Validator: <https://validator.w3.org/>

**html – class**

Sollen mehrere Elemente gleich formatiert werden, dann wird das Attribut „class“ verwendet.

<p class="AbsatzItalic"> just some random text </p>

Alle Absätze, die nun mit diesem Attribut versehen werden, können im CSS gemeinsam formatiert werden:

p.AbsatzItalic {

font-style: italic;

}

Achten Sie darauf, den Punkt zwischen dem Element und dem Namen der Klasse im CSS nicht zu vergessen.

**Hinweis: Sollten Sie diesen Code direkt in den Editor kopieren, müssen Sie die Anführungszeichen neu setzen, sonst werden Sie vom Browser nicht erkannt.**

**CSS – inline; inline-block; block**

Der <span> Tags ist ein sogenannter inline-Container. Ähnlich wie <div>, aber mit dem Unterschied, dass die Elemente im <span> Container in derselben Zeile dargestellt werden. Das bedeutet, dass Höhe und Breite dem Text angepasst werden. <div> Container werden immer mit einer Absatzmarke abgeschlossen und sie werden daher untereinander dargestellt.

Der Tag <p> ist ein sogenanntes „block“ Element, das bedeutet, dass die Höhe automatisch dem Inhalt und die Breite dem Bildschirm angepasst wird.

**Beispiel – htmlCode**

<span class="SpanDisplay"> Dies ist ein Span-Element </span>

<p class ="BlockDisplay"> Dieser Absatz ist ein "p" Element </p>

**Beispiel – CSS-Code**

span.SpanDisplay {

border-style: solid;

}

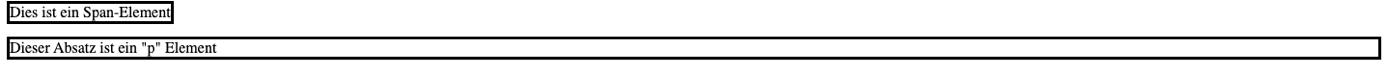
p.BlockDisplay {

border-style: solid;

**}**

*Hinweis: Es ist auf Websiten sehr unüblich und nicht sehr schön, wenn Rahmen verwendet werden. Der property „border-style“ wird oft verwendet, um die Grenzen der Box-Elemente beim Entwicklen zu sehen. Sie werden üblicherweise vor der Veröffentlichung einer Seite entfernt.*

Wenn die Seite angezeigt wird, ist der Unterschied zwischen inline und block offensichtlich:



Voreinstellungen, wie diese, dass <p> ein Block Element ist, nennt man default-Werte. Inline und block können über das CSS Property „display“ verändert werden. Im untenstehenden Beispiel wurde der Absatz „BlockDisplay“ auf inline geändert:

**Beispiel – CSS-Code**

span.SpanDisplay {

border-style: solid;

}

p.BlockDisplay {

border-style: solid;

display: inline;

}

**Aktualisierte Darstellung:**



Das <p> Element wurde nun in derselben Zeile positioniert und die Breite dem Text angepasst.

Folgende display Werte sind nützlich:

* inline: siehe oben
* block: Höhe wird dem Inhalt angepasst, Breite entspricht dem Bildschirm. Höhe und Breite können mit „width“ und „height“ angpasst werden.
* inline-block: Ähnlich wie „block“ nur, dass nicht automatisch ein Zeilenumbruch erfolgt.

Im folgenden Beispiel wurden beide Elemente von oben mit inline-block formatiert:

**Beispiel – CSS-Code**

span.SpanDisplay {

border-style: solid;

display: inline-block;

width: 200px;

height: 200px;

}

p.BlockDisplay {

border-style: solid;

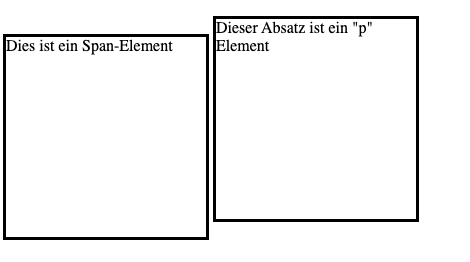
display: inline-block;

width: 200px;

height: 200px;

}

**Aktualisierte Darstellung:**



Bei <span> Containern wird davor und danach automatisch ein Zeilenumbruch gesetzt, daher ist wird er tiefer gesetzt dargestellt.

**CSS – Positionierung von Elementen**

Für das folgende Beispiel wurde mit folgendem Code gestartet:

**Beispiel html**

<p class="AbsatzLinks"> Dieser Absatz ist meistens Links, aber nicht immer. Und wenn nicht, dann wird er woanders dargestellt. </p>

<p class ="AbsatzRechts"> Das ist der andere Absatz und hat weniger Text. </p>

**Beispiel css**

p.AbsatzLinks {

border: solid;

display: inline-block;

vertical-align: top;

width: 40%;

height: 200px;

}

p.AbsatzRechts {

border:solid;

display: inline-block;

vertical-align: top;

margin-left: 20px;

width: 40%;

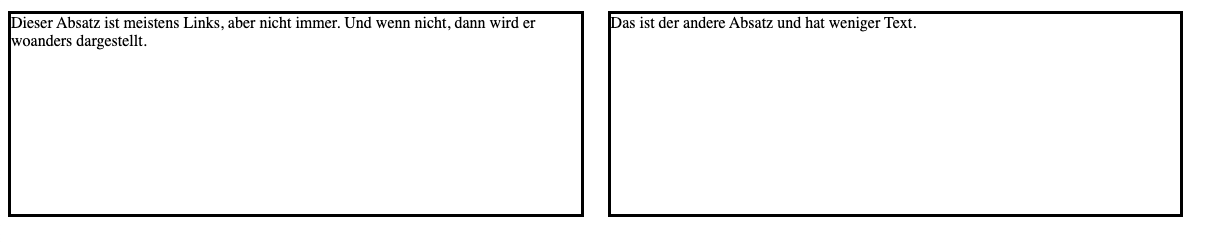
height: 200px;

}

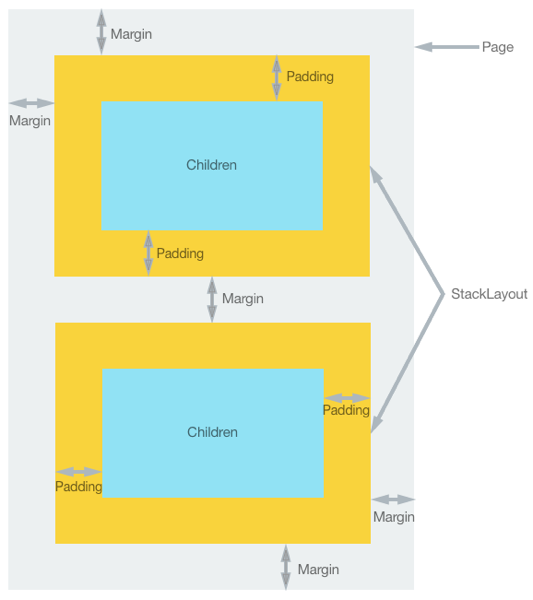
*Hinweis: Der property „vertical-align“ ist notwendig, damit beide Boxen auf derselben Linie dargestellt werden. Die Breite der Absätze wird in Abhängigkeit zum Bildschirm gewählt. Die Gefahr bei der absoluten Definition der Breite (mit px) ist, dass die Boxen z.B.: am Handy über den Rand des Bildschirms hinausragen.*

*Hinweis zu „border“: Der property „border“ nimmt mehrere Werte. Der 1. Wert entspricht dem border-style, der 2. Wäre die border-width.*

**Darstellung**



Um den Abstand zu definieren, werden oft die css-properties „margin“ und „padding“ verwendet. Der Unterschied ist im folgenden Bild dargestellt.



*Hinweis: Die Bezeichnung „parent“ und „children“ in Code bedeutet, dass die Formatierung für das „parent“-Element definiert wird (z.B.: <body>). Alle Elemente in dem „parent“ (z.B.: jedes <p> Element) erben die Formatierung.*

Im folgenden css-Code wird beispielsweise der gesamte der Inhalt der Seite etwas nach innen gerückt.

body {

margin: 4%;

}

Der Prozentwert bezieht sich dabei auf die Breite des Bildschirms des Ausgabegeräts.

**CSS – overflow**

Sollte der Inhalt einer Box aus Platzgründen nicht komplett dargestellt werden können, kann mit dem property „overflow“ definiert werden, was mit dem überstehenden Inhalt geschehen soll.

p {

overflow: scroll;

}

Im Beispiel wird eine Scroll-Leiste hinzugefügt, damit der gesamte Text gelesen werden kann.

**CSS – links**

Links verändern sich, zum Beispiel soll die Nutzerin sehen, auf welche Links sie schon geklickt hat oder es soll eine kurze Beschreibung angezeigt werden.

**Beispiel html**

<body>

<a class="LinkSchule" href="https://www.bhakwien22.at"> Website der BHAKWien22 </a>

</body>

**Beispiel css**

a.LinkSchule {

text-decoration: none;

}

Mit dem Selektor “text-decoration” kann festgelegt werden, ob der Link unterstrichen werden soll. Wird der Wert auf None gesetzt, wird der Link nicht unterstrichen.

Es gibt eine weitere Form von Selektoren, die oft bei Links verwendet werden und sich auf den Status des Links beziehen:

a:visited {Formatierung nach anklicken)

a:hover {Formatierung, wenn sich der Mauszeiger über dem Link befindet}

Beispiel:

a:hover {  
 color: hotpink;  
}