kapitälchen** erzielt werden, bei denen alle Buchstaben groß ausgegeben werden, die eigentlichen Großbuchstaben noch etwas größer; der Wert `normal` stellt die normale Möglichkeit der Groß-/Kleinschreibung wieder her

- `font-weight`: **Schriftgewicht**, d.h. wie dick (oder fett) oder dünn werden die Schriftzeichen dargestellt; neben dem Wert `normal` gibt es die Werte `bold, bolder, light, lighter`, um die Schrift fetter bzw. dünner auszugeben
  - daneben gibt es die Möglichkeit, numerische Werte zu verwenden zwischen 100 und 900 in 100er-Schritten, mit 400 als Standardwert in der Bedeutung `normal`.

Zusammenfassung : `font`

Stilregeln für Schriften können auch zusammengefasst werden in der Stilregel `font`. Nach optionalen Angaben zu `font-style`, `font-variant` und `font-weight` werden die Schriftgröße (`font-size`) und die Schriftfamilie (`font-family`) festgelegt. Oft ist es einfacher, die Regel `font` anstelle der einzelnen Regeln anzugeben.

Beispiel `font`

```
1    p { font: italic bold 12pt Verdana, sans-serif }
```

1.5.1.1 Schriften im Web

Die Verfügbarkeit von Schriften auf den Client-Systemen, die den Webbrowser ausführen, ist nicht sichergestellt. Schon Sicherheitsaspekte lassen es nicht zu, durch direkten Zugriff auf die Ressourcen des Client-Systems festzustellen, welche Schriften dort installiert sind. Die angegebenen Schriftfamilien stellen also nur einen Vorschlag dar. Kann der Webbrowser die angegebenen Schriften nicht finden, verwendet er die nächste entsprechende Systemschrift.

Schriftfamilien sicher angeben

Sie sollten auf die Verwendung ungewöhnlicher, kaum verbreiteter Schriften verzichten. Um sicherzustellen, dass zumindest die von Ihnen beabsichtigte generische Schriftfamilie (`serif`, `sans-serif`, `cursive`, `fantasy`, `monospace`) verwendet wird, sollten Sie die Möglichkeit nutzen, eine Liste von Schriftfamilien anzugeben und diese mit der Angabe der generischen Schriftfamilie abzuschließen. Der Webbrowser arbeitet die Liste von links nach rechts ab und versucht, die angegebene Schriftart auf dem Client-System zu verwenden.

Hinweis: Webfonts

Mit CSS3 / HTML5 sind die sog. Webfonts eingeführt worden, um auf einfache Weise Schriften, die auf dem Client-System nicht zur Verfügung stehen, einzubinden. Weitere Einzelheiten dazu werden in einem anderen Dokument behandelt.

Schriftgröße relativ angeben

Schriftgrößen können in relativen oder absoluten Einheiten angegeben werden. Als relative Einheiten können verwendet werden:

- `em`: Höhe einer Schrift
- `ex`: Höhe des Buchstabens x einer Schrift
- `px`: Bildpunkt eines Ausgabegeräts
- `%` : prozentuale Angabe.

Bei der Verwendung als Schriftgröße beziehen sich die Angaben zu `em`, `ex` und `%` auf die Schriftgröße des in der Dokumentstruktur hierarchisch übergeordneten HTML-Elements. `1.5em` oder `150%` geben also eine um die Hälfte größere Schrift an als die Schrift des hierarchisch übergeordneten HTML-Elements.

Die Einheit `px` ist eine relative Angabe, weil die tatsächliche Größe von der Auflösung des Ausgabegeräts abhängt und dementsprechend sehr unterschiedlich sein kann. Beachten Sie bei der Größenangabe mit Pixel, dass Sie bei Bildschirmen mit hoher Auflösung eventuell eine sehr kleine Darstellung der Schrift erzielen.

Wenn man eine relative Schriftgröße verwendet, wird häufig die Einheit `em` verwendet.

Schriftgröße absolut angeben

Absolute Einheiten sind:

- `in`: Inches (1 inch = 2,54 Zentimeter)
- `cm`: Zentimeter
- `mm`: Millimeter
- `pt`: Punkt (*Points*), hier als 1/72 Inch definiert
- `pc`: Pica (1 Pica = 12 Punkte).

Häufiger verwendet wird die Größenangabe `pt`, die ursprünglich aus dem Druckgewerbe stammt.

Weitere Angaben zur Schriftgröße

Die Schriftgröße kann auch mit Hilfe von Schlüsselwörtern absolut oder relativ angegeben werden:

- die Schlüsselwörter `xx-small`, `x-small`, `small`, `medium`, `large`, `x-large`, `xx-large` sind absolute Angaben, die sich aus der Zuordnung zu Voreinstellungen des Webbrowsers ergeben
- die Schlüsselwörter `larger` und `smaller` bewirken eine Veränderung der aktuell verwendeten Schriftgröße (Vergrößerung bzw. Verkleinerung); der Grad der Veränderung ergibt sich wiederum aus den Voreinstellungen des Webbrowsers.

Beispiel Schriftgröße

```
<!DOCTYPE html>
<html>
<head>
   <title>Schriftgrößen</title>
   <meta charset="UTF-8" />
   <style type="text/css">
      div { font-size: 14pt; font-family: Verdana, sans-serif; }
      #p1 { font-size: 60%; }
      #p2 { font-size: 1.5em; }
      #p3 { font-size: 1.5ex; }
      #p4 { font-size: 12px; }
   </style>
</head>
<body>
   <div>
      <p>Absatz 0: ohne Änderung</p>
      <p id="p1">Absatz 1: 60%</p>
      <p id="p2">Absatz 2; 1.5em</p>
      <p id="p3">Absatz 3; 1.5ex</p>
      <p id="p4">Absatz 4; 12px</p>
   </div>
</body>
</html>
```

Tipp: CSS-Eigenschaft `font` verwenden

> Eine Hinterlassenschaft aus den älteren Versionen von HTML ist die Verwendung des HTML-Elements `font`. Dieses Element gehört wie auch `b` zu den HTML-Elementen, die nicht die Dokumentstruktur, sondern die Präsentation bestimmen. Mit der klaren Trennung von Struktur und Präsentation wurde das HTML-Element `font` überflüssig und ist daher in den neuen HTML-Standards **nicht mehr enthalten**. Verwenden Sie zur Festlegung der Schriften und ihrer Eigenschaften konsequent die Möglichkeiten von CSS!

1.5.2 Farbe und Hintergrund

Mit Hilfe von CSS-Stilregeln können die Vordergrundfarbe - i.d.R. die Farbe des Textes eines HTML-Elements - und der Hintergrund festgelegt werden. Beim Hintergrund eines HTML-Elements besteht die Möglichkeit, Bilder zu verwenden.

Vordergrundfarbe

Die Vordergrundfarbe wird mit dem CSS-Stil `color` spezifiziert. Damit werden die in den HTML-Elementen vorhandenen Texte in der entsprechenden Farbe dargestellt.

Hintergrundfarbe

Die Hintergrundfarbe eines HTML-Elements wird für den gesamten Platz des HTML-Elements verwendet mit Ausnahme der Abstandsbereiche zu anderen HTML-Elementen (`margin`). Neben direkten Farbwerten ist es möglich, den Wert `transparent` zu verwenden, womit ein Durchscheinen ansonsten verdeckter HTML-Elemente ermöglicht wird. Der Wert `transparent` wird außerdem als Voreinstellung verwendet, wodurch bei der Schachtelung von HTML-Elementen der Eindruck entsteht, dass die Farbangabe für den Hintergrund vererbt wird. Tatsächlich scheint aber nur die Farbe des übergeordneten Elements durch, die Farbangabe wird also nicht an hierarchisch untergeordnete HTML-Elemente weitergegeben.

Beispiel

```
<!DOCTYPE html>
<html>
<head>
   <title>Vordergrund- und Hintergrundfarbe</title>
   <meta charset="UTF-8" />
   <style type="text/css">
      /* Stilregeln */
      .gelb_v  { color : yellow; }
      .rot_v   { color : red; }
      .blau_h  { background-color : blue; }
      .gruen_h { background-color : green; }
   </style>
</head>
<body>
   <div class="blau_h">
      <p class="gelb_v">erster Absatz</p>
      <p class="rot_v">zweiter Absatz</p>
      <p class="gruen_h">dritter Absatz</p>
   </div>
</body>
</html>
```

Farbwerte

Farbwerte werden entweder mit einem Schlüsselwort oder mit einem numerischen Wert angegeben. Der W3C-CSS-Standard führt 17 Standardfarbwerte auf, zu denen z.B. die im vorigen Beispiel verwendeten Werte `yellow`, `red`, `blue` und `green` zählen. Neben diesen Farbwerten werden im Standard noch sog. *Systemcolors* definiert, mit denen ein Bezug zu den Farbeinstellungen des Client-Systems, auf dem der Webbrowser ausgeführt wird, möglich ist. Ein typisches Beispiel ist der Farbwert, der durch das Schlüsselwort `ButtonFace`, benutzt werden kann : es wird die auf dem Client-System für Schalter eingesetzte Farbe benutzt.

Die numerische Spezifikation eines Farbwerts beruht auf dem RGB-Farbmodell : man gibt Anteile für die Grundfarben Rot, Grün und Blau an. Jeder Anteil kann einen Wert zwischen 0 (Farbe ist nicht vorhanden) und 255 (Farbe ist n höchster Sättigung vorhanden) annehmen. Es gibt wiederum verschiedene Arten, den numerischen Wert anzugeben:

- als eine Kombination von Hexadezimalwerten mit vorangestelltem Zeichen #, je zwei Stellen für die 3 Grundfarben Rot, Grün und Blau; Beispiel:
  - `#000000` entspricht Weiß (`white`)
  - `#00FF00` entspricht Grün (`green`)
  - `#FFFFFF` entspricht Schwarz (`black`)
- in einer funktionalen Schreibweise mit der Funktion `rgb`, die die Werte von 0 bis 255 als Parameter erhält; möglich sind auch prozentuale Werte; Beispiele:
  - `rgb(0,0,0)` entspricht Weiß (`white`)

- `rgb(0,255,0)` entspricht Grün (`green`)
- `rgb(255,255,255)` entspricht Schwarz (`black`)
- `rgb(100%, 0%, 0%)` entspricht Rot (`red`).

Beispiel

```
<!DOCTYPE html>
<html>
<head>
   <title>Farbwerte</title>
   <meta charset="UTF-8" />
   <style type="text/css">
      /* Stilregeln */
      .rot_v      { color : rgb(255,0,0); }
      .blau_v     { color : #0000FF; }
      .syscolor_h { background-color : ButtonFace; }
   </style>
</head>
<body>
   <p class="rot_v">erster Absatz</p>
   <p class="blau_v">zweiter Absatz</p>
   <p class="syscolor_h">dritter Absatz</p>
</body>
</html>
```

Bild als Hintergrund

Als Hintergrund eines HTML-Elements kann auch ein Bild verwendet werden. Damit verfolgt man häufig das Ziel, ansprechende Darstellungseffekte zu erreichen, ebenso wichtig ist aber in vielen Fällen, Sinnbilder, die bei der Bedienung von Webseiten von Bedeutung sind, einzubeziehen. Das ist zwar auch mit dem HTML-Element `img` möglich, aber vor allem bei einer häufigeren Verwendung umständlicher, wie das nachfolgende Beispiel zeigen wird. Außerdem können damit Webseiten mit 'Wasserzeichen' versehen werden.

Als Wert für den Stil `background-image` wird der Bezug auf eine Bildressource mit der funktionalen Schreibweise `url(<uri>)` verwendet, wobei `<uri>` durch entsprechenden Zugriffspfad zu ersetzen ist. Zusätzlich zum Stil `background-image` können folgende Stile angegeben werden:

- `background-repeat`: regelt, ob das Hintergrundbild wiederholt dargestellt werden soll
- `background-position`: gibt an, wie das Hintergrundbild im zur Verfügung stehenden Darstellungsbereich eines HTML-Elements positioniert werden soll, sowohl in horizontaler als in vertikaler Richtung
- `background-attachment`: ermöglicht es, ein Hintergrundbild bei einem nicht vollständig sichtbaren HTML-Element im sichtbaren Bereich zu halten.

Kombinierte Angaben

Wie bei den Stilregeln zu Schriften können die Angaben zu Element-Hintergründen mit einer Stilregel `background` zusammenfassend angegeben werden. Die Reihenfolge der einzelnen Angaben ist dabei ohne Bedeutung.

Beispiel

```
<!DOCTYPE html>
<html>
<head>
   <title>Hintergrundbilder</title>
```

```
5       <meta charset="UTF-8" />
6       <style type="text/css">
7          /* Stilregeln */
8          .button_first {
9             background-image: url(page-first.gif);
10            background-repeat: no-repeat;
11            background-position:left;
12            background-color: ButtonFace;
13            padding-left:20px;
14         }
15         /* zusammengefasste Form */
16         .button_last {
17            background: ButtonFace right url(page-last.gif) no-repeat;
18            padding-right:20px;
19         }
20         .Wiederholung {
21            background: url(ok.png) repeat;
22            width:100px;
23            height:100px;
24            margin-top:20px;
25         }
26      </style>
27   </head>
28   <body>
29      <div>
30         <button class="button_first">Anfang</button>
31         <button class="button_last">Ende</button>
32      </div>
33      <div class="Wiederholung">
34      </div>
35   </body>
36   </html>
```

1.5.3 Textformatierung

Zur Formatierung von Texten stehen Ihnen bei CSS eine Reihe von Eigenschaften zur Verfügung:

- Texteinzug: `text-indent`
- Textausrichtung: `text-align`
- Textdekoration: `text-decoration`
- Wort- und Buchstabenabstände: `word-spacing` und `letter-spacing`
- Automatische Umwandlung der Schreibweise: `text-transform`
- Behandlung von nicht sichtbarem Leerraum: `white-space`.

1.5.3.1 Texteinzug

Mit der CSS-Eigenschaft `text-indent` können Sie festlegen, in welchem Umfang in der ersten Zeile eines Textblocks der Text durch Leerraum eingerückt wird. Es wird folgende Syntax verwendet:

```
text-indent ::= "text-ident" ":" [ numerische Angabe absolut | numerische Angabe relativ ] ";"
```

1.5.3.2 Textausrichtung

Die Ausrichtung eines Textes innnerhalb eines Textblocks wird durch die CSS-Eigenschaft `text-align` festgelegt. Es wird folgende Syntax verwendet:

```
text-align ::= "text-align" ":" ["left" | "right" | "center" | "justify"] ";"
```

Die Alternativen haben folgende Bedeutung:

- `left`: der Text wird linksbündig ausgerichtet
- `right`: der Text wird rechtsbündig ausgerichtet
- `center`: der Text wird innerhalb des zur Verfügung stehenden Platzes zentriert ausgerichtet
- `justify`: der Text wird im Blocksatz ausgerichtet

1.5.3.3 Textdekoration

Mit der CSS-Eigenschaft `text-decoration` knn man erreichen, dass Text unterstrichen, durchgestrichen, überstrichen oder blinkend ausgegeben wird. Es wird folgende Syntax verwendet:

```
text-decoration ::= "text-decoration" ":" ["underline" | "line-through" | "overline" | "blink" | "none"] ";"
```

Der Eigenschaftswert `none` ist vor allem dann nutzbar, wenn eine Textauszeichnung, die automatisch für ein HTML-Element eingestellt wird, vollständig entfernt werden soll. Typisches Anwendungsbeispiel ist die Textauszeichnung, die bei Hyperlinks verwendet wird. Normalerweise werden diese durch eine Unterstreichung gekennzeichnet. Will man diese Textauszeichnung beseitigen, definiert man folgende CSS-Stilregel:

- `a:link, a:visited, a:active, a:hover { text-decoration: none; }`

Bei Links wird zwischen den Varianten Link (`link`), dem besuchten Link (`visited`) und dem aktiven Link (`active`) unterschieden. Für Links, über denen die Maus steht, wird das Verhalten durch `hover` festgelegt. Für jede Variante kann eine Stilregel angegeben werden.

1.5.3.4 Wort- und Buchstabenabstände

Wort- und Buchstabenabstände können mit den CSS-Eigenschaften `word-spacing` und `letter-spacing` eingestellt werden. Es wird folgende Syntax verwendet:

```
word-spacing ::= "word-spacing" ":" [ "normal" | numerische Angabe absolut | numerische Angabe relativ ] ";"
letter-spacing ::= "letter-spacing" ":" [ "normal" | numerische Angabe absolut | numerische Angabe relativ ] ";"
```

wobei jeweils der Wert “normal” die Voreinstellung ist.

Bei den numerischen Werten wird jeweils der **Zuwachs** an Leerraum zwischen den Wörtern bzw. Buchstaben angegeben. Mit dem Eigenschaftswert `normal` wird der normale Abstand eingestellt.

1.5.3.5 Automatische Umwandlung der Schreibweise

Die CSS-Eigenschaft `text-transform` erlaubt es, die Schreibweise von Texten automatisch anzupassen. Es wird folgende Syntax verwendet:

```
text-transform ::= "text-transform" ":" [ "capitalize" | "uppercase" | "lowercase" | "none" ] ";"
```

Die Alternativen bedeuten:

- `capitalize`: Wortanfänge als Großbuchstaben
- `uppercase`: nur Großbuchstaben
- `lowercase`: nur Kleinbuchstaben
- `none`: keine Textänderung durchführen.

1.5.3.6 Behandlung von nicht sichtbarem Leerraum

Bei der Darstellung von Fließtext werden normalerweise aufeinanderfolgende Leerzeichen und andere nicht sichtbare, zum Leeraum zählende Zeichen wie Tabulatoren zusammengefasst, Zeilenubrüche werden dort nach Bedarf platziert. Dieses Verhalten kann durch die CSS-Eigenschaft `white-space" beeinflusst werden. Es wird folgende Syntax verwendet:

```
white-space ::= "white-space" ":" [ "normal" | "pre" | "nowrap" | "pre-wrap" | "pre-line" ] ";"
```

Die Alternativen bedeuten:

- `normal`: Einstellen des Standardverhaltens
- `pre`: der Leerraum wird nicht zusammengefasst, sondern vorhandene Leerzeichen bleiben erhalt; es werden nur die vorhandenen Zeilenwechsel verwendet
- `nowrap`: der Leerraum wird zusammengefasst, im Text vorhandene Zeilenumbrüche werden nicht berüksichtigt
- `pre-wrap`: der Leerraum wird nicht zusammengefasst, sondern vorhandene Leerzeichen bleiben erhalt; Zeilenwechsel werden vorgenommen, wenn im Quelltext ein Zeilenwechsel vorliegt oder der zur Verfügung stehende Platz dies erfordert
- `pre-line`: der Leerraum wird zusammengefasst, im Text vorhandene Zeilenumbrüche werden berüksichtigt.

1.6 Das Box-Modell

Darstellbare HTML-Elemente werden in rechteckigen Bereichen, den Elementboxen, angezeigt. In Elementboxen können um den Inhalt herum Innenabstände, Rahmen und Außenabstände vorgesehen werden. HTML-Elemente wie `div` und `p` sind Block-Level-Elemente, deren Elementboxen im Elementfluss untereinander dargestellt werden. Inline-Level-Elemente wie das HTML-Element `span` bleiben dagegen Bestandteil von Textzeilen. Das ##Visual Formatting Model## in CSS2 regelt die Bestimmung von Weite und Höhe der Elementboxen und deren Anordnung. Darstellungsprobleme gibt es mit dem Internet-Explorer bis zur Version 6 durch die falsche Berechnung von Weite und Höhe.

CSS unterscheidet verschiedene Elementtypen, zu denen Blockelemente gehören. Das Boxmodell für Blockelemente definiert CSS-Eigenschaften für den Rahmen (`border`), die Breite (`width`), die Höhe (`height`), den Innenabstand (`padding`) und den Außenabstand (`margin`). Außenabstände von vertikal aneinander angrenzenden Blockelementen werden zusammengefasst. Bei unterschiedlichen Abständen wird der größere gewählt.

1.6.1 Elementboxen

Für jedes darstellbare HTML-Element wird ein rechteckiger Bereich vorgesehen, der **Elementbox** genannt wird. Jede Elementbox besteht aus:

- dem Inhaltsbereich (*content*), in dem sich der darstellbare Inhalt eines HTML-Elements befindet
- einem Innenabstand (*padding*)
- einem Rahmen (*border*)
- einem Außenabstand (*margin*).

Die einzelnen Bereiche umschließen sich wie Zwiebelschalen. Jeder Elementbox können mit Hilfe von CSS-Stilanweisungen außerdem eine Hintergrundfarbe und ein Hintergrundbild zugeordnet werden.

Im folgenden Beispiel wird mit Hilfe von Rahmen verdeutlicht, wo Elementboxen vorliegen.

Beispiel

```
<!DOCTYPE html>
<html>
<head>
   <title>Elementboxen / 1</title>
   <meta charset="UTF-8" />
   <style type="text/css">
      p {
         border: 1px dashed red;
      }
      strong {
         border: 2px solid gray;
      }
      em {
         border: 1px solid green;
      }
   </style>
</head>
<body>
   <p>
      Absatz mit hervorgehobenen
      <strong>Elementen</strong>, wie
      z.B. <em>diesem</em> hier.
   </p>
</body>
</html>
```

Mit CSS-Stileigenschaften sind Sie in der Lage, Innenabstand, Rahmen und Außenabstand gezielt festzulegen. Wenn Sie zu einem HTML-Element keine Angaben machen, werden die Voreinstellungen des Webbrowser verwendet.

Bezeichnung der Seiten

Die vier Seiten einer Elementbox werden mit *top* (oben), *right* (rechts), *bottom* (unten), *left* (links) bezeichnet und in dieser Reihenfolge - im Uhrzeigersinn - in den CSS-Stileigenschaften festgelegt.

1.6.2 Größe einer Box festlegen

Die Größe einer Box wird durch die Attribute `width` und `height` bestimmt, wenn es sich um ein Block-Level-Element handelt. Die CSS-Eigenschaft `box-sizing` bestimmt, wie die Berechnung der Gesamtgröße einer Box erfolgt:

- `box-sizing: content-box`:
  - dies ist der voreingestellte Standardwert
  - Weite und Höhe beziehen sich nur auf den eigentlichen Inhalt; die Gesamtgröße ergibt sich aus Weite und Höhe und den Angaben zu Innenabstand (`padding`), Rahmen (`border`) und Außenabstand (`margin`)
- `box-sizing: border-box`:
  - Weite und Höhe beziehen sich nur auf den eigentlichen Inhalt *und* den Angaben zu Innenabstand (`padding`) und Rahmen (`border`); die Gesamtgröße ergibt sich dann aus der angegebenen Weite und Höhe dem Außenabstand (`margin`).

Beispiel:

- es sei eine Weite von 100px und eine Höhe von 50px angegeben
- als Innenabstand wird ein Wert von 10px verwendet (alle Seiten!)
- als Rahmendicke wird eine Wert von 2px verwendet (alle Seiten!)

- als Außenabstand wird ein Wert von 20px verwendet (alle Seiten!)

Gesamtgröße bei `box-sizing: content-box`:

- Weite: `width` + 2*`padding` + 2*`border-width` + 2*`margin` ⇒ 100px + 20px + 4px + 40px = 164px
- Höhe: `height` + 2*`padding` + 2*`border-width` + 2*`margin` ⇒ 50px + 20px + 4px + 40px = 114px

Gesamtgröße bei `box-sizing: border-box`:

- Weite: `width` + 2*`margin` ⇒ 100px + 40px = 140px
- Höhe: `height` + 2*`margin` ⇒ 50px + 40px = 90px

1.6.3 Innenabstand festlegen

Mit dem Innenabstand können Sie einen Leerraum zwischen einem Rahmen und dem darstellbaren Inhalt eines HTML-Elements ermöglichen. Dieser Leerraum erhält die Hintergrundfarbe, die Sie dem Inhaltsbereich zuordnen. Die CSS-Stileigenschaft `padding` wird verwendet, um die Größe des Innenabstands zu bestimmen. Dabei haben Sie verschiedene Möglichkeiten, die Größe anzugeben:

- Sie können eine einheitliche Größe für alle Seiten angeben oder
- Sie können für jede Seite oder für gegenüberliegende Seiten die Größe angeben
  - wenn Sie nur zwei Werte angeben, ist die Reihenfolge der Seiten wichtig: der erste Wert legt den Innenabstand für `top` und `bottom` fest, der zweite Wert dann den Innenbstand für `right` und `left`
- Sie können die Größe in absoluten oder relativen Einheiten angeben.

Mit der CSS-Stileigenschaft `padding` können Sie eine zusammnengefasste Festlegung des Innenabstands vornehmen. Die CSS-Stileigenschaften `padding-top`,`padding-right`, `padding-bottom` und `padding-left` ermöglichen Ihnen die Festlegung individueller Eigenschaften für die einzelnen Seiten. Wenn Sie `padding` und eine der seitenspezifischen Eigenschaften gleichzeitig verwenden, kommt es auf die Reihenfolge in der CSS-Spezifikation an, welcher Eintrag der wirksame ist.

Das folgende Beispiel zeigt die Wirkung verschiedener Einstellungen für den Innenabstand sowie die Zuweisung einer Hintergrundfarbe.

Beispiel Innenabstand

```
<!DOCTYPE html>
<html>
<head>
   <title>Elementboxen / 2</title>
   <meta charset="UTF-8" />
   <style type="text/css">
      p {
         border: 1px solid red;
         padding: 0px;
         width: 400px;
      }
      #Mit {
         padding: 10px 5px;
      }
      #Mit2 {
         padding: 0.75em 15px 0.75em 15px;
         background-color: yellow;
         color: blue;
      }
   </style>
```

```
21    </head>
22    <body>
23       <p>
24          Ein Absatz ohne Innenraum,
25          wodurch der Text an den
26          Rand stößt. Das sieht in den
27          meisten Fällen nicht gut aus.
28       </p>
29       <p id="Mit">
30          Ein Absatz mit Innenraum,
31          wodurch der Text deutlich
32          vom Rand abgesetzt ist.
33       </p>
34       <p id="Mit2">
35          Ein zweiter Absatz mit Innenraum,
36          wodurch der Text deutlich
37          vom Rand abgesetzt ist.
38       </p>
39    </body>
40    </html>
```

1.6.4 Rahmen festlegen

Mit der CSS-Stileigenschaft `border` und definieren Sie einen Rahmen um ein HTML-Element. Ihnen stehen mehrere Einstellungen zur Verfügung, die für alle vier Seiten gelten:

- die Angabe der Dicke des Rahmens
  - sie wird entweder in symbolischer Form mit einem der Werte `thin`, `medium` oder `thick` angegeben oder
  - als exakter nicht negativer Wert z.B. in der Maßeinheit `px` oder `em`
- der Stil des Rahmens, bei dem Sie einen der folgenden Werte angeben
  - `none`, `hidden`, `dotted`, `dashed`, `solid`, `double`, `groove`, `ridge`, `inset`, `outset`
- die Farbe des Rahmens, die Sie wie andere Farben auch spezifizieren.

Die Stilangabe `none` wird verwendet, wenn Sie einem HTML-Element keinen Rahmen zuweisen wollen.

Syntax Rahmen

Die CSS-Stileigenschaft `border` wird folgendermaßen notiert:

```
border ::= "border" ":" [border-width] [border-style] [border-width] ";"
```

Die Reihenfolge der einzelnen Eigenschaften ist beliebig.

Wie bei der CSS-Stileigenschaft `padding` können Sie diese zusammengefasste Schreibweise auch für jede Seite der Elementbox getrennt angeben. Die vier möglichen CSS-Stileigenschaften sind dann `border-top`, `border-right`, `border-bottom` und `border-left`, die wie die CSS-Eigenschaft `border` definiert sind.

Darüberhinaus besteht die Möglichkeit, die einzelnen Eigenschaften `border-width`, `border-style` und `border-width` einzeln zu definieren, wobei wie bei `padding` ein Wert für alle Seiten, zwei Werte für einander gegenüberliegende Seiten oder ein Wert für jede Seite angegeben werden kann. Eine weitere Verfeinerung der Stilangaben ist durch die Verwendung seitenspezifischer Angaben möglich:

- Rahmendicke für jede Seite einzeln: `border-top-width`, `border-right-width`, `border-bottom-width`, `border-left-width`

- Rahmenstil für jede Seite einzeln: `border-top-style`, `border-right-style`, `border-bottom-style`, `border-left-style`
- Rahmenfarbe für jede Seite einzeln: `border-top-color`, `border-right-color`, `border-bottom-color`, `border-left-color`.

Beispiel Rahmen

```
<!DOCTYPE html>
<html>
<head>
   <title>Elementboxen / 3</title>
   <meta charset="UTF-8" />
   <style type="text/css">
      p {
         border: 5px solid red;
         padding: 5px;
         width: 400px;
      }
      #dotted {
         border: 2px dotted gray;
      }
      #dashed {
         border: 2px dashed gray;
      }
      #double {
         border: 5px double gray;
      }
      #groove {
         border: 1em groove gray;
      }
      #ridge {
         border: 10px ridge gray;
      }
      #inset {
         border: 10px inset gray;
      }
      #outset {
         border: 10px outset gray;
      }
   </style>
</head>
<body>
   <p>
      Rahmenstil solid
   </p>
   <p id="dotted">
      Rahmenstil dotted
   </p>
   <p id="dashed">
      Rahmenstil dashed
   </p>
   <p id="double">
      Rahmenstil double
   </p>
   <p id="groove">
      Rahmenstil groove
```

```
50        </p>
51        <p id="ridge">
52           Rahmenstil ridge
53        </p>
54        <p id="inset">
55           Rahmenstil inset
56        </p>
57        <p id="outset">
58           Rahmenstil outset
59        </p>
60     </body>
61     </html>
```

1.6.5 Außenabstand festlegen

Den Außenabstand einer Elementbox geben Sie mit der CSS-Stileigenschaft `margin` an. Die Spezifikation dieser Eigenschaft entspricht derjenigen der CSS-Stileigenschaft `padding`, Sie können also entweder die zusammengefasste Form verwenden oder für jede Seite eine eigene CSS-Stileigenschaft wie z.B. `margin-top` verwenden.

Der Außenabstand wird **nicht** in der Hintergrundfarbe der Elementbox dargestellt, sondern in der Hintergrundfarbe der umgebenden Elementbox. Die umgebende Elementbox kann sich auch aus dem HTML-Element `body` ergeben.

Im folgenden Beispiel wird verdeutlicht, wie man mit Außenabständen zusätzlichen Leerraum zwischen einzelnen HTML-Elementen bzw. deren Elementboxen erzeugt.

Beispiel

```
1     <!DOCTYPE html>
2     <html>
3     <head>
4        <title>Elementboxen / 4</title>
5        <meta charset="UTF-8" />
6        <style type="text/css">
7           p {
8              border: 1px solid;
9              padding: 10px;
10             margin: 0px;
11             width: 400px;
12          }
13          #Mit {
14             margin: 10px;
15          }
16          #Mit2 {
17             margin: 0px 10px 0px 10px;
18             background-color: yellow;
19             color: blue;
20          }
21          #Mit3 {
22             margin: 0px 10px 10px 10px;
23          }
24          #Mit4 {
25             margin: 20px 0px 0px 10px;
26          }
27       </style>
28    </head>
```

```
29    <body>
30       <p>
31          Absatz 1: mit Innenraum,
32          aber keinem Außenabstand.
33       </p>
34       <p id="Mit">
35          Absatz 2: mit Innenraum
36          und einheitlichem Außenabstand.
37       </p>
38       <p id="Mit2">
39          Absatz 3: mit Innenraum
40          und Außenabstand an den Seiten, aber
41          ohne Außenabstand oben und unten.
42       </p>
43       <p id="Mit3">
44          Absatz 4: mit Innenraum
45          und Außenabstand an den Seiten, aber
46          ohne Außenabstand oben und unten.
47       </p>
48       <p id="Mit4">
49          Absatz 5: mit Innenraum
50          und größerem Außenabstand oben.
51       </p>
52    </body>
53    </html>
```

Sie können im Beispiel gut erkennen, dass sich die Rahmen von Elementboxen berühren, wenn die Außenabstände der Elementboxen den Wert 0 aufweisen. Die Rahmen fallen nicht zusammen, beide Rahmen werden ausgegeben. Dadurch erscheint der Rahmen zwischen beiden Elementboxen in diesem Beispiel in doppelter Breite.

Vertikale Außenabstände fallen zusammen

Im Beispiel läßt sich ein weiterer Effekt beobachten. Zwischen Absatz 1 und Absatz 2 ist ein Platz von 10px Größe aufgrund der CSS-Stileigenschaften für Absatz 2 (CSS-Regel `Mit`) vorhanden. Zwischen Absatz 2 und Absatz 3 müsste ein Leerraum von 20px vorhanden sein, da die beiden CSS-Stilregeln `Mit` und `Mit2` jeweils einen Außenabstand unten bzw. oben von je 10px angeben. Man sieht aber, dass der Leerraum nur so groß ist wie zwischen Absatz 1 und Absatz 2. Zwischen Absatz 4 und 5 ist der Leerraum dagegen größer.

Grund dieses Effekts ist die Regel, dass vertikale Außenabstände übereinander angeordneter Elemente zusammenfallen, d.h. es wird der jeweils größere Wert verwendet, die Außenabstände werden **nicht** addiert:

- bei Absatz 1 ist der untere Außenabstand 0px, bei Absatz 2 der obere Außenabstand 10px, also wird der Wert 10px verwendet
- bei Absatz 2 ist der untere Außenabstand 10px, bei Absatz 3 der obere Außenabstand ebenfalls 10px, also wird der Wert 10px verwendet
- bei Absatz 4 ist der untere Außenabstand 10px, bei Absatz 5 der obere Außenabstand 20px, also wird der Wert 20px verwendet.

Warum wird dieser Effekt verwendet? Mit einem Außenabstand kann man bei einem HTML-Element einen sichtbaren Abstand zu anderen HTML-Elementen sicherstellen ohne durch eine Aufsummierung von Außenabständen optisch unschöne Wirkungen zu erzielen.

1.6.6 Elementfluss und CSS-Elementtypen

Wenn ein HTML-Dokument auf einem Ausgabemedium dargestellt werden soll, müssen Größe und Position der Elementboxen für jedes darstellbare HTML-Element bestimmt werden. Für die automatische Anordnung der Elementboxen ist von Bedeutung, ob die HTML-Elemente ganze Blöcke oder nur einzelne Textabschnitte repräsentieren. CSS unterscheidet dementsprechend zwischen Block-Level-Elementen und Inline-Level-Elementen.

Block-Level-Elemente

Block-Level-Elemente sind die HTML-Elemente, die die logische Dokumentenstruktur festlegen, z.B.:

- Überschriften: die HTML-Elemente `h1` bis `h6`
- Absätze und Abschnitte: die HTML-Elemente `p` und `div`
- geordnete und ungeordnete Listen sowie Definitionslisten: die HTML-Elemente `ol`, `ul` und `dl`
- Formulare und Tabellen: die HTML-Elemente `form` und `table`.

Inline-Level-Elemente

Inline-Level-Elemente sind die HTML-Elemente, die zur Kennzeichnung einzelner Textabschnitte verwendet werden oder eine spezielle Bedeutung haben, wie z.B. die Eingabefelder bei Formularen. Typische Beispiele für Inline-Level-Elemente sind:

- Textabschnitte: das HTML-Element `span`
- Hervorhebungen: die HTML-Elemente `strong` und `em`.

Visual-Formatting-Model

Die automatische Anordnung der Block- und Inline-Level-Elemente bezeichnet man als **Elementfluss**. Bei der Ausgabe auf einem Bildschirm bestimmt das **Visual Formatting Model**, das im CSS2-Standard des W3C beschrieben ist, die Regelungen für den Elementfluss:

- Block-Level-Elemente werden nacheinander in **vertikaler** Anordnung ausgegeben, d.h. nach jedem Block erfolgt quasi ein Zeilenumbruch
- Inline-Level-Elemente befinden sich innerhalb von Block-Level-Elementen und werden nacheinander **horizontal** - zeilenweise - angeordnet, ggf. erfolgt ein Umbruch auf eine neue Zeile, wenn der Platz in der Zeile nicht mehr ausreicht.

In den vorangegangenen Beispielen konnten Sie bereits den Elementfluss von Block-Level-Elementen beobachten: die einzelnen Absätze (HTML-Element `p`) wurden untereinander angeordnet.

Die Unterscheidung der Elementart hat auch Auswirkungen auf die Bestimmung der Größe der Elementboxen und die Möglichkeiten, auf die Positionierung Einfluss zu nehmen.

Die Elementart wird durch die CSS-Stileigenschaft `display` repräsentiert: bei Block-Level-Elementen finden Sie dort den Wert `block`, bei Inline-Level-Elementen den Wert `inline`. Es ist möglich, diesen Wert zu ändern, also aus einem Block-Level-Element ein Inline-Level-Element zu machen und umgekehrt. Davon sollten Sie normalerweise absehen.

Es gibt weitere Elementarten. Weitere Einzelheiten dazu werden in einem anderen Dokument behandelt.

1.6.7 Größe der Elementboxen

Die Größe der Elementboxen ist sowohl von den Eigenschaften abhängig, die Sie vorsehen, als auch vom Typ des HTML-Elements, das die Elementbox repräsentiert. Darüberhinaus wird bei CSS auch noch zwischen nicht-ersetzten und ersetzten Elementen unterschieden:

- nicht ersetzte Elemente sind alle darstellbaren HTML-Elemente, deren Inhalte sich direkt aus dem HTML-Dokument ergeben
- ersetzte Element sind alle darstellbaren HTML-Elemente, deren Inhalte aus externen Quellen stammen, z.B. Bilder, die mit Hilfe des HTML-Elements `img` eingefügt werden.

Ersetzte Elemente wie z.B. Bilder werden deshalb von CSS unterschiedlich behandelt, weil deren Größe und Seitenverhältnisse außerhalb festgelegt werden. Es handelt sich damit um Eigenschaften, die nicht bei der Darstellung verändert können. Z.B. kann die Darstellung von Bilder zwar neu skaliert werden, die dem Bild innewohnenden Eigenschaften werden aber nicht verändert.

Berechnung der Weite

Die Weite einer Elementbox ergibt sich aus der Weite des Inhaltsbereichs, die mit der CSS-Stileigenschaft `width` definiert wird, der Größe des Innen- und Außenabstands und der Rahmendicke. Die CSS-Stileigenschaft `width` ist aber nur bei Block-Level-Elementen anwendbar. Die Weite des Inhaltsbereichs von Inline-Level-Elementen wird automatisch anhand des Inhalts bestimmt.

Bei nicht ersetzten Block-Level-Elementen ergibt sich die Gesamtbreite der Elementbox aus der Summe des rechten und linken Außenabstands, der Dicke des rechten und linken Rahmens, des rechten und linken Innenabstands und der Weite des Inhaltsbereichs sowie der Breite eines Laufbalkens (*scrollbar*).

Für die CSS-Stilangaben `width`, `padding` und `margin` kann auch der Wert `auto` verwendet werden, der je nach Art des Elements unterschiedlich ausgewertet wird. Dieser Wert ist auch der voreingestellte Wert, wenn es sonst keine Angabe, auch keine webbrowserspezifische Angabe, gibt. Im Prinzip bedeutet die Angabe `auto` soviel Platz wie möglich für den Inhaltsbereich zu verwenden auf Kosten der Innen- und Außenabstände. Befindet sich die Elementbox in einer anderen Elementbox - die HTML-Elemente sind also geschachtelt - , dann beansprucht die innere Elementbox den gesamten Innenbereich der äußeren Elementbox. Fügen Sie der inneren Elementbox Abstände und Rahmen hinzu, wird der Platz für den Inhaltsbereich der inneren Elementbox kleiner.

Beispiel Weite

```
<!DOCTYPE html>
<html>
<head>
   <title>Elementboxen / 5</title>
   <meta charset="UTF-8" />
   <style type="text/css">
      .outer {
         border: 1px solid;
         width:300px;
         margin:5px;
         padding:0px;
      }
      .inner1 {
         width: auto;
         margin:0px;
         padding:0px;
      }
      .inner2 {
         border: 5px solid gray;
         width: auto;
         margin:0px;
         padding:0px;
      }
      .inner3 {
         border: 1px dashed;
         width: auto;
         margin:10px;
```

```
28              padding:10px;
29          }
30       </style>
31    </head>
32    <body>
33       <div class="outer">
34          <div class="inner1">
35             Abschnitt 1: etwas Text, um die
36             Größenverhältnisse zu verdeutlichen
37          </div>
38       </div>
39       <div class="outer">
40          <div class="inner2">
41             Abschnitt 2: etwas Text, um die
42             Größenverhältnisse zu verdeutlichen
43          </div>
44       </div>
45       <div class="outer">
46          <div class="inner3">
47             Abschnitt 3: etwas Text, um die
48             Größenverhältnisse zu verdeutlichen
49          </div>
50       </div>
51    </body>
52    </html>
```

Sie sehen im Beispiel, dass die inneren Abschnitte bei zusätzlichem Rahmen oder Abständen nicht zu einer Verbreiterung der äußeren Abschnitte führen, sondern die Inhaltsbereiche der inneren Abschnitte verkleinert werden.

Berechnung der Höhe

Die Höhe einer Elementbox ergibt sich aus der Höhe des Inhaltsbereichs, die mit der CSS-Stileigenschaft `height` definiert wird, der Größe des Innen- und Außenabstands und der Rahmendicke. Die CSS-Stileigenschaft `height` ist aber nur bei Block-Level-Elementen anwendbar. Die Höhe des Inhaltsbereichs von Inline-Level-Elementen wird automatisch anhand des Inhalts bestimmt.

Bei ersetzten Elementen, z.B. Bildern, bestimmen die Bildweite und Bildhöhe bzw. das Seitenverhältnis die Höhe entsprechend den Vorgaben, die sich aus der Berechnung der Weite ergeben.

Höhe und CSS-Stileigenschaft `overflow`

Bei nicht ersetzten Block-Level-Elementen, z.B. Abschnitten und Absätzen, fließt in die Bestimmung der Höhe die CSS-Stileigenschaft `overflow` mit ein, die verschiedene Werte entsprechend der folgenden Syntax annehmen kann:

```
overflow ::= "overflow" ":" [ "visible" | "hidden" | "scroll" | "auto" ] ";"
```

Dabei bedeuten

- `visible`: der Inhalt wird vollständig wiedergegeben, ggf. fließt er über vorgegebene Grenzen hinweg (**Voreinstellung**)
- `hidden`: der Inhalt wird entsprechend vorgegebenen Grenzen abgeschnitten, ein Verschiebung des sichtbaren Inhalts mit Laufbalken wird nicht angeboten
- `scroll`: der Inhalt wird entsprechend vorgegebenen Grenzen dargestellt, ein Verschiebung des sichtbaren Inhalts mit Laufbalken wird angeboten; die Laufbalken werden immer dargestellt, auch wenn der Inhalt vollständig dargestellt werden kann

- `auto`: es soll automatisch in den Modus `scroll` gewechselt werden, wenn der Inhalt nicht vollständig dargestellt werden kann.

Wenn also kein Wert für die CSS-Stileigenschaft `overflow` angegeben oder der Wert `visible` verwendet wird oder eine Höhenangabe `auto` vorliegt, wird die Höhe automatisch der Höhe der Elementbox entsprechend dem Inhalt des HTML-Elements angepasst. In allen anderen Fällen wird die vorgegebene Höhe beachtet, je nach Einstellung von `overflow` der Inhalt aber abgeschnitten oder dargestellt, ggf. verschiebbar.

Beispiel Höhe

```
<!DOCTYPE html>
<html>
<head>
   <title>Elementboxen / 6</title>
   <meta charset="UTF-8" />
   <style type="text/css">
      .outer {
         border: 1px solid;
         width:300px;
         height:50px;
         background-color: yellow;
         margin:5px;
         padding:0px;
      }
      .outer2 {
         border: 1px solid;
         width:300px;
         height:50px;
         margin-left:5px;
         margin-top:25px;
         padding:0px;
         overflow:hidden;
      }
      .outer3 {
         border: 1px solid;
         width:300px;
         height:auto;
         margin-left:5px;
         margin-top:25px;
         padding:0px;
         overflow:hidden;
      }
      .inner3 {
         border: 1px dashed;
         width: auto;
         margin:10px;
         padding:10px;
      }
   </style>
</head>
<body>
   <div class="outer">
      <div class="inner3">
         Abschnitt : etwas Text, um die
         Größenverhältnisse zu verdeutlichen
      </div>
```

```
47        </div>
48        <div class="outer2">
49           <div class="inner3">
50              Abschnitt : etwas Text, um die
51              Größenverhältnisse zu verdeutlichen
52           </div>
53        </div>
54        <div class="outer3">
55           <div class="inner3">
56              Abschnitt : etwas Text, um die
57              Größenverhältnisse zu verdeutlichen
58           </div>
59        </div>
60     </body>
61     </html>
```

Sie sehen im Beispiel, dass der äußere, gelb eingefärbte Abschnitt seine Größe nicht verändert und der dort eingebettete Abschnitt vollständig dargestellt wird. Dabei fließt er über die Grenzen des äußeren Abschnitts hinaus.

Im unteren, nicht eingefärbten Abschnitt (CSS-Klasse `outer2`) wird dagegen die CSS-Stileigenschaft `overflow` auf den Wert `hidden` gesetzt und der eingebettete Abschnitt daher nur teilweise - entsprechend der Größe des äußeren Abschnitts - wiedergegeben.

Im dritten äußeren Abschnitt wird eine automatische Höhenberechnung eingestellt, die Angabe `overflow` ist daher ohne Bedeutung. Der Inhalt wird vollständig dargestellt, indem die Größe des umgebenden Abschnitts angepasst wird.

1.7 HTML-Elemente mit CSS positionieren

HTML-Elemente können relativ zu anderen HTML-Elementen positioniert werden, ohne damit die automatische Anordnung der HTML-Elemente grundsätzlich zu verändern. Bei absoluter Positionierung wird ein HTML-Element mit Bezug auf das HTML-Element `body` angeordnet oder mit Bezug auf ein umgebendes, relativ positioniertes Element.

Bei der normalen Ausgabe eines HTML-Dokuments auf einem Bildschirm oder einem Drucker wird die Anordnung der Elemente automatisch anhand des Elementtyps bestimmt (siehe auch Baustein HTML-Grundlagen und HTML-Elemente mit CSS positionieren):

- Block-Level-Elemente, die sich im HTML-Dokument auf gleicher Ebene befinden, werden in Blöcken untereinander angeordnet
- Inline-Elemente werden innerhalb eines Blocks zeilenweise angeordnet, Zeilenwechsel werden bei Bedarf vorgenommen.

Typische Block-Level-Elemente sind die HTML-Elemente `div` und `p`. Typische Inline-Elemente sind die Text-Elemente wie `em`, `strong` und `span`.

Diese Standardanordnung der HTML-Elemente kann mit der CSS-Stileigenschaft `position` verändert werden. Es wird folgende Syntax verwendet:

```
position ::= "position" ":" [ "static" | "relative" | "absolute" | "fixed" ] ";"
```

Die Positionierung eines HTML-Elements wird auch durch die CSS-Stileigenschaft `float` beeinflusst. Weitere Einzelheiten dazu werden in einem anderen Dokument behandelt.

Mit dem Wert `static` (**Voreinstellung**) wird eine Positionierung entsprechend der normalen, automatischen Anordnung von Block-Level-Elementen und Inline-Elementen eingestellt.

1.7.1 Relative Positionierung

Eine relative Positionierung eines HTML-Elements erzielen Sie mit dem Wert `relative`. Das HTML-Element wird dann zunächst nach dem normalen, automatischen Verfahren angeordnet, wodurch sich der Ursprungspunkt (linke obere Ecke des Elements) und seine Ausdehnung ergeben.

Wenn die automatische Anordnung abgeschlossen ist, wird das HTML-Element um den Betrag gegenüber seiner ursprünglichen Position verschoben, den die CSS-Eigenschaften `top`, `left`, `bottom`, `right` anzeigen. Andere nachfolgende HTML-Elemente werden dadurch nicht in ihrer Positionierung verändert.

Im ersten Beispiel wird die automatische Anordnung der HTML-Elemente verdeutlicht. Die dreit Abschnitte werden untereinander angeordnet, die angegebenen Weiten und Höhen werden beachtet.

Beispiel

```
<!DOCTYPE html>
<html>
<head>
   <title>Positionierung / 1</title>
   <meta charset="UTF-8" />
   <style type="text/css">
      div {
         border: 2px dashed red;
         height: 100px;
         width:80%;
      }
      #r1 {
         border: 2px solid gray;
      }
   </style>
</head>
<body>
   <div>
      Der erste Abschnitt.
   </div>
   <div id="r1">
      Der zweite Abschnitt.
   </div>
   <div>
      Der dritte Abschnitt.
   </div>
</body>
</html>
```

Im zweiten Beispiel wird der zweite, mittlere Abschnitt mit der Identifikation `r1` relativ positioniert: dazu wird eine eigene CSS-Stilregel verwendet, die sich nur auf dieses Element bezieht. Es ist gut zu erkennen, dass zunächst alle Abschnitte entsprechend der automatischen Anordnung positioniert werden und dann der mittlere Abschnitt verschoben wird. Wie Sie sehen, ist dabei eine Überlappung von Elementen möglich !

Beispiel

```
<!DOCTYPE html>
<html>
<head>
   <title>Positionierung / 2</title>
```

```
5       <meta charset="UTF-8" />
6       <style type="text/css">
7          div {
8             border: 2px dashed red;
9             height: 100px;
10            width:80%;
11         }
12         #r1 {
13            border: 2px solid gray;
14            position: relative;
15            top: 60px;
16            left: 60px;
17         }
18      </style>
19   </head>
20   <body>
21      <div>
22         Der erste Abschnitt.
23      </div>
24      <div id="r1">
25         Der zweite Abschnitt.
26      </div>
27      <div>
28         Der dritte Abschnitt.
29      </div>
30   </body>
31   </html>
```

Das dritte Beispiel verdeutlicht, dass Elemente, die in einem relativ positionierten Element vorhanden sind, mit verschoben werden.

Beispiel

```
1    <!DOCTYPE html>
2    <html>
3    <head>
4       <title>Positionierung / 3</title>
5       <meta charset="UTF-8" />
6       <style type="text/css">
7          div {
8             border: 2px dashed red;
9             height: 100px;
10            width:80%;
11         }
12         p {
13            border: 1px solid gray;
14         }
15         #r1 {
16            border: 2px solid gray;
17            position: relative;
18            top: 60px;
19            left: 60px;
20         }
21      </style>
22   </head>
23   <body>
24      <div>
25         Der erste Abschnitt.
```

```
26        </div>
27        <div id="r1">
28           <p>Absatz im zweiten Abschnitt.</p>
29        </div>
30        <div>
31           Der dritte Abschnitt.
32        </div>
33     </body>
34     </html>
```

1.7.2 Absolute Positionierung

Eine grundsätzlich andere Art der Positionierung wird mit dem CSS-Eigenschaftswert `absolute` eingestellt. Die absolute Positionierung von HTML-Elementen weist folgende Besonderheiten auf:

- die Elemente werden nicht bei der normalen, automatischen Anordnung berücksichtigt, d.h. nachfolgende Elemente werden positioniert, als ob es das absolut positionierte Element nicht gäbe
- die CSS-Eigenschaften `top`, `bottom`, `left` und `right` werden berücksichtigt.

Im Beispiel sieht man, dass der zweite Abschnitt keine Rolle bei der automatischen Anordnung der anderen Abschnitte spielt. Im absolut positionierten Abschnitt werden enthaltene Elemente mit positioniert.

Beispiel

```
1     <!DOCTYPE html>
2     <html>
3     <head>
4        <title>Positionierung / 4</title>
5        <meta charset="UTF-8" />
6        <style type="text/css">
7           div {
8              border: 2px dashed red;
9              height: 100px;
10             width:80%;
11          }
12          p {
13             border: 1px solid gray;
14          }
15          #r1 {
16             border: 2px solid gray;
17             position: absolute;
18             top: 60px;
19             left: 60px;
20          }
21       </style>
22    </head>
23    <body>
24       <div>
25          Der erste Abschnitt.
26       </div>
27       <div id="r1">
28          <p>Absatz im zweiten Abschnitt.</p>
29       </div>
30       <div>
31          Der dritte Abschnitt.
32       </div>
```

```
33    </body>
34    </html>
```

1.7.3 Absolute Positionierung bei untergeordneten Elementen

Die absolute Positionierung erfolgt immer in Bezug auf ein umgebendes Element, wenn dieses bestimmte Anforderungen erfüllt:

- es handelt sich um das HTML-Element `body`, oder
- es handelt sich um ein Block-Level-Element mit relativer oder absoluter Positionierung.

In allen anderen Fällen wird die absolute Positionierung in Bezug auf den Ursprung des HTML-Element `body` vorgenommen.

Den Unterschied können Sie anhand der beiden folgenden Beispiele erkennen:

- im ersten Beispiel enthält der mittlere Abschnitt mit der Identifikation `r1` ein absolut positioniertes HTML-Element `div` mit der Identifikation `r1sub`; da das umgebende Element (`r1`) die Positionierungsart `static` verwendet (Standardwert, da keine Angabe vorliegt), wird das Element `r1sub` in Bezug auf das HTML-Element `body` positioniert
- im zweiten Beispiel erhält das Element `r1` die Positionierungsart `relative`, damit das enthaltene Element `r1sub` jetzt in Bezug auf den Ursprung von `r1` absolut positioniert wird; da für `r1` **kein Zuwachs** in der Position angegeben wird (es gibt keine Angaben dazu), wirkt sich die relative Positionierung optisch nicht aus, sie wurde nur zur Veränderung des Bezugs der absoluten Positionierung des enthaltenen Elements eingesetzt !

Beispiel

```
1     <!DOCTYPE html>
2     <html>
3     <head>
4        <title>Positionierung / 5</title>
5        <meta charset="UTF-8" />
6        <style type="text/css">
7           div {
8              border: 2px dashed red;
9              height: 100px;
10             width:80%;
11          }
12          p {
13             border: 1px solid gray;
14          }
15          #r1 {
16             border: 2px solid gray;
17          }
18          #r1sub {
19             border: none;
20             background-color: lightgrey;
21             position: absolute;
22             top: 60px;
23             left: 60px;
24          }
25       </style>
26    </head>
27    <body>
28       <div>
29          Der erste Abschnitt.
30       </div>
31       <div id="r1">
32          <div id="r1sub">
```

```
            <p>Absatz im zweiten Abschnitt.</p>
         </div>
      </div>
      <div>
         Der dritte Abschnitt.
      </div>
   </body>
   </html>
```

Beispiel

```
<!DOCTYPE html>
<html>
<head>
   <title>Positionierung / 6</title>
   <meta charset="UTF-8" />
   <style type="text/css">
      div {
         border: 2px dashed red;
         height: 100px;
         width:80%;
      }
      p {
         border: 1px solid gray;
      }
      #r1 {
         border: 2px solid gray;
         position:relative;
      }
      #r1sub {
         border: none;
         background-color: lightgrey;
         position: absolute;
         top: 60px;
         left: 60px;
      }
   </style>
</head>
<body>
   <div>
      Der erste Abschnitt.
   </div>
   <div id="r1">
      <div id="r1sub">
         <p>Absatz im zweiten Abschnitt.</p>
      </div>
   </div>
   <div>
      Der dritte Abschnitt.
   </div>
</body>
</html>
```

1.7.4 Positionierungsart `fixed`

Die Positionierungsart `fixed` ist eine spezielle Form der absoluten Positionierung. Die zuvor erläuterten Zusammenhänge sind auch hier zu beachten. Zusätzlich wird nach der Bestimmung der absoluten Position die daraus resultierende Positionierung im Webbrowser-Fenster auch bei einem eventuellen Scrollen des Inhalts der Webseite beibehalten, das Element ist also auf dem Bildschirm optisch 'fixiert', woraus sich auch der Name der Positionierungsart erklärt. Man verwendet die Positionierungsart `fixed`, wenn man vor allem bei längeren Webseiten, die in jedem Fall ein Scrollen erfordern, Inhalte immer sichtbar machen will. Oft werden damit Hyperlinks gruppiert und fixiert, um wie ein Menü verwendet zu werden.

Beispiel

```
<!DOCTYPE html>
<html>
<head>
   <title>Positionierung / 7</title>
   <meta charset="UTF-8" />
   <style type="text/css">
      div {
         border: 2px dashed red;
         height: 100px;
         width:80%;
      }
      p {
         border: 1px solid gray;
      }
      #r1 {
         border: 2px solid gray;
         position:relative;
      }
      #r1sub {
         border: none;
         background-color: lightgrey;
         position: fixed;
         top: 60px;
         left: 60px;
      }
   </style>
</head>
<body>
   <div>
      Der erste Abschnitt.
   </div>
   <div id="r1">
      <div id="r1sub">
         <p>Absatz im zweiten Abschnitt.</p>
      </div>
   </div>
   <div>
      Der dritte Abschnitt.
   </div>
</body>
</html>
```