Kostenloses eBook

LERNEN CSS

Free unaffiliated eBook created from
Stack Overflow contributors.

#css

Inhaltsverzeichnis

Über

You can share this PDF with anyone you feel could benefit from it, downloaded the latest version from: css

It is an unofficial and free CSS ebook created for educational purposes. All the content is extracted from Stack Overflow Documentation, which is written by many hardworking individuals at Stack Overflow. It is neither affiliated with Stack Overflow nor official CSS.

Use the content presented in this book at your own risk; it is not guaranteed to be correct nor accurate, please send your feedback and corrections to info@zzzprojects.com

Kapitel 1: Erste Schritte mit CSS

Bemerkungen

Stile können auf verschiedene Arten verfasst werden, sodass die Wiederverwendung und der Gültigkeitsbereich unterschiedlich sind, wenn sie in einem HTML-Quelldokument angegeben werden. *Externe* Stylesheets können in HTML-Dokumenten wiederverwendet werden. *Eingebettete* Stylesheets gelten für das gesamte Dokument, in dem sie angegeben werden. *Inline*-Stile gelten nur für die einzelnen HTML-Elemente, für die sie angegeben werden.

Versionen

Ausführung	Veröffentlichungsdatum
1	1996-12-17
2	1998-05-12
3	2015-10-13

Examples

Externes Stylesheet

Ein externes CSS-Stylesheet kann auf eine beliebige Anzahl von HTML-Dokumenten angewendet werden, indem in jedem HTML-Dokument ein `<link>` -Element eingefügt wird.

Das Attribut `rel` des `<link>` -Tags muss auf `"stylesheet"` und das `href` Attribut auf den relativen oder absoluten Pfad zum Stylesheet gesetzt werden. Während die Verwendung relativer URL-Pfade im Allgemeinen als bewährte Methode gilt, können auch absolute Pfade verwendet werden. In HTML5 kann das `type` Attribut weggelassen werden .

Es wird empfohlen, das `<link>` -Tag im `<head>` -Tag der HTML-Datei zu platzieren, damit die Stile vor den Elementen geladen werden, die sie formatieren. Andernfalls sehen Benutzer einen Blitz mit nicht gestylten Inhalten .

Beispiel

hallo-world.html

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8" />
```

```
        <link rel="stylesheet" type="text/css" href="style.css">
    </head>
    <body>
        <h1>Hello world!</h1>
        <p>I ♥ CSS</p>
  </body>
</html>
```

style.css

```
h1 {
    color: green;
    text-decoration: underline;
}
p {
    font-size: 25px;
    font-family: 'Trebuchet MS', sans-serif;
}
```

Stellen Sie sicher, dass Sie den korrekten Pfad zu Ihrer CSS-Datei in der Datei href angeben. Befindet sich die CSS-Datei im selben Ordner wie Ihre HTML-Datei, ist kein Pfad erforderlich (wie im obigen Beispiel). Wenn sie jedoch in einem Ordner gespeichert wird, geben Sie diesen wie `href="foldername/style.css"` .

```
<link rel="stylesheet" type="text/css" href="foldername/style.css">
```

Externe Stylesheets gelten als die beste Methode, um mit CSS zu arbeiten. Dafür gibt es einen ganz einfachen Grund: Wenn Sie eine Site mit beispielsweise 100 Seiten verwalten, die alle von einem einzigen Stylesheet gesteuert werden und Ihre Linkfarben von Blau auf Grün ändern möchten, ist die Änderung viel einfacher in Ihrer CSS-Datei und lassen Sie die Änderungen auf allen 100 Seiten "kaskadieren", als auf 100 separate Seiten zu gehen und die gleiche Änderung 100 Mal vorzunehmen. Wenn Sie das Erscheinungsbild Ihrer Website vollständig ändern möchten, müssen Sie nur diese eine Datei aktualisieren.

Sie können beliebig viele CSS-Dateien in Ihre HTML-Seite laden.

```
<link rel="stylesheet" type="text/css" href="main.css">
<link rel="stylesheet" type="text/css" href="override.css">
```

CSS-Regeln werden mit einigen Grundregeln angewendet, und die Reihenfolge spielt eine Rolle. Wenn Sie beispielsweise eine main.css-Datei mit etwas Code darin haben:

```
p.green { color: #00FF00; }
```

Alle Ihre Absätze mit der 'grünen' Klasse werden in hellem Grün geschrieben. Sie können dies jedoch mit einer anderen .css-Datei überschreiben, indem Sie sie *nach* main.css einfügen. Sie können override.css mit dem folgenden Code nach main.css haben, zum Beispiel:

```
p.green { color: #006600; }
```

Jetzt werden alle Ihre Absätze mit der 'grünen' Klasse dunkler als hellgrün geschrieben.

Es gelten andere Prinzipien wie die! Wichtige Regel, Spezifität und Vererbung.

Wenn jemand Ihre Website zum ersten Mal besucht, lädt der Browser den HTML-Code der aktuellen Seite und die verknüpfte CSS-Datei herunter. Wenn sie dann zu einer anderen Seite navigieren, muss ihr Browser nur den HTML-Code dieser Seite herunterladen. Die CSS-Datei wird zwischengespeichert und muss daher nicht erneut heruntergeladen werden. Da Browser das externe Stylesheet zwischenspeichern, werden Ihre Seiten schneller geladen.

Interne Stile

CSS, das in `<style></style>` -Tags innerhalb eines HTML-Dokuments eingeschlossen ist, funktioniert wie ein externes Stylesheet, außer dass es im HTML-Dokument lebt, das es formatiert, und nicht in einer separaten Datei. Es kann daher nur auf das Dokument angewendet werden, in dem es enthalten ist lebt. Beachten Sie, dass dieses Element im Inneren der sein *muss* `<head>` Element für HTML - Validierung (obwohl es in allen gängigen Browsern funktionieren , wenn in platziert `body`).

```
<head>
    <style>
        h1 {
            color: green;
            text-decoration: underline;
        }
        p {
            font-size: 25px;
            font-family: 'Trebuchet MS', sans-serif;
        }
    </style>
</head>
<body>
    <h1>Hello world!</h1>
    <p>I ♥ CSS</p>
</body>
```

Inline-Styles

Verwenden Sie Inline-Stile, um den Stil auf ein bestimmtes Element anzuwenden. Beachten Sie, dass dies ***nicht*** optimal ist. Das Platzieren von Stilregeln in einem `<style>` -Tag oder einer externen CSS-Datei wird empfohlen, um eine Unterscheidung zwischen Inhalt und Präsentation zu gewährleisten.

Inline-Stile überschreiben jedes CSS in einem `<style>` -Tag oder einem externen `<style>` . Während dies unter Umständen nützlich sein kann, verringert diese Tatsache in den meisten Fällen die Wartbarkeit eines Projekts.

Die Stile im folgenden Beispiel gelten direkt für die Elemente, an die sie angehängt sind.

```
<h1 style="color: green; text-decoration: underline;">Hello world!</h1>
<p style="font-size: 25px; font-family: 'Trebuchet MS';">I ♥ CSS</p>
```

Inline-Stile sind im Allgemeinen die sicherste Methode, um die Kompatibilität zwischen verschiedenen E-Mail-Clients, -Programmen und -geräten zu gewährleisten. Das Schreiben kann jedoch zeitaufwändig sein und ist in der Verwaltung etwas schwierig.

CSS @import-Regel (eine der CSS-Regeln)

Die @import-CSS-Regel at-rule wird zum Importieren von Stilregeln aus anderen Stylesheets verwendet. Diese Regeln müssen allen anderen Arten von Regeln vorangehen, mit Ausnahme von @charset-Regeln. Da es sich nicht um eine verschachtelte Anweisung handelt, kann @import nicht innerhalb von at-rules der bedingten Gruppe verwendet werden. `@import` .

Wie benutze ich @import?

Sie können die @ import-Regel auf folgende Arten verwenden:

A. Mit interner Stilmarke

```
   <style>
    @import url('/css/styles.css');
   </style>
```

B. Mit externem Stylesheet

Die folgende Zeile importiert eine CSS-Datei mit dem Namen `additional-styles.css` im Stammverzeichnis in die CSS-Datei, in der sie angezeigt wird:

```
@import '/additional-styles.css';
```

Der Import von externem CSS ist ebenfalls möglich. Ein häufiger Anwendungsfall sind Schriftdateien.

```
@import 'https://fonts.googleapis.com/css?family=Lato';
```

Ein optionales zweites Argument für die `@import` Regel ist eine Liste von Medienabfragen:

```
@import '/print-styles.css' print;
@import url('landscape.css') screen and (orientation:landscape);
```

CSS mit JavaScript ändern

Reines JavaScript

Es ist möglich , hinzuzufügen, zu entfernen oder CSS - Eigenschaft Werte mit JavaScript durch ein Element ändert `style` Eigenschaft.

```
var el = document.getElementById("element");
el.style.opacity = 0.5;
```

```
el.style.fontFamily = 'sans-serif';
```

Beachten Sie, dass Stileigenschaften im unteren Kamelfallstil benannt werden. In dem Beispiel sehen Sie, dass die css-Eigenschaft `font-family` in Javascript `fontFamily` wird.

Alternativ zur direkten Bearbeitung von Elementen können Sie in JavaScript ein `<style>` oder `<link>` -Element erstellen und an den `<body>` oder `<head>` des HTML-Dokuments anhängen.

jQuery

Das Ändern von CSS-Eigenschaften mit jQuery ist noch einfacher.

```
$('#element').css('margin', '5px');
```

Wenn Sie mehr als eine Stilregel ändern müssen:

```
$('#element').css({
    margin: "5px",
    padding: "10px",
    color: "black"
});
```

jQuery bietet zwei Möglichkeiten zum Ändern von CSS-Regeln, die Bindestriche enthalten (z. B. `font-size`). Sie können sie in Anführungszeichen oder den Namen der Stilregel setzen.

```
$('.example-class').css({
    "background-color": "blue",
    fontSize: "10px"
});
```

Siehe auch

- JavaScript-Dokumentation - CSS-Stil lesen und ändern .
- jQuery-Dokumentation - CSS-Manipulation

Stylinglisten mit CSS

Es gibt drei verschiedene Eigenschaften für die Gestaltung von Listenelementen: `list-style-type` , `list-style-image` und `list-style-position` , die in dieser Reihenfolge deklariert werden sollten. Die Standardwerte sind "disc", "Outside" und "None". Jede Eigenschaft kann separat deklariert werden oder mithilfe der Kurzschreibeigenschaft im `list-style` .

`list-style-type` definiert die Form oder den Typ des Aufzählungspunkts, der für jedes Listenelement verwendet wird.

Einige der zulässigen Werte für den Listentyp `list-style-type` :

- Scheibe
- Kreis
- Quadrat
- Dezimal
- niederrömisch
- Oberrömisch
- keiner

(Eine vollständige Liste finden Sie im Wiki für W3C-Spezifikationen.)

Um quadratische Aufzählungspunkte für jedes Listenelement zu verwenden, würden Sie beispielsweise das folgende Eigenschaftswertpaar verwenden:

```
li {
    list-style-type: square;
}
```

Die Eigenschaft **`list-style-image`** bestimmt, ob das Symbol für das Listenelement mit einem Bild festgelegt ist, und akzeptiert den Wert `none` oder eine URL, die auf ein Bild verweist.

```
li {
    list-style-image: url(images/bullet.png);
}
```

Die Eigenschaft **`list-style-position`** legt fest, wo der Listenelement-Marker positioniert werden soll. Er akzeptiert einen der beiden Werte: "inside" oder "outside".

```
li {
  list-style-position: inside;
}
```

Erste Schritte mit CSS online lesen: https://riptutorial.com/de/css/topic/293/erste-schritte-mit-css

Kapitel 2: 2D-Transformationen

Syntax

- **Transformation drehen**
- umwandeln: drehen (<Winkel>)
- **Übersetzen Sie Transform**
- transform: übersetzen (<Länge oder Prozentsatz> [, <Länge oder Prozentsatz>]?)
- transform: translateX (<länge oder prozentual>)
- transform: translateY (<länge oder prozentual>)
- **Schiefe verwandeln**
- Transformation: Schräglauf (<Winkel> [, <Winkel>]?)
- transform: skewX (<Winkel>)
- transform: skewY (<angle>)
- **Scale Transform**
- Transformation: Maßstab (<Skalierungsfaktor> [, <Skalierungsfaktor>]?)
- transform: scaleX (<Skalierungsfaktor>)
- transform: scaleY (<scale-factor>)
- **Matrix-Transformation**
- transform: matrix (<number> [, <number>] {5,5})

Parameter

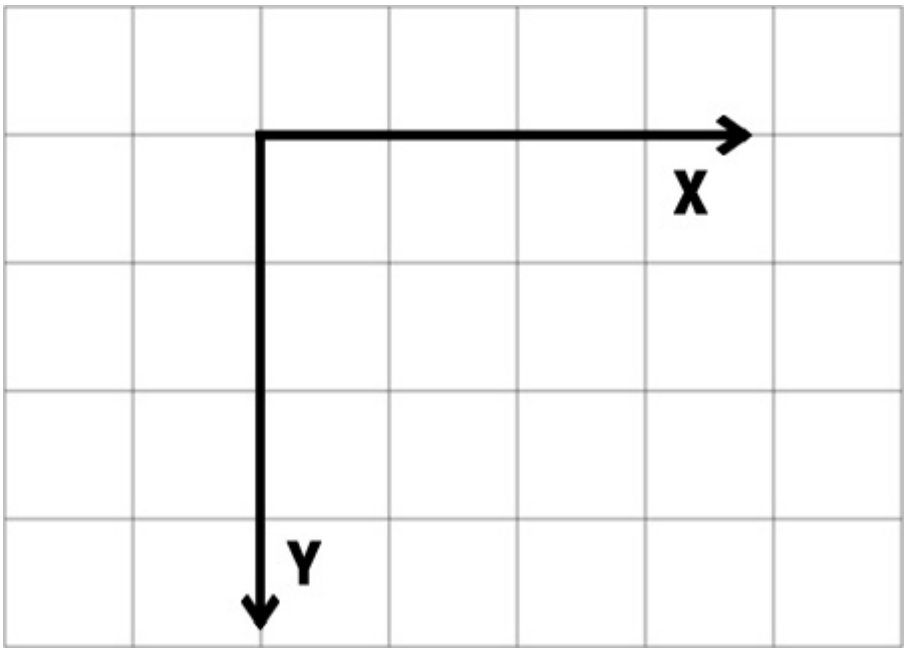
Funktion / Parameter	Einzelheiten
`rotate(x)`	Definiert eine Transformation, mit der das Element um einen festen Punkt auf der Z-Achse verschoben wird
`translate(x,y)`	Verschiebt die Position des Elements auf der X- und Y-Achse
`translateX(x)`	Verschiebt die Position des Elements auf der X-Achse
`translateY(y)`	Verschiebt die Position des Elements auf der Y-Achse
`scale(x,y)`	Ändert die Größe des Elements auf der X- und Y-Achse
`scaleX(x)`	Ändert die Größe des Elements auf der X-Achse
`scaleY(y)`	Ändert die Größe des Elements auf der Y-Achse
`skew(x,y)`	Scherabbildung oder Transvektion, die jeden Punkt eines Elements um einen bestimmten Winkel in jede Richtung verzerrt
`skewX(x)`	Horizontale Scherabbildung, die jeden Punkt eines Elements um einen bestimmten Winkel in horizontaler Richtung verzerrt

Funktion / Parameter	Einzelheiten
`skewY(y)`	Vertikale Scherabbildung, die jeden Punkt eines Elements um einen bestimmten Winkel in vertikaler Richtung verzerrt
`matrix()`	Definiert eine 2D-Transformation in Form einer Transformationsmatrix.
Winkel	Der Winkel, um den das Element gedreht oder geneigt werden soll (abhängig von der Funktion, mit der es verwendet wird). Der Winkel kann in Grad (`deg`), Gradian (`grad`), Radiant (`rad`) oder `turn` (`turn`) angegeben werden. In der Funktion `skew()` ist der zweite Winkel optional. Wenn nicht angegeben, gibt es keine (0) Schräglage in der Y-Achse.
Länge oder Prozentsatz	Die als Länge oder Prozentsatz ausgedrückte Entfernung, um die das Element übersetzt werden soll. In der Funktion `translate()` ist der zweite Längen- oder Prozentsatz optional. Wenn nicht angegeben, gibt es keine 0-Verschiebung in der Y-Achse.
Skalierungsfaktor	Eine Zahl, die definiert, wie oft das Element in der angegebenen Achse skaliert werden soll. Bei der Funktion `scale()` ist der zweite Skalierungsfaktor optional. Wenn nicht angegeben, wird der erste Skalierungsfaktor auch für die Y-Achse angewendet.

Bemerkungen

2D-Koordinatensystem

Transformationen werden nach einem 2D-X / Y-Koordinatensystem durchgeführt. Die X-Achse geht von rechts nach links und die Y-Achse geht nach unten, wie in der folgenden Abbildung dargestellt:



Ein positives `translateY()` geht also nach unten und ein positives `translateX()` geht richtig.

Browserunterstützung und Präfixe

- IE unterstützt diese Eigenschaft seit IE9 mit dem Präfix `-ms-` . Ältere Versionen und Edge benötigen das Präfix nicht
- Firefox unterstützt es seit Version 3.5 und benötigt bis Version 15 das Präfix `-moz-`
- Chrome seit Version 4 und bis Version 34 benötigt das Präfix `-webkit-`
- Safari benötigt das Präfix `-webkit-` bis Version 8
- Opera benötigt das Präfix `-o-` für Version 11.5 und das Präfix `-webkit-` von Version 15 bis 22
- Android benötigt das Präfix `-webkit-` von Version 2.1 bis 4.4.4

Beispiel für eine vorangestellte Transformation:

```
-webkit-transform: rotate(45deg);
    -ms-transform: rotate(45deg);
        transform: rotate(45deg);
```

Examples

Drehen

HTML

```
<div class="rotate"></div>
```

CSS

```
.rotate {
```

```
    width: 100px;
    height: 100px;
    background: teal;
    transform: rotate(45deg);
}
```

In diesem Beispiel wird das DIV um 45 Grad im Uhrzeigersinn gedreht. Das Rotationszentrum befindet sich in der Mitte des div, `50%` von links und `50%` von oben. Sie können den Drehpunkt ändern, indem Sie die Eigenschaft `transform-origin` festlegen.

```
transform-origin: 100% 50%;
```

Das obige Beispiel setzt den Drehpunkt auf die Mitte des rechten Endes.

Rahmen

HTML

```
<div class="scale"></div>
```

CSS

```
.scale {
    width: 100px;
    height: 100px;
    background: teal;
    transform: scale(0.5, 1.3);
}
```

In diesem Beispiel wird das div auf `100px * 0.5 = 50px` auf der X-Achse und auf `100px * 1.3 = 130px` auf der Y-Achse `100px * 1.3 = 130px` .

Der Mittelpunkt der Transformation befindet sich in der Mitte des Bereichs, `50%` von links und `50%` von oben.

Übersetzen

HTML

```
<div class="translate"></div>
```

CSS

```
.translate {
    width: 100px;
    height: 100px;
    background: teal;
    transform: translate(200px, 50%);
}
```

In diesem Beispiel wird das div um 200px auf der X-Achse und um `100px * 50% = 50px` auf der Y-Achse `100px * 50% = 50px` .

Sie können auch Übersetzungen für eine einzelne Achse angeben.

Auf der X-Achse:

```
.translate {
    transform: translateX(200px);
}
```

Auf der Y-Achse:

```
.translate {
    transform: translateY(50%);
}
```

Neigung

HTML

```
<div class="skew"></div>
```

CSS

```
.skew {
    width: 100px;
    height: 100px;
    background: teal;
    transform: skew(20deg, -30deg);
}
```

In diesem Beispiel wird das div auf der X-Achse um 20 Grad und auf der Y-Achse um - 30 Grad verzerrt.
Der Mittelpunkt der Transformation befindet sich in der Mitte des Bereichs, `50%` von links und `50%` von oben.

Sehen Sie das Ergebnis hier .

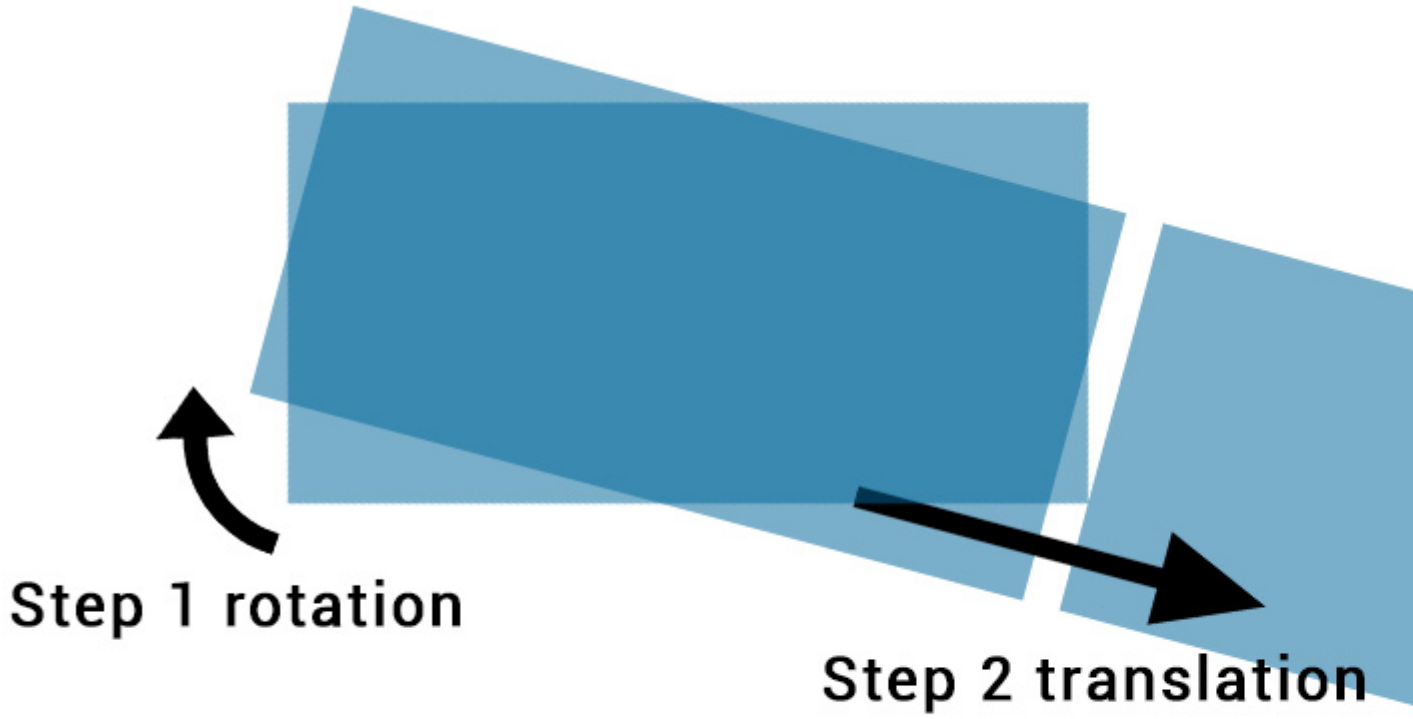
Mehrere Transformationen

Mehrere Transformationen können wie folgt auf ein Element in einer Eigenschaft angewendet werden:

```
transform: rotate(15deg) translateX(200px);
```

Dadurch wird das Element um 15 Grad im Uhrzeigersinn gedreht und dann um 200 Pixel nach rechts verschoben.

Bei verketteten Transformationen **bewegt sich das Koordinatensystem mit dem Element** . Dies bedeutet, dass die Übersetzung nicht horizontal ist, sondern um eine Achse um 15 Grad im Uhrzeigersinn gedreht wird, wie in der folgenden Abbildung dargestellt:



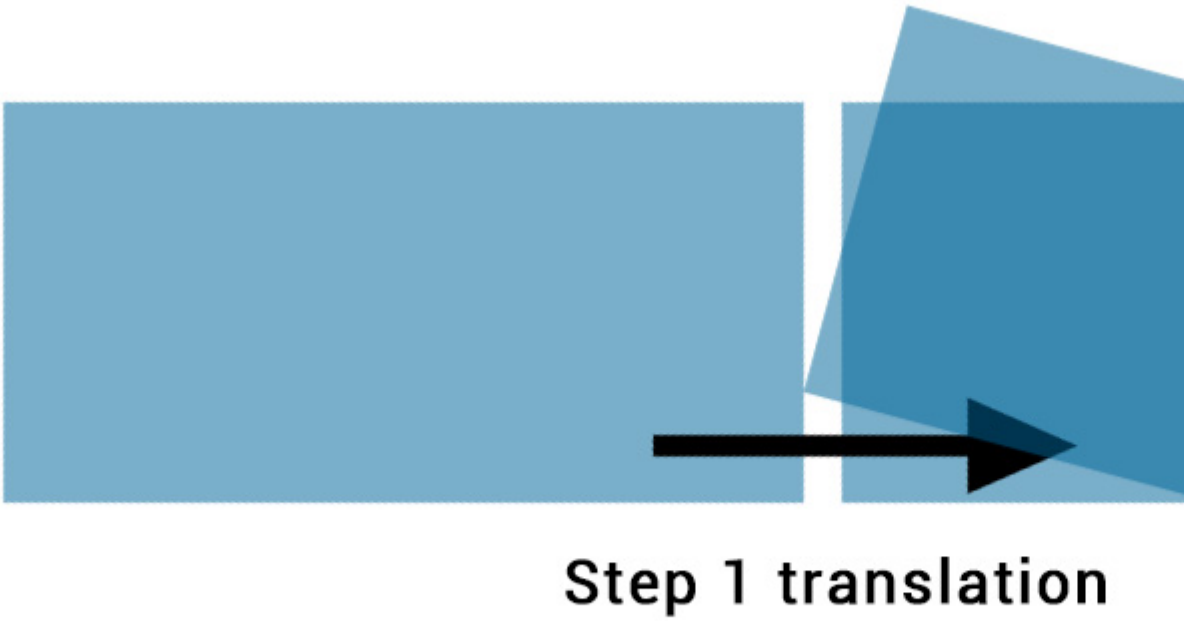
Wenn Sie die Reihenfolge der Transformationen ändern, wird die Ausgabe geändert. Das erste Beispiel wird anders sein als

```
transform: translateX(200px) rotate(15deg);
```

```
<div class="transform"></div>
```

```
.transform {
  transform: rotate(15deg) translateX(200px);
}
```

Wie in diesem Bild gezeigt:



Ursprung umwandeln

Transformationen werden in Bezug auf einen Punkt durchgeführt, der durch die Eigenschaft `transform-origin` definiert wird.

Die Eigenschaft nimmt 2 Werte an: `transform-origin: XY;`

Im folgenden Beispiel wird das erste div (`.tl`) mit `transform-origin: 0 0;` um die obere linke Ecke gedreht `transform-origin: 0 0;` und das zweite (`.tr`) wird um seine obere rechte Ecke mit dem `transform-origin: 100% 0` . Die Drehung wird beim **Schwebeflug** angewendet:

HTML:

```
<div class="transform originl"></div>
<div class="transform origin2"></div>
```

CSS:

```
.transform {
    display: inline-block;
    width: 200px;
    height: 100px;
    background: teal;
    transition: transform 1s;
}
```

```
.origin1 {
    transform-origin: 0 0;
}

.origin2 {
    transform-origin: 100% 0;
}

.transform:hover {
    transform: rotate(30deg);
}
```

Der Standardwert für die Eigenschaft transform-origin beträgt `50% 50%` ist die Mitte des Elements.

2D-Transformationen online lesen: https://riptutorial.com/de/css/topic/938/2d-transformationen

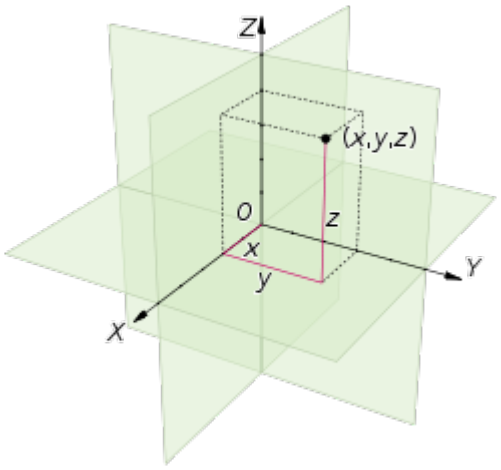
Kapitel 3: 3D-Transformationen

Bemerkungen

Koordinatensystem

3D-Transformationen werden gemäß einem `(x, y, z)` -Koordinatensystem im euklidischen Raum durchgeführt .

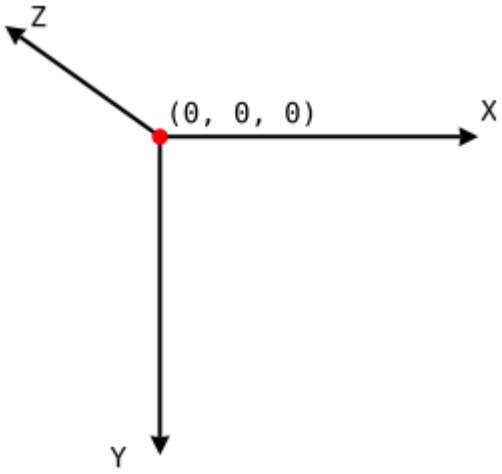
Das folgende Bild zeigt ein Beispiel für Koordinaten im euklidischen Raum:



In CSS

- Die `x` Achse repräsentiert die Horizontale (links und rechts)
- Die `y` Achse repräsentiert die Vertikale (auf und ab)
- Die `z` Achse repräsentiert die Tiefe (vorwärts und rückwärts / näher und weiter)

Das folgende Bild zeigt, wie diese Koordinaten in CSS übersetzt werden:



Examples

3D Würfel

3D-Transformationen können zum Erstellen vieler 3D-Formen verwendet werden. Hier ist ein einfaches Beispiel eines 3D-CSS-Cubes:

HTML:

```
<div class="cube">
  <div class="cubeFace"></div>
  <div class="cubeFace face2"></div>
</div>
```

CSS:

```
body {
  perspective-origin: 50% 100%;
  perspective: 1500px;
  overflow: hidden;
}
.cube {
  position: relative;
  padding-bottom: 20%;
  transform-style: preserve-3d;
  transform-origin: 50% 100%;
  transform: rotateY(45deg) rotateX(0);
}
.cubeFace {
  position: absolute;
  top: 0;
  left: 40%;
  width: 20%;
  height: 100%;
  margin: 0 auto;
  transform-style: inherit;
  background: #C52329;
  box-shadow: inset 0 0 0 5px #333;
  transform-origin: 50% 50%;
  transform: rotateX(90deg);
  backface-visibility: hidden;
}
.face2 {
  transform-origin: 50% 50%;
  transform: rotatez(90deg) translateX(100%) rotateY(90deg);
}
.cubeFace:before, .cubeFace:after {
  content: '';
  position: absolute;
  width: 100%;
  height: 100%;
  transform-origin: 0 0;
  background: inherit;
  box-shadow: inherit;
  backface-visibility: inherit;
}
.cubeFace:before {
  top: 100%;
  left: 0;
  transform: rotateX(-90deg);
```

```
}
.cubeFace:after {
  top: 0;
  left: 100%;
  transform: rotateY(90deg);
}
```

Dieses Beispiel anzeigen
In der Demo wird zusätzliches Styling hinzugefügt, und beim Schweben wird eine Transformation angewendet, um die 6 Flächen des Würfels anzuzeigen.

Sollte bemerkt werden, dass:

- 4 Gesichter werden mit Pseudoelementen erstellt
- verkettete Transformationen werden angewendet

Sicht auf die Rückseite

Die `backface-visibility` bezieht sich auf 3D-Transformationen.

Mit 3D-Transformationen und der `backface-visibility` können Sie ein Element so drehen, dass die ursprüngliche Vorderseite eines Elements nicht mehr zum Bildschirm zeigt.

Dies würde beispielsweise ein Element vom Bildschirm wegklappen:

JSFIDDLE

```
<div class="flip">Loren ipsum</div>
<div class="flip back">Lorem ipsum</div>
```

```
.flip {
  -webkit-transform: rotateY(180deg);
  -moz-transform: rotateY(180deg);
  -ms-transform: rotateY(180deg);
  -webkit-backface-visibility: visible;
  -moz-backface-visibility:    visible;
  -ms-backface-visibility:     visible;
}

.flip.back {
  -webkit-backface-visibility: hidden;
  -moz-backface-visibility:    hidden;
  -ms-backface-visibility:     hidden;
}
```

Firefox 10+ und IE 10+ unterstützen die `backface-visibility` ohne Präfix. Opera, Chrome, Safari, iOS und Android benötigen alle die `-webkit-backface-visibility` .

Es hat 4 Werte:

1. **visible** (Standardeinstellung) - Das Element ist immer sichtbar, auch wenn Sie nicht auf den Bildschirm gerichtet sind.
2. **ausgeblendet** - das Element ist nicht sichtbar, wenn es nicht auf den Bildschirm gerichtet

ist.
3. **vererben** - Die Eigenschaft erhält ihren Wert vom übergeordneten Element
4. **initial** - setzt die Eigenschaft auf ihren Standard, der sichtbar ist

Kompasszeiger oder Nadelform mithilfe von 3D-Transformationen

CSS

```
div.needle {
  margin: 100px;
  height: 150px;
  width: 150px;
  transform: rotateY(85deg) rotateZ(45deg);
  /* presentational */
  background-image: linear-gradient(to top left, #555 0%, #555 40%, #444 50%, #333 97%);
  box-shadow: inset 6px 6px 22px 8px #272727;
}
```

HTML

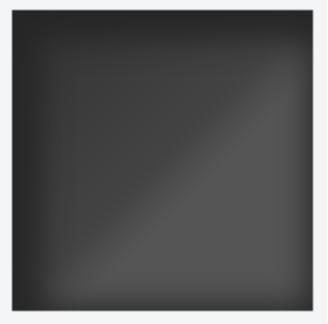
```
<div class='needle'></div>
```

Im obigen Beispiel wird eine Nadel- oder Kompasszeigerform mithilfe von 3D-Transformationen erstellt. Wenn wir die `rotate` auf ein Element anwenden, geschieht die Rotation im Allgemeinen nur in der Z-Achse, und im besten Fall erhalten wir nur Rauten. Wenn jedoch eine `rotateY` Transformation hinzugefügt wird, `rotateY` sich das Element in der Y-Achse und sieht somit aus wie eine Nadel. Je mehr die Y-Achse gedreht wird, desto stärker wirkt das Element.

Die Ausgabe des obigen Beispiels wäre eine Nadel, die auf ihrer Spitze ruht. Zum Erstellen einer Nadel, die auf ihrer Basis ruht, sollte die Rotation entlang der X-Achse und nicht entlang der Y-Achse erfolgen. Der Wert der `transform` müsste also so etwas wie `rotateX(85deg) rotateZ(45deg);` .

Dieser Stift verwendet einen ähnlichen Ansatz, um etwas zu kreieren, das dem Safari-Logo oder einem Kompasszifferblatt ähnelt.

Screenshot eines Elements ohne Transformation:



Screenshot eines Elements mit nur 2D-Transformation:

Screenshot eines Elements mit 3D-Transformation:

Effekt des Textes 3D mit Schatten

HTML:

```
<div id="title">
  <h1 data-content="HOVER">HOVER</h1>
</div>
```

CSS:

```
*{margin:0;padding:0;}
html,body{height:100%;width:100%;overflow:hidden;background:#0099CC;}
#title{
  position:absolute;
  top:50%; left:50%;
  transform:translate(-50%,-50%);
  perspective-origin:50% 50%;
  perspective:300px;
}
h1{
  text-align:center;
  font-size:12vmin;
  font-family: 'Open Sans', sans-serif;
  color:rgba(0,0,0,0.8);
  line-height:1em;
```

```
  transform:rotateY(50deg);
  perspective:150px;
  perspective-origin:0% 50%;
}
h1:after{
  content:attr(data-content);
  position:absolute;
  left:0;top:0;
  transform-origin:50% 100%;
  transform:rotateX(-90deg);
  color:#0099CC;
}
#title:before{
  content:'';
  position:absolute;
  top:-150%; left:-25%;
  width:180%; height:328%;
  background:rgba(255,255,255,0.7);
  transform-origin: 0 100%;
  transform: translatez(-200px) rotate(40deg) skewX(35deg);
  border-radius:0 0 100% 0;
}
```

Beispiel mit zusätzlichem Hover-Effekt anzeigen

In diesem Beispiel wird der Text so umgewandelt, dass er aussieht, als würde er vom Benutzer weg auf den Bildschirm gehen.

Der Schatten wird entsprechend transformiert, sodass er dem Text folgt. Da es mit einem Pseudoelement und dem `data` wird, erbt es die Transformationen von ihrem übergeordneten

Element (dem H1-Tag).

Das weiße "Licht" wird mit einem `#title` Element `#title` . Es ist schief und verwendet den Randradius für die abgerundete Ecke.

3D-Transformationen online lesen: https://riptutorial.com/de/css/topic/2446/3d-transformationen

Kapitel 4: Animationen

Syntax

- `transition: <property> <duration> <timing-function> <delay>;`
- `@keyframes <identifier>`
- `[ [ from | to | <percentage> ] [, from | to | <percentage> ]* block ]*`

Parameter

Übergang	
Parameter	Einzelheiten
Eigentum	Entweder die CSS-Eigenschaft, auf die umgeschaltet werden soll, oder `all` , die alle übergangsfähigen Eigenschaften angibt.
Dauer	Übergangszeit, entweder in Sekunden oder Millisekunden.
Timing-Funktion	Gibt eine Funktion an, die definiert, wie Zwischenwerte für Eigenschaften berechnet werden. Übliche Werte sind `ease` , `linear` und `step-end` . Weitere Informationen finden Sie in der Entlastungsfunktion .
verzögern	Wartezeit in Sekunden oder Millisekunden, bevor die Animation abgespielt wird.
@keyframes	
[von | zu | `<percentage>`]	Sie können entweder eine festgelegte Zeit mit einem Prozentwert oder zwei Prozentwerte (`10%, 20%`) für einen Zeitraum angeben, in dem die festgelegten Attribute des Keyframes festgelegt werden.
`block`	Beliebig viele CSS-Attribute für den Keyframe.

Examples

Animationen mit der Übergangseigenschaft

Die CSS- `transition` nützlich für einfache Animationen. Sie ermöglicht die Animation von nummerischen CSS-Eigenschaften zwischen Zuständen.

Beispiel

```
.Example{
    height: 100px;
    background: #fff;
}

.Example:hover{
    height: 120px;
    background: #ff0000;
}
```

Zeige Ergebnis

Wenn Sie den `.Example` über ein Element mit der Klasse `.Example` , springt die `.Example` sofort auf `120px` und die Hintergrundfarbe auf Rot (`#ff0000`).

Durch das Hinzufügen der `transition` können diese Änderungen im Laufe der Zeit auftreten:

```
.Example{
    ...
    transition: all 400ms ease;
}
```

Zeige Ergebnis

Der Wert `all` wendet den Übergang auf alle kompatiblen (auf Zahlen basierenden) Eigenschaften an. Jeder kompatible Eigenschaftsname (z. B. `height` oder `top`) kann für dieses Schlüsselwort verwendet werden.

`400ms` gibt an, wie viel Zeit der Übergang benötigt. In diesem Fall dauert die Änderung der Höhe des Elements 400 Millisekunden.

Schließlich wird der Wert `ease` ist die Animationsfunktion, die bestimmt , wie die Animation abgespielt wird. `ease` bedeutet, langsam zu beginnen, zu beschleunigen und wieder langsam zu beenden. Andere Werte sind `linear` , `ease-out` und `ease-in` .

Browserübergreifende Kompatibilität

Die `transition` wird im Allgemeinen von allen großen Browsern mit Ausnahme von IE 9 gut unterstützt. Verwenden Sie für frühere Versionen von Firefox- und Webkit-basierten Browsern die Herstellerpräfixe wie folgt:

```
.Example{
    transition:         all 400ms ease;
    -moz-transition:    all 400ms ease;
    -webkit-transition: all 400ms ease;
}
```

Hinweis: Die `transition` kann Änderungen zwischen zwei beliebigen numerischen Werten unabhängig von der Einheit animieren. Es kann auch zwischen Einheiten `100px` , z. B. `100px` bis

`50vh` . Es kann jedoch nicht zwischen einer Zahl und einem Standardwert oder einem automatischen Wert `100px` werden, z. B. die Höhe eines Elements von `100px` auf `auto` .

Steigerung der Animationsleistung mithilfe des Attributs `will-change`

Beim Erstellen von Animationen und anderen GPU-schweren Aktionen ist es wichtig, das Attribut " `will-change` zu verstehen.

Beide CSS-Keyframes und die `transition` verwenden die GPU-Beschleunigung. Die Leistung wird erhöht, indem Berechnungen auf die GPU des Geräts geladen werden. Dazu erstellen Sie Malebenen (Teile der Seite, die einzeln gerendert werden), die zur Berechnung an die GPU ausgelagert werden. Die `will-change` Eigenschaft teilt dem Browser mit, was animiert wird, wodurch der Browser kleinere Farbbereiche erstellen kann, wodurch die Leistung erhöht wird.

Die `will-change` Eigenschaft akzeptiert eine durch Kommas getrennte Liste von Eigenschaften, die animiert werden sollen. Wenn Sie beispielsweise ein Objekt transformieren und seine Deckkraft ändern möchten, geben Sie Folgendes an:

```
.Example{
    ...
    will-change: transform, opacity;
}
```

Hinweis: Verwenden Sie nur `will-change` . Das Festlegen `will-change` für jedes Element auf einer Seite kann zu Leistungsproblemen führen, da der Browser möglicherweise versucht, Farbebenen für jedes Element zu erstellen, wodurch der Verarbeitungsaufwand der GPU erheblich erhöht wird.

Animationen mit Keyframes

Für mehrstufige CSS-Animationen können Sie CSS `@keyframes` erstellen. Mit Keyframes können Sie mehrere Animationspunkte, so genannte Keyframes, definieren, um komplexere Animationen zu definieren.

Basisbeispiel

In diesem Beispiel erstellen wir eine grundlegende Hintergrundanimation, die zwischen allen Farben wechselt.

```
@keyframes rainbow-background {
          0%     { background-color: #ff0000; }
     8.333%      { background-color: #ff8000; }
    16.667%      { background-color: #ffff00; }
    25.000%      { background-color: #80ff00; }
    33.333%      { background-color: #00ff00; }
    41.667%      { background-color: #00ff80; }
    50.000%      { background-color: #00ffff; }
    58.333%      { background-color: #0080ff; }
    66.667%      { background-color: #0000ff; }
    75.000%      { background-color: #8000ff; }
```

```
    83.333%     { background-color: #ff00ff; }
    91.667%     { background-color: #ff0080; }
    100.00%     { background-color: #ff0000; }
}

.RainbowBackground {
    animation: rainbow-background 5s infinite;
}
```

Zeige Ergebnis

Hier gibt es einige verschiedene Dinge zu beachten. Zuerst die eigentliche `@keyframes` Syntax.

```
@keyframes rainbow-background{
```

Damit wird der Name der Animation auf `rainbow-background` .

```
0%     { background-color: #ff0000; }
```

Dies ist die Definition für ein Keyframe innerhalb der Animation. Der erste Teil, in diesem Fall `0%` , definiert, wo sich der Keyframe während der Animation befindet. Die `0%` bedeuten, dass es 0% der gesamten Animationszeit von Anfang an ist.

Die Animation wechselt automatisch zwischen den Keyframes. `8.333%` die nächste Hintergrundfarbe auf `8.333%` , benötigt die Animation 8,333% der Zeit, um zwischen diesen Keyframes zu wechseln.

```
.RainbowBackground {
    animation: rainbow-background 5s infinite;
}
```

Mit diesem Code wird unsere Animation an alle Elemente `.RainbowBackground` die über die Klasse `.RainbowBackground` verfügen.

Die tatsächliche Animationseigenschaft akzeptiert die folgenden Argumente.

- **Animationsname** : Der Name unserer Animation. In diesem Fall `rainbow-background`
- **Animationsdauer** : Wie lange dauert die Animation, in diesem Fall 5 Sekunden.
- Anzahl der **Animations-Iterationen (optional)** : Gibt an, wie oft die Animation wiederholt wird. In diesem Fall wird die Animation unbegrenzt fortgesetzt. Standardmäßig wird die Animation einmal abgespielt.
- **Animationsverzögerung (optional)** : Gibt an, wie lange gewartet werden soll, bevor die Animation beginnt. Der Standardwert ist 0 Sekunden und kann negative Werte annehmen. Zum Beispiel würde `-2s` die Animation 2 Sekunden in ihre Schleife starten.
- **Animations-Timing-Funktion (optional)** : Bestimmt die Geschwindigkeitskurve der Animation. Die Standardeinstellung ist " `ease` , wenn die Animation langsam beginnt, schneller wird und langsam endet.

In diesem speziellen Beispiel geben sowohl die `0%` als auch die `100% { background-color: #ff0000; }` . Wenn zwei oder mehr Keyframes einen Zustand gemeinsam haben, können sie in einer einzigen

Anweisung angegeben werden. In diesem Fall könnten die beiden `0%` und `100%` Zeilen durch diese eine Zeile ersetzt werden:

```
0%, 100%     { background-color: #ff0000; }
```

Browserübergreifende Kompatibilität

Bei älteren WebKit-basierten Browsern müssen Sie das Herstellerpräfix sowohl für die `@keyframes` Deklaration als auch für die `animation`

```
@-webkit-keyframes{}

-webkit-animation: ...
```

Syntax-Beispiele

Unser erstes Syntaxbeispiel zeigt die Animationskürzel-Eigenschaft unter Verwendung aller verfügbaren Eigenschaften / Parameter:

```
  animation: 3s         ease-in           1s      2                 reverse     both
paused       slidein;
  /*         duration | timing-function | delay | iteration-count | direction | fill-mode |
play-state | name   */
```

Unser zweites Beispiel ist etwas einfacher und zeigt, dass einige Eigenschaften weggelassen werden können:

```
  animation: 3s         linear            1s      slidein;
  /*         duration | timing-function | delay | name   */
```

Unser drittes Beispiel zeigt die minimalste Deklaration. Beachten Sie, dass der Name der Animation und die Dauer der Animation angegeben werden müssen:

```
  animation: 3s         slidein;
  /*         duration | name */
```

Es ist auch erwähnenswert, dass bei der Verwendung der Animationskurzschrift die Reihenfolge der Eigenschaften einen Unterschied macht. Natürlich kann der Browser Ihre Dauer mit Ihrer Verzögerung verwechseln.

Wenn Kürze nicht Ihr Ding ist, können Sie auch die Abkürzungseigenschaft überspringen und jede Eigenschaft einzeln schreiben:

```
animation-duration: 3s;
animation-timing-function: ease-in;
animation-delay: 1s;
animation-iteration-count: 2;
```

```
animation-direction: reverse;
animation-fill-mode: both;
animation-play-state: paused;
animation-name: slidein;
```

Animationen online lesen: https://riptutorial.com/de/css/topic/590/animationen

Kapitel 5: Bemerkungen

Syntax

- /* Kommentar */

Bemerkungen

- Kommentare in CSS beginnen immer mit `/*` und enden mit `*/`
- Kommentare können nicht verschachtelt werden

Examples

Einzelne Zeile

```
/* This is a CSS comment */
div {
   color: red; /* This is a CSS comment */
}
```

Mehrfachzeile

```
/*
    This
    is
    a
    CSS
    comment
*/
div {
   color: red;
}
```

Bemerkungen online lesen: https://riptutorial.com/de/css/topic/1625/bemerkungen

Kapitel 6: Benutzerdefinierte Eigenschaften (Variablen)

Einführung

Mit CSS-Variablen können Autoren wiederverwendbare Werte erstellen, die in einem CSS-Dokument verwendet werden können.

In CSS ist es zum Beispiel üblich, eine einzige Farbe im gesamten Dokument wiederzuverwenden. Vor CSS-Variablen würde dies bedeuten, dass derselbe Farbwert in einem Dokument mehrmals verwendet wird. Mit CSS-Variablen kann der Farbwert einer Variablen zugewiesen und an mehreren Stellen referenziert werden. Dies macht das Ändern von Werten einfacher und ist semantischer als die Verwendung herkömmlicher CSS-Werte.

Syntax

- : root {} / * Pseudoklasse, die eine umfassendere Definition von Variablen ermöglicht * /
- - Variablenname: *Wert* ; / * Variable definieren * /
- var (- Variablenname, *Standardwert*) / * definierte Variable mit Standardwert-Fallback verwenden * /

Bemerkungen

CSS-Variablen gelten derzeit als experimentelle Technologie.

BROWSER SUPPORT / KOMPATIBILITÄT

Firefox: Version **31** + *(standardmäßig aktiviert)*

Mehr Infos von Mozilla

Chrome: Version **49** + *(standardmäßig aktiviert)* .

"Diese Funktion kann in Chrome Version 48 zum Testen aktiviert werden, indem die `experimental Web Platform` *Funktion aktiviert wird. Geben Sie* `chrome://flags/` *in Ihre Chrome-Adressleiste ein, um auf diese Einstellung zuzugreifen."*

IE: Nicht unterstützt.

Edge: In Entwicklung

Safari: Version **9.1 und höher**

Examples

Variable Farbe

```
:root {
  --red: #b00;
  --blue: #4679bd;
  --grey: #ddd;
}
.Bx1 {
  color: var(--red);
  background: var(--grey);
  border: 1px solid var(--red);
}
```

Variable Abmessungen

```
:root {
  --W200: 200px;
  --W10: 10px;
}
.Bx2 {
  width: var(--W200);
  height: var(--W200);
  margin: var(--W10);
}
```

Variable Kaskadierung

CSS-Variablen kaskadieren auf ähnliche Weise wie andere Eigenschaften und können sicher wiederhergestellt werden.

Sie können Variablen mehrmals definieren, und nur die Definition mit der höchsten Spezifität gilt für das ausgewählte Element.

Annahme dieses HTML:

```
<a class="button">Button Green</a>
<a class="button button_red">Button Red</a>
<a class="button">Button Hovered On</a>
```

Wir können dieses CSS schreiben:

```
.button {
  --color: green;
  padding: .5rem;
  border: 1px solid var(--color);
  color: var(--color);
}

.button:hover {
  --color: blue;
```

```
}

.button_red {
  --color: red;
}
```

Und bekomme dieses Ergebnis:

Gültig / Invaliden

Benennung Wenn Sie CSS-Variablen benennen, enthält sie nur Buchstaben und Bindestriche wie andere CSS-Eigenschaften (z. B. Zeilenhöhe, -moz-box-size), sollte jedoch mit doppelten Bindestrichen (-) beginnen

```
//These are Invalids variable names
--123color: blue;
--#color: red;
--bg_color: yellow
--$width: 100px;

//Valid variable names
--color: red;
--bg-color: yellow
--width: 100px;
```

CSS-Variablen unterscheiden zwischen Groß- und Kleinschreibung.

```
/* The variable names below are all different variables */
--pcolor: ;
--Pcolor: ;
--pColor: ;
```

Leer gegen Leerzeichen

```
/* Invalid */
    --color:;

/* Valid */
  --color: ; /* space is assigned */
```

Verkettungen

```
    /* Invalid - CSS doesn't support concatenation*/
    .logo{
        --logo-url: 'logo';
        background: url('assets/img/' var(--logo-url) '.png');
    }

    /* Invalid - CSS bug */
```

```
    .logo{
        --logo-url: 'assets/img/logo.png';
        background: url(var(--logo-url));
    }

    /* Valid */
    .logo{
        --logo-url: url('assets/img/logo.png');
        background: var(--logo-url);
    }
```

Vorsicht bei der Verwendung von Einheiten

```
    /* Invalid */
    --width: 10;
    width: var(--width)px;

    /* Valid */
    --width: 10px;
    width: var(--width);

    /* Valid */
    --width: 10;
    width: calc(1px * var(--width)); /* multiply by 1 unit to convert */
    width: calc(1em * var(--width));
```

Mit Medienanfragen

Sie können Variablen innerhalb von Medienabfragen neu setzen und diese neuen Werte an jedem Ort kaskadieren lassen, was bei Vorprozessorvariablen nicht möglich ist.

Hier ändert eine Medienabfrage die Variablen, die zum Einrichten eines sehr einfachen Gitters verwendet werden:

HTML

```
<div></div>
<div></div>
<div></div>
<div></div>
```

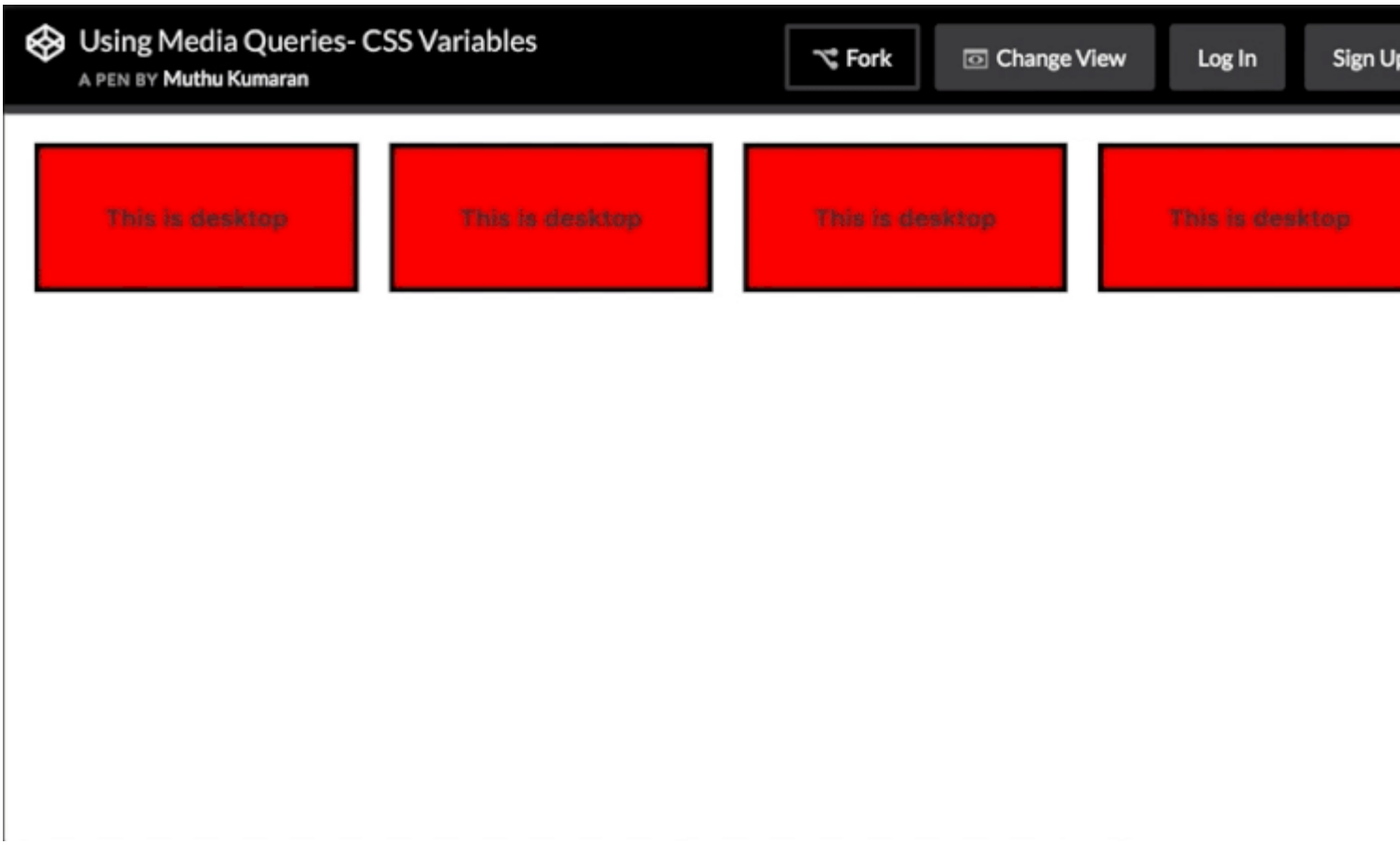
CSS

```
:root{
    --width: 25%;
    --content: 'This is desktop';
}
@media only screen and (max-width: 767px){
    :root{
        --width:50%;
        --content: 'This is mobile';
    }
}
@media only screen and (max-width: 480px){
    :root{
```

```
        --width:100%;
    }
}

div{
    width: calc(var(--width) - 20px);
    height: 100px;
}
div:before{
    content: var(--content);
}

/* Other Styles */
body {
    padding: 10px;
}

div{
    display: flex;
    align-items: center;
    justify-content: center;
    font-weight:bold;
    float:left;
    margin: 10px;
    border: 4px solid black;
    background: red;
}
```

Sie können versuchen, die Größe des Fensters in dieser CodePen-Demo zu ändern

Hier ist ein animierter Screenshot der Größenänderung in Aktion:



Benutzerdefinierte Eigenschaften (Variablen) online lesen:
https://riptutorial.com/de/css/topic/1755/benutzerdefinierte-eigenschaften--variablen-

Kapitel 7: Beschneiden und Maskieren

Syntax

- **Ausschnitt**
- Clip-Pfad: <Clip-Quelle> | [<Grundform> || <Clip-Geometrie-Box>] | keiner
- **Maskieren**
- Maskenbild: [keine | <mask-reference>] #
- Maskenmodus: [<Maskenmodus>] #
- Maskenwiederholung: [<Wiederholungsstil] #
- Maskenposition: [<Position>] #
- Masken-Clip: [<Geometrie-Box> | kein Clip]#
- Maskenursprung: [<Geometrie-Box>] #
- Maskengröße: [<bg-size>] #
- Maskenverbund: [<Compositing-Operator>] #
- Maske: [<Maskenreferenz> <Maskierungsmodus>? || <position> [/ <bg-size>]? || <repeat-style> || <Geometrie-Box> || [<Geometrie-Box> | kein Clip] || <compositing-operator>] #

Parameter

Parameter	Einzelheiten
Clip-Quelle	Eine URL, die auf ein Inline-SVG-Element (oder) ein SVG-Element in einer externen Datei verweisen kann, das die Definition des Clippfads enthält.
Grundform	Bezieht sich auf eins zwischen `inset()` , `circle()` , `ellipse()` oder `polygon()` . Mit einer dieser Funktionen wird der Beschneidungspfad definiert. Diese Formfunktionen funktionieren genauso wie in Shapes für Floats
Clip-Geometrie-Box	Dies kann unter `content-box` , `padding-box` , `border-box` , `margin-box` , `fill-box` , `stroke-box` , `view-box` als Werte stehen. Wenn dieser Wert ohne Wert für <Grundform> angegeben wird, werden die Kanten der entsprechenden Box als Pfad für das Beschneiden verwendet. Bei Verwendung mit einer <Grundform> fungiert diese als Referenzfeld für die Form.
Maskenreferenz	Dies kann `none` oder ein Bild oder eine Referenz-URL für eine Maskenbildquelle sein.
Wiederholungsstil	Dies gibt an, wie die Maske in der X- und Y-Achse wiederholt oder gekachelt werden soll. Die unterstützten Werte sind " `repeat-x` , " `repeat-y` , " `repeat` , " `space` , " `round` , " `no-repeat` .

Parameter	Einzelheiten
Maskenmodus	Kann `alpha` oder `luminance` oder `auto` und gibt an, ob die Maske als Alphamaske oder Luminanzmaske behandelt werden soll. Wenn kein Wert angegeben wird und die Maskenreferenz ein direktes Bild ist, wird sie als Alphamaske betrachtet (oder). Wenn die Maskenreferenz eine URL ist, wird sie als Luminanzmaske betrachtet.
Position	Dies gibt die Position der einzelnen Maskenebenen an und ähnelt der Eigenschaft `background-position` . Der Wert kann in einer 1-Wert-Syntax (wie `top` , `10%`) oder in einer 2-Wert-Syntax (wie `top right` , `50% 50%`) angegeben werden.
Geometrie-Box	Dies gibt die Box an, auf die die Maske zugeschnitten werden soll (*Maskenbereich*), oder die Box, die als Referenz für den Ursprung der Maske (*Maskenpositionierungsbereich*) verwendet werden soll, abhängig von der Eigenschaft. Die Liste der möglichen Werte ist `content-box` , `padding-box` , `border-box` , `margin-box` , `fill-box` , `stroke-box` , `view-box` . Ausführliche Informationen zur Funktionsweise dieser Werte finden Sie in der W3C-Spezifikation .
BG-Größe	Dies stellt die Größe jeder Maskenbildebene dar und hat dieselbe Syntax wie die `background-size` . Der Wert kann Länge oder Prozentsatz oder Auto oder Cover oder enthalten. Länge, Prozentsatz und Auto können entweder als einzelner Wert oder als einzelner Wert für jede Achse angegeben werden.
Compositing-Operator	Dies kann einer der folgenden Werte sein: `add` , `subtract` , `exclude` , `multiply` pro Layer und definiert die Art des Compositing-Vorgangs, der für diesen Layer verwendet werden soll, mit denen darunter. Ausführliche Erläuterungen zu jedem Wert finden Sie in den W3C-Spezifikationen .

Bemerkungen

CSS-Clipping und -Maskierung sind sehr neue Konzepte und daher ist die Browserunterstützung für diese Eigenschaften relativ gering.

Masken:

Wie zum Zeitpunkt des Schreibens (Juli '16) unterstützen Chrome, Safari und Opera diese Eigenschaften mit dem Präfix `-webkit-` .

Firefox erfordert keine Präfixe, unterstützt jedoch nur Masken, wenn SVG- `mask` verwendet werden. Für Inline SVG `mask` die Syntax - `mask: url(#msk)` , während für die Verwendung von `mask` in einer externen SVG - Datei ist die Syntax `mask: url('yourfilepath/yourfilename.svg#msk')` . `#msk`

in beiden Fällen bezieht sich auf die `id` des `mask` auf die verwiesen wird. Wie in dieser Antwort angegeben , unterstützt Firefox derzeit keine anderen Parameter als `mask-reference` in der `mask` Eigenschaft.

Internet Explorer (und Edge) bietet noch keine Unterstützung für diese Eigenschaft.

Die `mask-mode` Eigenschaft wird derzeit von keinem Browser **mit oder ohne** Präfixe unterstützt.

Clip-Pfad:

Zum Zeitpunkt des Schreibens (Juli '16) unterstützt Chrome, Safari und Opera den `clip-path` wenn der Pfad mit grundlegenden Formen (wie `circle` , `polygon`) oder der `url(#clipper)` mit Inline-SVG erstellt wird. Sie unterstützen kein Beschneiden basierend auf Formen, die Bestandteil externer SVG-Dateien sind. Sie benötigen außerdem das Präfix `-webkit` .

Firefox unterstützt nur die `url()` Syntax für den `clip-path` während Internet Explorer (und Edge) keine Unterstützung bieten.

Examples

Ausschnitt (Polygon)

CSS:

```
div{
  width:200px;
  height:200px;
  background:teal;
  clip-path: polygon(0 0, 0 100%, 100% 50%); /* refer remarks before usage */
}
```

HTML:

```
<div></div>
```

Im obigen Beispiel wird ein **polygonaler** Beschneidungspfad verwendet, um das quadratische Element (200 x 200) in eine Dreiecksform zu schneiden. Die Ausgangsform ein Dreieck ist, weil der Pfad an beginnt (das heißt, erster Koordinaten at) `0 0` - das ist die linke obere Ecke der Box, geht dann zu `0 100%` - die Ecke unten links des Kastens und dann schließlich zu `100% 50%` was nichts anderes als der rechte Mittelpunkt der Box ist. Diese Pfade schließen sich selbst (dh der Startpunkt ist der Endpunkt), und die endgültige Form ist die eines Dreiecks.

Dies kann auch für ein Element mit einem Bild oder einem Farbverlauf als Hintergrund verwendet werden.

Beispiel ansehen

Ausgabe:

Ausschnitt (Kreis)

CSS:

```
div{
  width: 200px;
  height: 200px;
  background: teal;
  clip-path: circle(30% at 50% 50%); /* refer remarks before usage */
}
```

HTML

```
<div></div>
```

Dieses Beispiel zeigt, wie Sie ein div an einen Kreis schneiden. Das Element wird in einen Kreis geschnitten, dessen Radius 30% beträgt, basierend auf den Abmessungen des Bezugsrahmens, dessen Mittelpunkt in der Mitte des Bezugsrahmens liegt. Da hier kein <Clip-Geometrie-Feld> (also Referenzfeld) bereitgestellt wird, wird das `border-box` des Elements als Referenzfeld verwendet.

Die Kreisform muss einen Radius und einen Mittelpunkt mit `(x,y)` -Koordinaten haben:

```
circle(radius at x y)
```

Beispiel ansehen

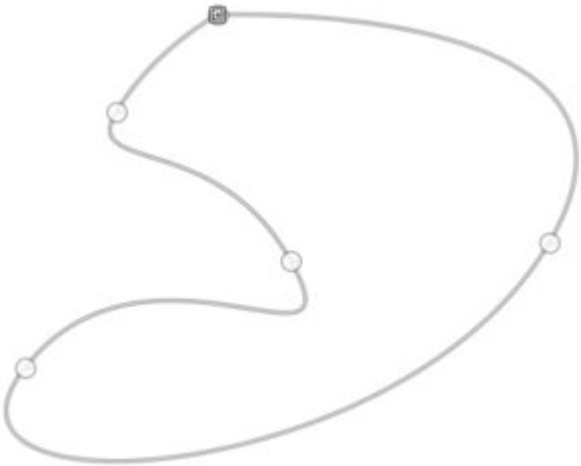
Ausgabe:

Beschneiden und Maskieren: Überblick und Unterschied

Mit dem **Ausschneiden** und **Maskieren können** Sie bestimmte Teile von Elementen transparent oder undurchsichtig machen. Beide können auf jedes HTML-Element angewendet werden.

Ausschnitt

Clips sind Vektorpfade. Außerhalb dieses Pfads ist das Element transparent, innerhalb undurchsichtig. Daher können Sie eine `clip-path` Eigenschaft für Elemente definieren. Jedes grafische Element, das auch in SVG vorhanden ist, kann hier als Funktion zur Definition des Pfads verwendet werden. Beispiele sind `circle()` , `polygon()` oder `ellipse()` .



Beispiel

```
clip-path: circle(100px at center);
```

Das Element ist nur innerhalb dieses Kreises sichtbar, der sich in der Mitte des Elements befindet und einen Radius von 100px hat.

Maskieren

Masken ähneln Clips, aber anstatt einen Pfad zu definieren, definieren Sie eine Maske, die sich

über dem Element befindet. Sie können sich diese Maske als Bild vorstellen, das hauptsächlich aus zwei Farben besteht: Schwarz und Weiß.

Luminanzmaske : Schwarz bedeutet, dass die Region undurchsichtig ist und weiß, dass sie transparent ist. Es gibt jedoch auch einen grauen Bereich, der halbtransparent ist, sodass Sie glatte Übergänge machen können.

Alpha-Maske : Nur in den transparenten Bereichen der Maske ist das Element undurchsichtig.

Dieses Bild kann beispielsweise als Luminanzmaske verwendet werden, um einem Element einen sehr sanften Übergang von rechts nach links und von undurchsichtig nach transparent zu ermöglichen.

Mit der `mask` Eigenschaft können Sie den Maskentyp und ein Bild angeben, das als Ebene verwendet werden soll.

Beispiel

```
mask: url(masks.svg#rectangle) luminance;
```

Ein in `masks.svg` definiertes Element, das als `rectangle` bezeichnet wird, wird als **Luminanzmaske** für das Element verwendet.

Einfache Maske, die ein Bild von Vollton zu Transparent überblendet

CSS

```
div {
  height: 200px;
  width: 200px;
  background: url(http://lorempixel.com/200/200/nature/1);
  mask-image: linear-gradient(to right, white, transparent);
}
```

HTML

```
<div></div>
```

Im obigen Beispiel gibt es ein Element mit einem Bild als Hintergrund. Die auf das Bild angewendete Maske (unter Verwendung von CSS) lässt das Bild so aussehen, als würde es von links nach rechts ausgeblendet.

Die Maskierung wird erreicht, indem ein `linear-gradient` , der von Weiß (links) bis Transparent (rechts) als Maske geht. Da es sich um eine Alphamaske handelt, wird das Bild transparent, wenn die Maske transparent ist.

Ausgabe ohne Maske:

Ausgabe mit der Maske:

Hinweis: Wie in den Anmerkungen erwähnt, funktioniert das obige Beispiel nur in Chrome, Safari und Opera, wenn es mit dem Präfix `-webkit` . Dieses Beispiel (mit einem `linear-gradient` als Maskenbild) wird in Firefox noch nicht unterstützt.

Verwenden Sie Masken, um ein Loch in die Mitte eines Bildes zu schneiden

CSS

```
div {
  width: 200px;
  height: 200px;
```

```
  background: url(http://lorempixel.com/200/200/abstract/6);
  mask-image: radial-gradient(circle farthest-side at center, transparent 49%, white 50%); /*
check remarks before using */
}
```

HTML

Im obigen Beispiel wird ein transparenter Kreis in der Mitte mit einem `radial-gradient` Dieser wird dann als Maske verwendet, um die Wirkung eines Kreises zu erzeugen, der aus der Mitte eines Bildes ausgeschnitten wird.

Bild ohne Maske:

Bild mit Maske:

Verwenden von Masken zum Erstellen von Bildern mit unregelmäßigen Formen

CSS

```
div { /* check remarks before usage */
  height: 200px;
  width: 400px;
  background-image: url(http://lorempixel.com/400/200/nature/4);
```

```
  mask-image: linear-gradient(to top right, transparent 49.5%, white 50.5%), linear-
gradient(to top left, transparent 49.5%, white 50.5%), linear-gradient(white, white);
  mask-size: 75% 25%, 25% 25%, 100% 75%;
  mask-position: bottom left, bottom right, top left;
  mask-repeat: no-repeat;
}
```

HTML

```
<div></div>
```

Im obigen Beispiel werden drei Bilder mit `linear-gradient` (die, wenn sie an den entsprechenden Positionen platziert werden, 100% x 100% der Behältergröße abdecken) als Masken verwendet, um einen transparenten, dreieckigen Schnitt am unteren Rand des Bildes zu erzeugen.

Bild ohne Maske:

Bild mit der Maske:

Beschneiden und Maskieren online lesen: https://riptutorial.com/de/css/topic/3721/beschneiden-und-maskieren

Kapitel 8: Block-Formatierungskontexte

Bemerkungen

> [Ein Blockformatierungskontext ist Teil eines visuellen CSS-Rendering einer Webseite. Dies ist der Bereich, in dem das Layout von Blockboxen auftritt und in dem Floats miteinander interagieren.] [1]

[1]: https://developer.mozilla.org/en-US/docs/Web/Guide/CSS/Block_formatting_context MDN

Examples

Verwenden der Überlaufeigenschaft mit einem anderen Wert als sichtbar

```
img{
  float:left;
  width:100px;
  margin:0 10px;
}
.div1{
  background:#f1f1f1;
  /* does not create block formatting context */
}
.div2{
  background:#f1f1f1;
  overflow:hidden;
  /* creates block formatting context */
}
```

ate Fork Set as base Tidy Collaborate Embed

```html
<img src="https://static.pexels.com/photos/33695/matryoshka-red-fly-agaric-mushroom-mushrooms-forest-medium.jpg" alt="" />
<div class=div1>
<p>Lorem ipsum dolor sit amet, cum no paulo mollis pertinacia. Eam in velit graecis, sea mucius insolens ne. Amet doming at has, omnis errem an cum. Eu vim appareat persecuti, ea putant definitionem has, vis ea legendos expetenda. No eros graeci minimum nam, justo augue instructior usu ne. At ludus suscipit disputationi vel.</p>

<p>Ad case omnis nam, mutat deseruisse persequeris eos ad, in tollit debitis sea. Cu eos munere virtute vituperata. Exerci bonorum sed id, id nec tantas praesent complectitur. Vel cu legendos mediocritatem. Enim liberavisse ei sea.</p>
</div>

<img src="https://static.pexels.com/photos/129574/pexels-photo-129574-medium.jpeg" alt="" />
<div class=div2>
<p>Lorem ipsum dolor sit amet, cum no paulo mollis pertinacia. Eam in velit graecis, sea mucius insolens ne. Amet doming at has, omnis errem an cum. Eu vim appareat persecuti, ea putant definitionem has, vis ea legendos expetenda. No eros graeci minimum nam, justo augue instructior usu ne. At ludus suscipit disputationi vel.</p>

<p>Ad case omnis nam, mutat deseruisse persequeris eos ad, in tollit debitis sea. Cu eos munere virtute vituperata. Exerci bonorum sed id, id nec tantas praesent complectitur. Vel cu legendos mediocritatem. Enim liberavisse ei sea.</p>
</div>
```

HTML

```css
img{
  float:left;
  width:100px;
  margin:0 10px;
}
.div1{
  background:#f1f1f1;
}
.div2{
  background:#f1f1f1;
  overflow:hidden;
  /* creates block formatting c
}
```

Lorem ipsum dolor
doming at has, omni
expetenda. No eros

Ad case omnis nam, mutat deseruisse
sed id, id nec tantas praesent complecti

Lorem ipsum dolor
doming at has, omni
expetenda. No eros

Ad case omnis nam,
Exerci bonorum sed
sea.

https://jsfiddle.net/MadalinaTn/qkwwmu6m/2/

> Wenn Sie die Überlaufeigenschaft mit einem anderen Wert als sichtbar (Standardwert) verwenden, wird ein neuer Blockformatierungskontext erstellt. Dies ist technisch notwendig - wenn ein Float mit dem Bildlaufelement geschnitten wird, wird der Inhalt zwangsweise neu geschrieben.

Dieses Beispiel, das zeigt, wie mehrere Absätze mit einem schwebenden Bild interagieren, ähnelt diesem Beispiel auf css-tricks.com.

2 : https://developer.mozilla.org/en-US/docs/Web/CSS/overflow MDN

Block-Formatierungskontexte online lesen: https://riptutorial.com/de/css/topic/5069/block-formatierungskontexte

Kapitel 9: Box Schatten

Syntax

- Box-Schatten: keine | h-Schatten-V-Schatten-Unschärfefarbe |

Parameter

Parameter	Einzelheiten
Einsatz	Standardmäßig wird der Schatten als Schlagschatten behandelt. Das Schlüsselwort "Inset" zeichnet den Schatten innerhalb des Rahmens / Rahmens.
Versatz-x	die horizontale Entfernung
Versatz-y	der vertikale Abstand
Unschärfe-Radius	0 standardmäßig Wert kann nicht negativ sein. Je größer der Wert, desto größer und heller wird der Schatten.
Ausbreitungsradius	0 standardmäßig Positive Werte bewirken, dass sich der Schatten ausdehnt. Bei negativen Werten schrumpft der Schatten.
Farbe	kann aus verschiedenen Notationen bestehen: ein Farbschlüsselwort, hexadezimal, `rgb()` , `rgba()` , `hsl()` , `hsla()`

Bemerkungen

Browser-Unterstützung:

- Chrome 10.0
- IE 9.0
- Firefox 4.0 3,5 moz
- Safari 5.1 3.1 -webkit-
- Opera 10.5

Examples

Schlagschatten

JSFiddle: https://jsfiddle.net/UnsungHero97/80qod7aL/

HTML

```
<div class="box_shadow"></div>
```

CSS

```
.box_shadow {
  -webkit-box-shadow: 0px 0px 10px -1px #444444;
     -moz-box-shadow: 0px 0px 10px -1px #444444;
          box-shadow: 0px 0px 10px -1px #444444;
}
```

innerer Schlagschatten

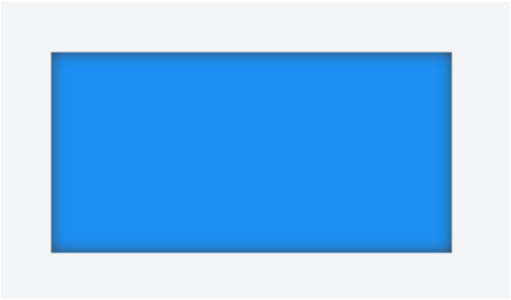
HTML

```
<div class="box_shadow"></div>
```

CSS

```
.box_shadow {
  background-color: #1C90F3;
  width: 200px;
  height: 100px;
  margin: 50px;
  -webkit-box-shadow: inset 0px 0px 10px 0px #444444;
     -moz-box-shadow: inset 0px 0px 10px 0px #444444;
          box-shadow: inset 0px 0px 10px 0px #444444;
}
```

Ergebnis:



JSFiddle: https://jsfiddle.net/UnsungHero97/80qod7aL/1/

Nur-Bottom-Drop-Schatten mit einem Pseudoelement

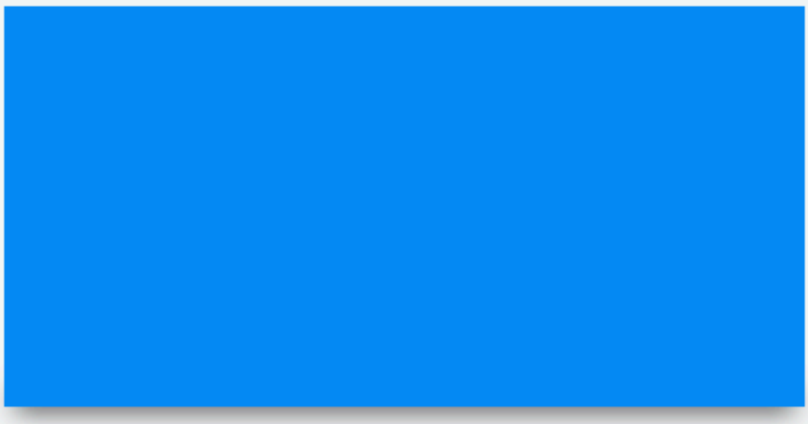
JSFiddle: https://jsfiddle.net/UnsungHero97/80qod7aL/2/

HTML

```
<div class="box_shadow"></div>
```

CSS

```
.box_shadow {
  background-color: #1C90F3;
  width: 200px;
  height: 100px;
  margin: 50px;
}

.box_shadow:after {
  content: "";
  width: 190px;
  height: 1px;
  margin-top: 98px;
  margin-left: 5px;
  display: block;
  position: absolute;
  z-index: -1;
  -webkit-box-shadow: 0px 0px 8px 2px #444444;
     -moz-box-shadow: 0px 0px 8px 2px #444444;
          box-shadow: 0px 0px 8px 2px #444444;
}
```



mehrere Schatten

JSFiddle: https://jsfiddle.net/UnsungHero97/80qod7aL/5/

HTML

```
<div class="box_shadow"></div>
```

CSS

```
.box_shadow {
  width: 100px;
  height: 100px;
```

```
  margin: 100px;
  box-shadow:
    -52px -52px 0px 0px #f65314,
    52px -52px 0px 0px #7cbb00,
    -52px 52px 0px 0px #00a1f1,
    52px 52px 0px 0px #ffbb00;
}
```

Box Schatten online lesen: https://riptutorial.com/de/css/topic/1746/box-schatten

Kapitel 10: Browserstile normalisieren

Einführung

Jeder Browser verfügt über einen Standardsatz von CSS-Stilen, die zum Rendern von Elementen verwendet werden. Diese Standardstile sind möglicherweise in allen Browsern nicht konsistent, weil die Sprachspezifikationen unklar sind. Daher können Basisstile interpretiert werden, Browser entsprechen möglicherweise nicht den angegebenen Spezifikationen oder Browser verfügen nicht über Standardstile für neuere HTML-Elemente. Daher möchten Benutzer möglicherweise Standardstile für so viele Browser wie möglich normalisieren.

Bemerkungen

Meyer's Reset führt, obwohl es effektiv ist, die gleichen Änderungen an fast jedem häufig verwendeten Element durch. Dies hat zur Folge, dass die Fenster des Web-Browser-Inspektors immer wieder mit den gleichen angewendeten Stilen verschmutzt werden. Dies führt zu mehr Arbeit für den Browser (mehr Regeln für mehr Elemente). Auf der anderen Seite ist die Normalize-Technik viel fokussierter und weniger breitstrahlend. Dies vereinfacht die Arbeit des Browsers und führt zu weniger Unordnung in den Browserprüfwerkzeugen.

Examples

normalize.css

Browser verfügen über einen Standardsatz von CSS-Stilen, die sie zum Rendern von Elementen verwenden. Einige dieser Stile können sogar mithilfe der Browsereinstellungen angepasst werden, um z. B. die Standardeinstellungen für Schriftart und -größe zu ändern. Die Stile enthalten die Definition, welche Elemente unter anderem Blockebene oder Inline sein sollen.

Da diese Standardstile aufgrund der Sprachspezifikationen einen gewissen Spielraum haben, und da Browser den Spezifikationen möglicherweise nicht ordnungsgemäß folgen, können sie sich von Browser zu Browser unterscheiden.

Hier kommt normalize.css ins Spiel. Sie überschreibt die häufigsten Inkonsistenzen und behebt bekannte Fehler.

Was tut es

- Im Gegensatz zu vielen CSS-Resets werden nützliche Standardwerte beibehalten.
- Normalisiert Stile für eine Vielzahl von Elementen.
- Behebt Fehler und häufig auftretende Browserinkonsistenzen.
- Verbessert die Benutzerfreundlichkeit mit subtilen Modifikationen.
- Erläutert, was der Code mit ausführlichen Kommentaren macht.

Indem Sie normalize.css in Ihr Projekt einbinden, wird Ihr Design in den verschiedenen Browsern ähnlicher und konsistenter aussehen.

Unterschied zu reset.css

Sie haben vielleicht von `reset.css` . Was ist der Unterschied zwischen den beiden?

Während normalize.css für Konsistenz sorgt, indem verschiedene Eigenschaften auf einheitliche Standardwerte festgelegt werden, erreicht reset.css Konsistenz, **indem** alle grundlegenden Formatierungen **entfernt werden** , die ein Browser möglicherweise anwendet. Auch wenn dies zunächst nach einer guten Idee klingt, bedeutet dies tatsächlich, dass Sie **alle** Regeln selbst schreiben müssen, was sich jedoch gegen einen soliden Standard richtet.

Ansätze und Beispiele

CSS-Resets verwenden separate Ansätze für die Browser-Standardeinstellungen. Das Reset-CSS von Eric Meyer ist schon eine Weile im Gange. Sein Ansatz macht viele der Browserelemente zunichte, von denen bekannt ist, dass sie auf der Rückseite Probleme verursachen. Das Folgende ist von seiner Version (v2.0 | 20110126) CSS Reset.

```
html, body, div, span, applet, object, iframe,
h1, h2, h3, h4, h5, h6, p, blockquote, pre,
a, abbr, acronym, address, big, cite, code,
del, dfn, em, img, ins, kbd, q, s, samp,
small, strike, strong, sub, sup, tt, var,
b, u, i, center,
dl, dt, dd, ol, ul, li,
fieldset, form, label, legend,
table, caption, tbody, tfoot, thead, tr, th, td,
article, aside, canvas, details, embed,
figure, figcaption, footer, header, hgroup,
menu, nav, output, ruby, section, summary,
time, mark, audio, video {
    margin: 0;
    padding: 0;
    border: 0;
    font-size: 100%;
    font: inherit;
    vertical-align: baseline;
}
```

Eric Meyers Reset-CSS

Normalisieren Sie CSS auf der anderen Seite und behandeln Sie viele davon getrennt. Das folgende Beispiel zeigt die Version (v4.2.0) des Codes.

```
/**
 * 1. Change the default font family in all browsers (opinionated).
 * 2. Correct the line height in all browsers.
 * 3. Prevent adjustments of font size after orientation changes in IE and iOS.
 */

/* Document
```

```
   ========================================================================== */

html {
  font-family: sans-serif; /* 1 */
  line-height: 1.15; /* 2 */
  -ms-text-size-adjust: 100%; /* 3 */
  -webkit-text-size-adjust: 100%; /* 3 */
}

/* Sections
   ========================================================================== */

/**
 * Remove the margin in all browsers (opinionated).
 */

body {
  margin: 0;
}

/**
 * Add the correct display in IE 9-.
 */

article,
aside,
footer,
header,
nav,
section {
  display: block;
}

/**
 * Correct the font size and margin on `h1` elements within `section` and
 * `article` contexts in Chrome, Firefox, and Safari.
 */

h1 {
  font-size: 2em;
  margin: 0.67em 0;
}
```

CSS normalisieren

Browserstile normalisieren online lesen: https://riptutorial.com/de/css/topic/1211/browserstile-normalisieren

Kapitel 11: Browserunterstützung und Präfixe

Parameter

Präfix	Browser (s)
`-webkit-`	Google Chrome, Safari, neuere Versionen von Opera 12 und höher, Android, Blackberry und UC-Browser
`-moz-`	Mozilla Firefox
`-ms-`	Internet Explorer, Edge
`-o-` , `-xv-`	Opera bis Version 12
`-khtml-`	Konquerer

Bemerkungen

Herstellerpräfixe ermöglichen die Unterstützung der Vorschau für neue CSS-Funktionen, deren Funktionalität von der Spezifikation noch nicht empfohlen wird.

> **Es wird empfohlen, in Produktionsumgebungen keine Herstellerpräfixe zu verwenden.** Diese Präfixe sind vorhanden, um neue Funktionen zu testen, die noch nicht abgeschlossen sind, und das Verhalten ist von Natur aus unerwartet. Einfach Präfixe für alte Browser **nicht** gewähren Browser - Unterstützung , wie Sie die Funktion nicht garantieren kann nicht geändert im Laufe der Zeit unterschiedlich auszuführen, und es immer noch in den alten Browsern gebrochen werden *konnte* , Sie zu unterstützen behaupten.
>
> Wenn die Unterstützung älterer Browser wichtig ist, sollten Sie stattdessen JavaScript oder andere Lösungen verwenden, um die Auswirkungen zu imitieren und die Unterstützung für alte Browser wirklich zu garantieren.

Browser verwenden ihre Präfixe und ignorieren die Eigenschaften, die sie nicht verstehen.

HINWEIS : Präfixe sollten immer vor der offiziellen, uneingeschränkten Syntax stehen. Andernfalls würden sie mit den vorangestellten Eigenschaften überschrieben, was am Ende eine andere Implementierung sein kann.

Wenn ein Browser eine unpräfixierte und eine vorangestellte Version einer Eigenschaft unterstützt, hat die zuletzt zu deklarierende Eigenschaft Vorrang.

Examples

Übergänge

```
div {
    -webkit-transition: all 4s ease;
       -moz-transition: all 4s ease;
         -o-transition: all 4s ease;
            transition: all 4s ease;
}
```

Verwandeln

```
div {
    -webkit-transform: rotate(45deg);
       -moz-transform: rotate(45deg);
        -ms-transform: rotate(45deg);
         -o-transform: rotate(45deg);
            transform: rotate(45deg);
}
```

Browserunterstützung und Präfixe online lesen:
https://riptutorial.com/de/css/topic/1138/browserunterstutzung-und-prafixe

Kapitel 12: CSS Image Sprites

Syntax

- **// Hintergrundposition verwenden**
  Hintergrund: URL ("Sprite-Image.png");
  Hintergrundposition: -20px 50px;
- **// Abkürzung für Hintergrundeigenschaften**
  Hintergrund: URL ("sprite-image.png") -20px 50px;

Bemerkungen

In einigen Anwendungsfällen geraten Sprites langsam in Ungnade und werden durch Symbol-Webfonts oder SVG-Bilder ersetzt .

Examples

Eine grundlegende Implementierung

Was ist ein Image-Sprite?

Ein Image-Sprite ist ein einzelnes Asset, das sich in einem Image-Sprite-Blatt befindet. Ein Image-Sprite-Sheet ist eine Image-Datei, die mehrere Assets enthält, die daraus extrahiert werden können.

Zum Beispiel:

Das Bild oben ist ein Bildsprite-Blatt, und jeder dieser Sterne ist ein Sprite innerhalb des Sprite-Blattes. Diese Sprite-Sheets sind hilfreich, da sie die Leistung verbessern, indem sie die Anzahl der HTTP-Anforderungen reduzieren, die ein Browser möglicherweise stellen muss.

Wie implementierst du eine? Hier ist ein Beispielcode.

HTML

```
<div class="icon icon1"></div>
```

```
<div class="icon icon2"></div>
<div class="icon icon3"></div>
```

CSS

```
.icon {
    background: url(“icons-sprite.png”);
    display: inline-block;
    height: 20px;
    width: 20px;
}
.icon1 {
      background-position: 0px 0px;
}
.icon2 {
      background-position: -20px 0px;
}
.icon3 {
      background-position: -40px 0px;
}
```

Durch Festlegen der Breite und Höhe des Sprites und der Eigenschaft background-position in CSS (mit einem x- und y-Wert) können Sie Sprites mithilfe von CSS auf einfache Weise aus einem Sprite-Sheet extrahieren.

CSS Image Sprites online lesen: https://riptutorial.com/de/css/topic/3690/css-image-sprites

Kapitel 13: CSS-Entwurfsmuster

Einführung

Diese Beispiele dienen der Dokumentation von CSS-spezifischen Entwurfsmustern wie BEM , OOCSS und SMACSS .

Diese Beispiele dienen NICHT zum Dokumentieren von CSS-Frameworks wie Bootstrap oder Foundation .

Bemerkungen

Diese Beispiele dienen der Dokumentation von CSS-spezifischen Methoden / Entwurfsmustern.

Diese Methoden umfassen Folgendes, sind jedoch nicht ausschließlich:

- BEM
- OOCSS
- SMACSS

Diese Beispiele dienen NICHT zum Dokumentieren von CSS-Frameworks wie Bootstrap oder Foundation . Sie können zwar Beispiele für die Anwendung eines oder mehrerer CSS-Methoden / Designmuster in einem CSS-Framework hinzufügen, diese Beispiele konzentrieren sich jedoch auf die Methodologien / Designmuster mit diesem speziellen Framework und auf die Verwendung des Frameworks selbst.

Examples

BEM

BEM steht für `Blocks, Elements and Modifiers` . Diese Methode wurde ursprünglich vom russischen Technologieunternehmen Yandex konzipiert , erlangte jedoch auch bei amerikanischen und westeuropäischen Webentwicklern eine gewisse Wirkung.

Wie das Gleiche impliziert, geht es bei der BEM-Methode um die Komponentisierung Ihres HTML- und CSS-Codes in drei Arten von Komponenten:

- **Blöcke:** eigenständige Objekte, die eigenständig Bedeutung haben

  Beispiele sind `header` , `container` , `menu` , `checkbox` und `textbox`

- **Elemente:** Teil von Blöcken, die keine eigenständige Bedeutung haben und semantisch an ihre Blöcke gebunden sind.

  Beispiele sind `menu item` , `list item` , `checkbox caption` und `header title`

- **Modifikatoren:** Markierungen für einen Block oder ein Element, mit denen das Aussehen oder Verhalten geändert werden soll

  Beispiele hierfür sind `disabled` , `highlighted` , `checked` , `fixed` , `size big` und `color yellow`

Das Ziel von BEM ist es, die Lesbarkeit, Wartbarkeit und Flexibilität Ihres CSS-Codes zu optimieren. Um dies zu erreichen, müssen Sie die folgenden Regeln anwenden.

- Blockstile sind niemals von anderen Elementen auf einer Seite abhängig
- Blöcke sollten einen einfachen, kurzen Namen haben und _ oder - Zeichen vermeiden
- Verwenden Sie bei der `blockname__elementname` Elementen Selektoren des Formats `blockname__elementname`
- Verwenden Sie beim `blockname--modifiername` Selektoren für das Format `blockname--modifiername` und `blockname__elementname--modifiername`
- Elemente oder Blöcke, die über Modifizierer verfügen, sollten alles von dem Block oder Element erben, der geändert wird, mit Ausnahme der Eigenschaften, die der Modifizierer ändern soll

Code-Beispiel

Wenn Sie BEM auf Ihre Formularelemente anwenden, sollten Ihre CSS-Selektoren in etwa wie folgt aussehen:

```
.form { }                       // Block
.form--theme-xmas { }           // Block + modifier
.form--simple { }               // Block + modifier
.form__input { }                // Block > element
.form__submit { }               // Block > element
.form__submit--disabled { }     // Block > element + modifier
```

Das entsprechende HTML sollte ungefähr so aussehen:

```
<form class="form form--theme-xmas form--simple">
  <input class="form__input" type="text" />
  <input class="form__submit form__submit--disabled" type="submit" />
</form>
```

CSS-Entwurfsmuster online lesen: https://riptutorial.com/de/css/topic/10823/css-entwurfsmuster

Kapitel 14: CSS-Objektmodell (CSSOM)

Bemerkungen

Das CSS Object Model (CSSOM) ist eine Spezifikation für sich.

Den aktuellen Entwurf finden Sie hier: https://www.w3.org/TR/cssom-1/

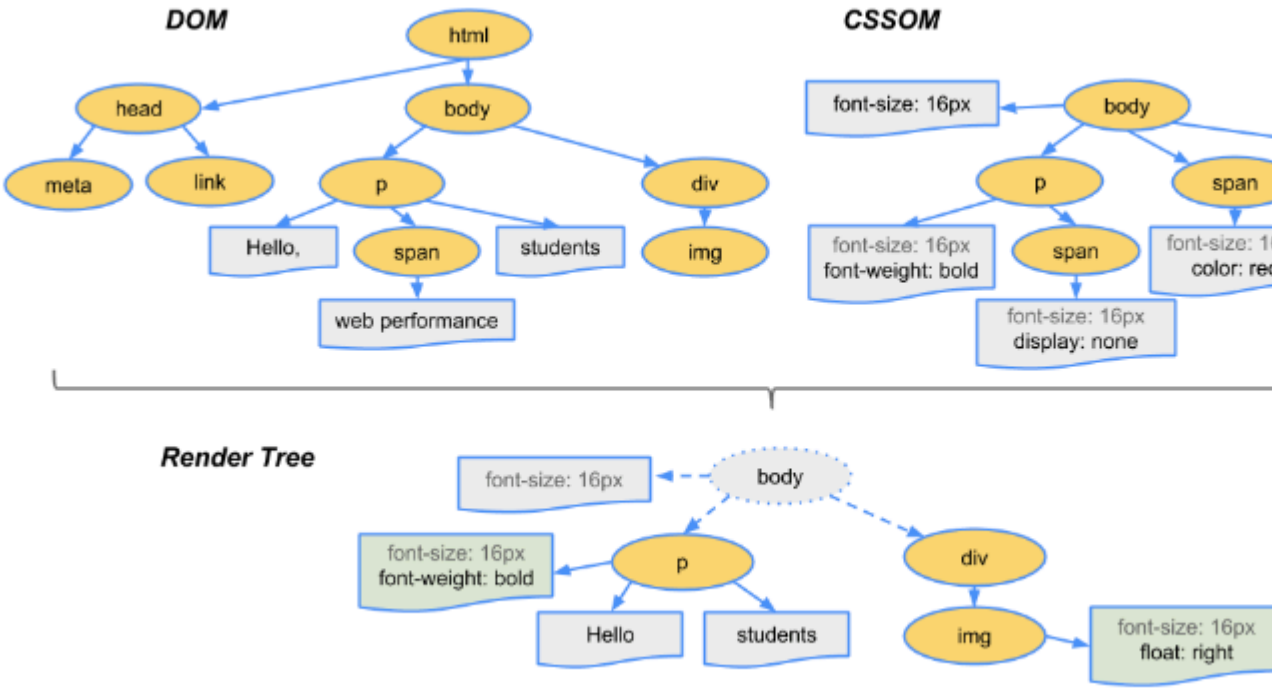
Examples

Einführung

Der Browser identifiziert Token aus Stylesheet und konvertiert sie in Knoten, die in einer Baumstruktur verknüpft sind. Die gesamte Karte aller Knoten mit den zugehörigen Stilen einer Seite wäre das CSS-Objektmodell.

Um die Webseite anzuzeigen, führt ein Webbrowser die folgenden Schritte aus.

1. Der Webbrowser untersucht Ihren HTML-Code und erstellt das DOM (Document Object Model).
2. Der Webbrowser untersucht Ihr CSS und erstellt das CSSOM (CSS Object Model).
3. Der Webbrowser kombiniert das DOM und das CSSOM, um einen Render-Baum zu erstellen. Der Webbrowser zeigt Ihre Webseite an.



Hinzufügen einer Hintergrundbildregel über CSSOM

Um eine Hintergrundbildregel über CSSOM hinzuzufügen, rufen Sie zunächst einen Verweis auf die Regeln des ersten Stylesheets ab:

```
var stylesheet = document.styleSheets[0].cssRules;
```

Dann erhalten Sie einen Verweis auf das Ende des Stylesheets:

```
var end = stylesheet.length - 1;
```

Fügen Sie schließlich eine Hintergrundbildregel für das body-Element am Ende des Stylesheets ein:

```
stylesheet.insertRule("body { background-image:
url('http://cdn.sstatic.net/Sites/stackoverflow/img/favicon.ico'); }", end);
```

CSS-Objektmodell (CSSOM) online lesen: https://riptutorial.com/de/css/topic/4961/css-objektmodell--cssom-

Kapitel 15: Cursor-Styling

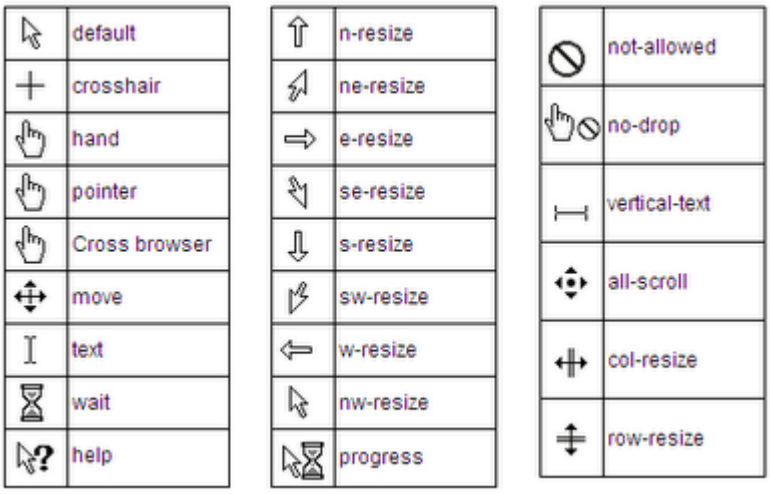
Syntax

- Cursor: auto | default | Keine | Kontextmenü | Hilfe | Zeiger | Fortschritt | warte | Zelle | Fadenkreuz | Text | vertikaler Text | Alias | kopieren | bewegen | no-drop | nicht erlaubt | e-resize | n-resize | ne-resize | nw-resize | s-resize | Größe ändern | sw-resize | w-resize | ew-resize | ns-resize | nesw-resize | nwse-resize | col-resize | Zeilengröße ändern all-scroll | Vergrößern | Verkleinern | schnappen | greifen

Examples

Cursortyp ändern

```
cursor: value;
```



Beispiele:

Wert	Beschreibung
keiner	Für das Element wird kein Cursor gerendert
Auto	Standard. Der Browser setzt einen Cursor
Hilfe	Der Cursor zeigt an, dass Hilfe verfügbar ist
warten	Der Cursor zeigt an, dass das Programm beschäftigt ist
Bewegung	Der Cursor zeigt an, dass etwas verschoben werden soll
Zeiger	Der Cursor ist ein Zeiger und zeigt eine Verknüpfung an

Zeigerereignisse

Mit der Eigenschaft pointer-events können Sie steuern, wie HTML-Elemente auf Maus- / Berührungsereignisse reagieren.

```
.disabled {
  pointer-events: none;
}
```

In diesem Beispiel

> 'none' verhindert alle Klick-, Status- und Cursoroptionen des angegebenen HTML-Elements [[1]]

Andere gültige Werte für HTMl-Elemente sind:

- Auto;
- erben.

1. https://css-tricks.com/almanac/properties/p/pointer-events/

Andere Ressourcen:

- https://developer.mozilla.org/en-US/docs/Web/CSS/pointer-events
- https://davidwalsh.name/pointer-events

Caret-Farbe

Die Caret-Color-CSS-Eigenschaft gibt die Farbe des Caret-Elements an, den sichtbaren Indikator für die Einfügemarke in einem Element, an dem Text und anderer Inhalt durch Eingabe oder Bearbeitung des Benutzers eingefügt werden.

HTML

```
<input id="example" />
```

CSS

```
#example {
  caret-color: red;
}
```

Ressourcen:

- https://developer.mozilla.org/en-US/docs/Web/CSS/caret-color

Cursor-Styling online lesen: https://riptutorial.com/de/css/topic/1742/cursor-styling

Kapitel 16: Das Boxmodell

Syntax

- Box-Dimensionierung: *Parameter* ;

Parameter

Parameter	Detail
`content-box`	Breite und Höhe des Elements umfassen nur den Inhaltsbereich.
`padding-box`	Breite und Höhe des Elements umfassen Inhalt und Auffüllung.
`border-box`	Breite und Höhe des Elements umfassen Inhalt, Auffüllung und Rand.
`initial`	Setzt das Boxmodell auf den Standardstatus.
`inherit`	Übernimmt das Boxmodell des übergeordneten Elements.

Bemerkungen

Über Padding-Box

Dieser Wert wurde nur von **Firefox** implementiert und sollte daher nicht verwendet werden. Es wurde in Firefox Version 50.0 entfernt.

Examples

Was ist das Boxmodell?

Die Kanten

Der Browser erstellt für jedes Element im HTML-Dokument ein Rechteck. Das Rahmenmodell beschreibt, wie die Auffüllung, der Rand und der Rand zum Inhalt hinzugefügt werden, um dieses Rechteck zu erstellen.

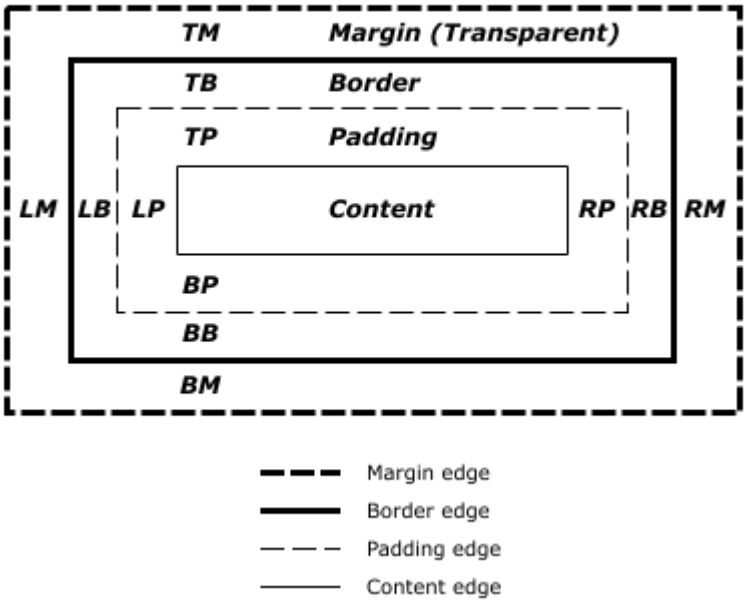


Diagramm aus CSS2.2 Arbeitsentwurf

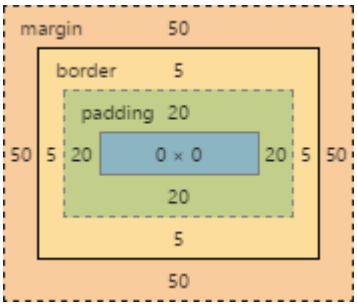
Der Umfang jedes der vier Bereiche wird als *Kante bezeichnet* . Jede Kante definiert eine *Box.*

- Das innerste Rechteck ist das **Inhaltsfeld** . Die Breite und Höhe davon hängt vom gerenderten Inhalt des Elements ab (Text, Bilder und alle untergeordneten Elemente).
- Als nächstes folgt die **Auffüllbox** , wie sie durch die `padding` Eigenschaft definiert wird. Wenn keine `padding` definiert ist, entspricht die Auffüllkante der Inhaltskante.
- Dann haben wir die **Grenze Box,** wie sie in der definierten `border` Eigenschaft. Wenn es keine `border` definiert ist , ist die Randkante an den Polsterrand gleich.
- Die äußerste Rechteck ist das **Randfeld** , wie durch die definierten `margin` Eigenschaft. Wenn keine `margin` definiert ist, entspricht die Randkante der Randkante.

Beispiel

```
div {
    border: 5px solid red;
    margin: 50px;
    padding: 20px;
}
```

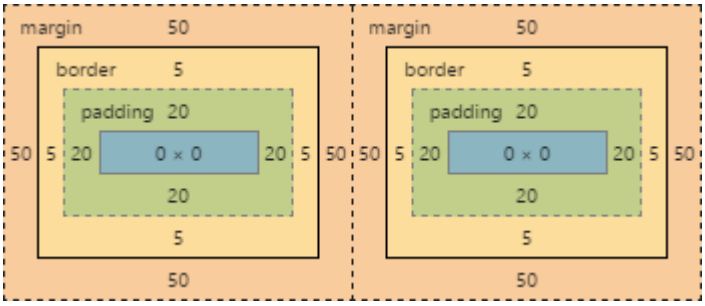
Dieses CSS formatiert alle `div` Elemente so, dass sie einen oberen, rechten, unteren und linken Rand mit `5px` Breite von `5px` . ein oberer, rechter, unterer und linker Rand von `50px` ; und eine obere, rechte, untere und linke Polsterung von `20px` . Wenn Sie den Inhalt ignorieren, wird unsere generierte Box folgendermaßen aussehen:



Screenshot des Bedienfelds Elementstile von Google Chrome

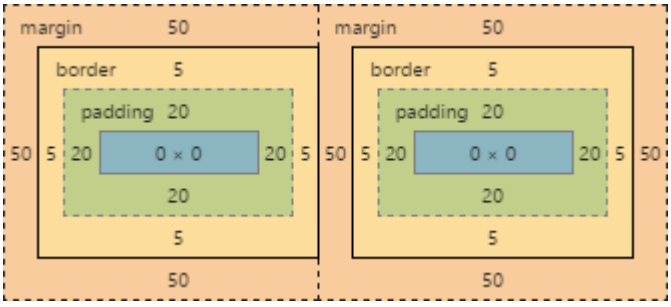
- Da es keinen Inhalt gibt, hat der Inhaltsbereich (das blaue Kästchen in der Mitte) keine Höhe oder Breite (0px zu 0px).
- Die Polsterung Box standardmäßig ist die gleiche Größe wie die Content - Box sowie die 20px Breite an allen vier Kanten oben mit dem wir definieren `padding` Eigenschaft (40px von 40px).
- Das Rahmenfeld hat dieselbe Größe wie das Füllfeld, zuzüglich der 5px-Breite, die wir oben mit der `border` Eigenschaft definieren (50px zu 50px).
- Schließlich hat das Randfeld die gleiche Größe wie das Rahmenfeld, zuzüglich der 50px-Breite, die wir oben mit der `margin` Eigenschaft definieren (wodurch unser Element eine Gesamtgröße von 150px zu 150px hat).

Jetzt geben wir unserem Element ein Geschwister mit demselben Stil. Der Browser betrachtet das Box-Modell beider Elemente, um herauszufinden, wo das neue Element in Bezug auf den Inhalt des vorherigen Elements positioniert werden soll:



Der Inhalt jedes Elements ist durch eine Lücke von 150 Pixel getrennt, aber die Kästchen der beiden Elemente berühren sich.

Wenn wir dann unser erstes Element so anpassen, dass es keinen rechten Rand hat, befindet sich der rechte Rand an derselben Position wie der rechte Rand, und unsere beiden Elemente würden jetzt so aussehen:



Box-Sizing

Das Standard - Box - Modell (`content-box`) kann kontraintuitiv sein, da die `width` / `height` für ein Element nicht seine tatsächliche Breite oder Höhe auf dem Bildschirm darstellen wird, sobald Sie anfangen , das Hinzufügen `padding` und `border` Stile an das Element.

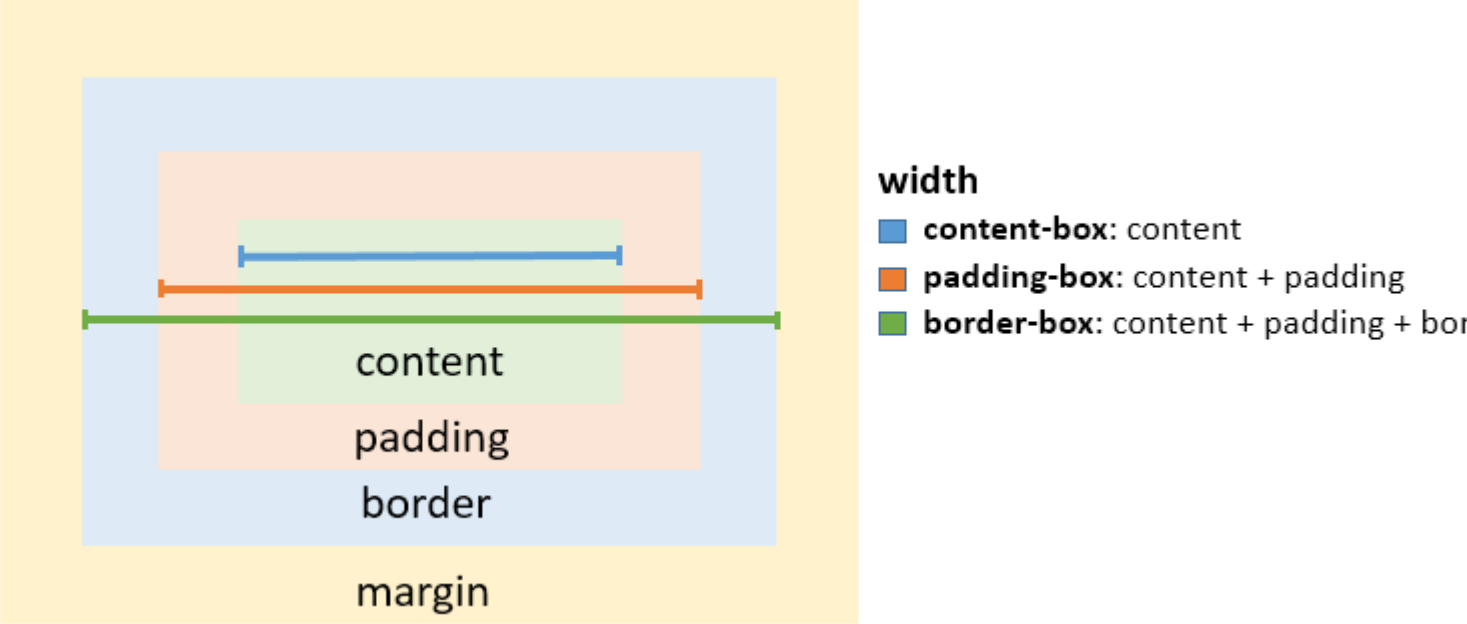
Das folgende Beispiel veranschaulicht dieses potenzielle Problem mit der `content-box` :

```
textarea {
    width: 100%;
    padding: 3px;
    box-sizing: content-box; /* default value */
}
```

Da der Abstand zur Breite des Textbereichs hinzugefügt wird, ist das resultierende Element ein Textbereich, der breiter als 100% ist.

Glücklicherweise ermöglicht CSS es uns, das Boxmodell mit der `box-sizing` Eigenschaft für ein Element zu ändern. Es gibt drei verschiedene Werte für die Eigenschaft:

- `content-box` : Das allgemeine Boxmodell - Breite und Höhe enthält nur den Inhalt, nicht die Auffüllung oder den Rand
- `padding-box` : Breite und Höhe umfassen den Inhalt und die Füllung, nicht jedoch den Rand
- `border-box` : Breite und Höhe umfassen den Inhalt, die Auffüllung sowie den Rand



Um das obige `textarea` Problem zu lösen, können Sie einfach die Eigenschaft " `box-sizing` in " `padding-box` oder " `border-box` ändern. `border-box` wird am häufigsten verwendet.

```
textarea {
    width: 100%;
```

```
    padding: 3px;
    box-sizing: border-box;
}
```

Um ein bestimmtes Box-Modell auf jedes Element auf der Seite anzuwenden, verwenden Sie das folgende Snippet:

```
html {
    box-sizing: border-box;
}

*, *:before, *:after {
    box-sizing: inherit;
}
```

In dieser Codierungsbox `box-sizing:border-box;` wird nicht direkt auf `*` angewendet, sodass Sie diese Eigenschaft leicht für einzelne Elemente überschreiben können.

Das Boxmodell online lesen: https://riptutorial.com/de/css/topic/646/das-boxmodell

Kapitel 17: Eigenschaft filtern

Syntax

- filter: none (Standardwert)
- filter: initial (Standardeinstellung ist none);
- filter: vererben (Standardwert ist der übergeordnete Wert);
- Filter: Unschärfe (px)
- Filter: Helligkeit (Anzahl |%)
- Filter: Kontrast (Anzahl |%)
- Filter: Schattenwurf (horizontal-shadow-px vertikal-shadow-px shadow-blur-px shadow - Farbe ausbreiten
- Filter: Graustufen (Anzahl |%)
- Filter: Farbton-Drehung (Grad)
- Filter: invertieren (Anzahl |%)
- Filter: Deckkraft (Anzahl |%)
- Filter: gesättigt (Anzahl |%)
- Filter: Sepia (Anzahl |%)

Parameter

Wert	Beschreibung
Unschärfe (x)	Verwischt das Bild um x Pixel.
Helligkeit (x)	Hellt das Bild auf einen Wert über 1,0 oder 100% auf. Darunter wird das Bild dunkler.
Kontrast (x)	Bietet bei einem Wert über 1,0 oder 100% mehr Kontrast zum Bild. Darunter wird das Bild weniger gesättigt.
Schlagschatten (h, v, x, y, z)	Verleiht dem Bild einen Schatten. h und v können negative Werte haben. x, y und z sind optional.
Graustufen (x)	Zeigt das Bild in Graustufen mit einem Maximalwert von 1,0 oder 100% an.
Farbton drehen (x)	Wendet eine Farbrotation auf das Bild an
invertieren (x)	Kehrt die Farbe des Bildes mit einem Maximalwert von 1,0 oder 100% um.
Deckkraft (x)	Legt fest, wie undurchsichtig / transparent das Bild mit einem Höchstwert von 1,0 oder 100% ist.
gesättigt (x)	Sättigt das Bild bei einem Wert über 1,0 oder 100%. Darunter

Wert	Beschreibung
	beginnt das Bild zu sättigen.
Sepia (x)	Wandelt das Bild mit einem Maximalwert von 1,0 oder 100% in Sepia um.

Bemerkungen

1. Da der Filter eine experimentelle Funktion ist, sollten Sie das Präfix -webkit verwenden. Es kann sich in Syntax und Verhalten ändern, aber die Änderungen werden wahrscheinlich gering sein.
2. Es wird möglicherweise in älteren Versionen der wichtigsten Browser nicht unterstützt. In mobilen Browsern wird dies möglicherweise nicht unterstützt.
3. Versuchen Sie aufgrund der relativ eingeschränkten Unterstützung, `box-shadow` anstelle von `filter: drop-shadow()` . Verwenden Sie `opacity` anstelle von `filter: opacity()` .
4. Es kann durch Javascript / jQuery animiert werden. Verwenden `object.style.WebkitFilter` für Javascript `object.style.WebkitFilter` .
5. Überprüfen Sie W3Schools oder MDN für weitere Informationen.
6. W3Schools hat auch eine Demo - Seite für alle die verschiedenen Arten von Filterwerten.

Examples

Schlagschatten (wenn möglich Box-Schatten verwenden)

HTML

```
<p>My shadow always follows me.</p>
```

CSS

```
p {
    -webkit-filter: drop-shadow(10px 10px 1px green);
    filter: drop-shadow(10px 10px 1px green);
}
```

Ergebnis

Mehrere Filterwerte

Um mehrere Filter zu verwenden, trennen Sie jeden Wert durch ein Leerzeichen.

HTML

```
<img src='donald-duck.png' alt='Donald Duck' title='Donald Duck' />
```

CSS

```
img {
    -webkit-filter: brightness(200%) grayscale(100%) sepia(100%) invert(100%);
    filter: brightness(200%) grayscale(100%) sepia(100%) invert(100%);
}
```

Ergebnis

Farbton drehen

HTML

```
<img src='donald-duck.png' alt='Donald Duck' title='Donald Duck' />
```

CSS

```
img {
    -webkit-filter: hue-rotate(120deg);
    filter: hue-rotate(120deg);
}
```

Ergebnis

Farbe umkehren

HTML

```
<div></div>
```

CSS

```
div {
    width: 100px;
    height: 100px;
    background-color: white;
    -webkit-filter: invert(100%);
    filter: invert(100%);
}
```

Ergebnis

Wechselt von Weiß zu Schwarz.

Verwischen

HTML

```
<img src='donald-duck.png' alt='Donald Duck' title='Donald Duck' />
```

CSS

```
img {
    -webkit-filter: blur(1px);
    filter: blur(1px);
}
```

Ergebnis

Willst du deine Brille reiben?

Eigenschaft filtern online lesen: https://riptutorial.com/de/css/topic/1567/eigenschaft-filtern

Kapitel 18: Einzelelementformen

Examples

Quadrat

Um ein Quadrat zu erstellen, definieren Sie ein Element mit Breite und Höhe. Im folgenden Beispiel haben wir ein Element mit einer `width` und `height` von jeweils 100 Pixel.

```
<div class="square"></div>
```

```
.square {
    width: 100px;
    height: 100px;
    background: rgb(246, 156, 85);
}
```

Dreiecke

Um ein CSS-Dreieck zu erstellen, definieren Sie ein Element mit einer Breite und Höhe von 0 Pixel. Die Dreiecksform wird unter Verwendung von Rahmeneigenschaften gebildet. Bei einem Element mit 0 Höhe und Breite bilden die 4 Ränder (oben, rechts, unten, links) jeweils ein Dreieck. Hier ist ein Element mit 0 Höhe / Breite und 4 verschiedenen farbigen Rändern.

Wenn Sie einige Ränder auf transparent setzen und andere auf eine Farbe, können wir verschiedene Dreiecke erstellen. In dem Aufwärts-Dreieck setzen wir beispielsweise den unteren Rand auf die gewünschte Farbe und dann den linken und den rechten Rand auf transparent. Hier ist ein Bild, dessen linker und rechter Rand leicht schattiert sind, um zu zeigen, wie das Dreieck geformt wird.

Die Abmessungen des Dreiecks können geändert werden, indem Sie die verschiedenen Randbreiten ändern - größer, kürzer, schief, usw. Die folgenden Beispiele zeigen alle ein Dreieck mit 50 x 50 Pixeln.

Dreieck - zeigt nach oben

```
<div class="triangle-up"></div>
```

```
.triangle-up {
  width: 0;
  height: 0;
  border-left: 25px solid transparent;
  border-right: 25px solid transparent;
  border-bottom: 50px solid rgb(246, 156, 85);
}
```

Dreieck - zeigt nach unten

```
<div class="triangle-down"></div>
```

```
.triangle-down {
  width: 0;
  height: 0;
  border-left: 25px solid transparent;
  border-right: 25px solid transparent;
  border-top: 50px solid rgb(246, 156, 85);
}
```

Dreieck - nach rechts zeigend

```
<div class="triangle-right"></div>
```

```
.triangle-right {
  width: 0;
  height: 0;
  border-top: 25px solid transparent;
  border-bottom: 25px solid transparent;
  border-left: 50px solid rgb(246, 156, 85);
}
```

Dreieck - zeigt nach links

```
<div class="triangle-left"></div>
```

```
.triangle-left {
  width: 0;
  height: 0;
  border-top: 25px solid transparent;
  border-bottom: 25px solid transparent;
  border-right: 50px solid rgb(246, 156, 85);
}
```

Dreieck - nach oben / rechts zeigen

```
<div class="triangle-up-right"></div>
```

```
.triangle-up-right {
  width: 0;
  height: 0;
  border-top: 50px solid rgb(246, 156, 85);
  border-left: 50px solid transparent;
}
```

Dreieck - nach oben / links zeigen

```
<div class="triangle-up-left"></div>
```

```
.triangle-up-left {
  width: 0;
  height: 0;
  border-top: 50px solid rgb(246, 156, 85);
  border-right: 50px solid transparent;
}
```

Dreieck - nach unten / rechts zeigen

```
<div class="triangle-down-right"></div>
```

```
.triangle-down-right {
  width: 0;
  height: 0;
  border-bottom: 50px solid rgb(246, 156, 85);
  border-left: 50px solid transparent;
}
```

Dreieck - nach unten / links zeigen

```
<div class="triangle-down-left"></div>
```

```
.triangle-down-left {
  width: 0;
  height: 0;
  border-bottom: 50px solid rgb(246, 156, 85);
  border-right: 50px solid transparent;
```

```
}
```

Platzt

Ein Burst ähnelt einem Stern, aber die Punkte erstrecken sich weniger vom Körper. Stellen Sie sich eine Burst-Form als Quadrat vor, mit zusätzlichen, leicht gedrehten Quadraten, die darüber angeordnet sind.

Die zusätzlichen Quadrate werden mit den Elementen `::before` und `::after` psuedo erstellt.

8 Punkt Burst

Ein 8-Punkt-Burst besteht aus 2 geschichteten Quadraten. Das untere Quadrat ist das Element selbst, das zusätzliche Quadrat wird mit dem Pseudoelement `:before` . Die Unterseite ist um 20 ° gedreht, das obere Quadrat um 135 °.

```
<div class="burst-8"></div>

.burst-8 {
  background: rgb(246, 156, 85);
  width: 40px;
  height: 40px;
  position: relative;
  text-align: center;
  -ms-transform: rotate(20deg);
  transform: rotate(20eg);
}

.burst-8::before {
  content: "";
  position: absolute;
  top: 0;
  left: 0;
  height: 40px;
  width: 40px;
  background: rgb(246, 156, 85);
  -ms-transform: rotate(135deg);
  transform: rotate(135deg);
}
```

12 Punkt Burst

Ein 12-Punkt-Burst besteht aus 3 geschichteten Quadraten. Das untere Quadrat ist das Element selbst, die zusätzlichen Quadrate werden mit den Pseudoelementen `:before` und `:after` . Die Unterseite wird um 0 ° gedreht, das nächste Quadrat wird um 30 ° und die Oberseite um 60 ° gedreht.

```
<div class="burst-12"></div>

.burst-12 {
  width: 40px;
  height: 40px;
  position: relative;
  text-align: center;
  background: rgb(246, 156, 85);
}

.burst-12::before, .burst-12::after {
  content: "";
  position: absolute;
  top: 0;
  left: 0;
  height: 40px;
  width: 40px;
  background: rgb(246, 156, 85);
}

.burst-12::before {
  -ms-transform: rotate(30deg);
  transform: rotate(30deg);
}

.burst-12::after {
  -ms-transform: rotate(60deg);
  transform: rotate(60deg);
}
```

Kreise und Ellipsen

Kreis

Um einen **Kreis** zu erstellen, definieren Sie ein Element mit gleicher `width` und `height` (ein *Quadrat*) und setzen Sie dann die Eigenschaft `border-radius` dieses Elements auf `50%` .

HTML

```
<div class="circle"></div>
```

CSS

```
.circle {
   width: 50px;
   height: 50px;
   background: rgb(246, 156, 85);
   border-radius: 50%;
}
```

Ellipse

Eine **Ellipse** ähnelt einem Kreis, jedoch mit unterschiedlichen Werten für `width` und `height` .

HTML

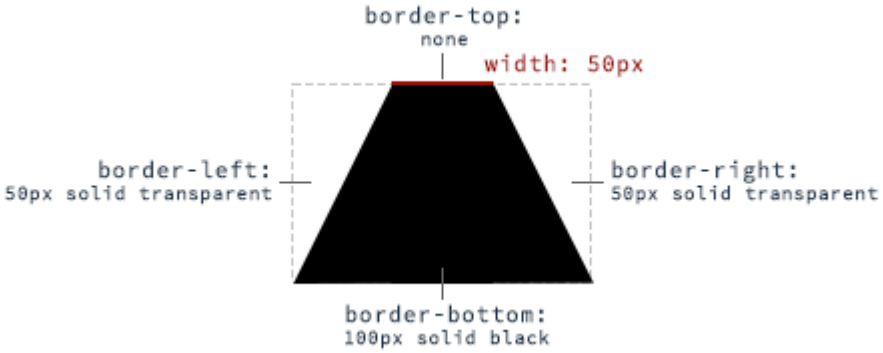
```
<div class="oval"></div>
```

CSS

```
.oval {
  width: 50px;
  height: 80px;
  background: rgb(246, 156, 85);
  border-radius: 50%;
}
```

Trapezoid

Ein Trapez kann durch ein Blockelement mit einer Höhe von Null (Höhe von `0px`), einer Breite größer als Null und einem Rand gebildet werden, der bis auf eine Seite transparent ist:



HTML:

```
<div class="trapezoid"></div>
```

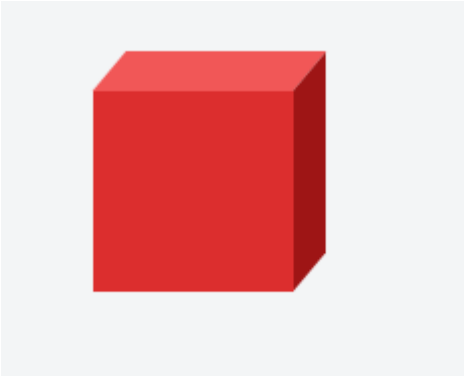
CSS:

```
.trapezoid {
    width: 50px;
    height: 0;
    border-left: 50px solid transparent;
    border-right: 50px solid transparent;
    border-bottom: 100px solid black;
}
```

Durch Ändern der Randseiten kann die Ausrichtung des Trapezes eingestellt werden.

Würfel

Dieses Beispiel zeigt, wie Sie einen Cube mit den 2D-Transformationsmethoden `skewX()` und `skewY()` für `skewY()` erstellen.



HTML:

```
<div class="cube"></div>
```

CSS:

```
.cube {
  background: #dc2e2e;
  width: 100px;
  height: 100px;
  position: relative;
  margin: 50px;
}

.cube::before {
  content: '';
  display: inline-block;
  background: #f15757;
  width: 100px;
  height: 20px;
  transform: skewX(-40deg);
  position: absolute;
  top: -20px;
  left: 8px;
```

```
}

.cube::after {
  content: '';
  display: inline-block;
  background: #9e1515;
  width: 16px;
  height: 100px;
  transform: skewY(-50deg);
  position: absolute;
  top: -10px;
  left: 100%;
}
```

Siehe Demo

Pyramide

Verwendung von Grenzen und 2D - Transformationsverfahren Dieses Beispiel zeigt , wie eine **Pyramide** erzeugen `skewY()` und `rotate()` auf pseudo - Elementen.

HTML:

```
<div class="pyramid"></div>
```

CSS:

```
.pyramid {
  width: 100px;
  height: 200px;
  position: relative;
  margin: 50px;
}

.pyramid::before,.pyramid::after {
  content: '';
  display: inline-block;
  width: 0;
  height: 0;
  border: 50px solid;
  position: absolute;
}

.pyramid::before {
  border-color: transparent transparent #ff5656 transparent;
```

```
  transform: scaleY(2) skewY(-40deg) rotate(45deg);
}

.pyramid::after {
  border-color: transparent transparent #d64444 transparent;
  transform: scaleY(2) skewY(40deg) rotate(-45deg);
}
```

Einzelelementformen online lesen: https://riptutorial.com/de/css/topic/2862/einzelelementformen

Kapitel 19: Erbe

Syntax

- *Eigenschaft:* erben;

Examples

Automatische Vererbung

Vererbung der grundlegende Mechanismus von CSS, durch den die berechneten Werte einiger Eigenschaften eines Elements auf seine untergeordneten Elemente angewendet werden. Dies ist besonders nützlich, wenn Sie Ihren Elementen einen globalen Stil zuweisen möchten, anstatt die besagten Eigenschaften für jedes einzelne Element in Ihrem Markup festlegen zu müssen.

Übliche Eigenschaften, die automatisch vererbt werden, sind: `font` , `color` , `text-align` , `line-height` .

Nehmen Sie das folgende Stylesheet an:

```
#myContainer {
  color: red;
  padding: 5px;
}
```

Dies gilt für `color: red` nicht nur für das Element `<div>` , sondern auch für die Elemente `<h3>` und `<p>` . Aufgrund der Art der `padding` dieser Wert jedoch ***nicht*** an diese Elemente vererbt.

```
<div id="myContainer">
  <h3>Some header</h3>
  <p>Some paragraph</p>
</div>
```

Erzwungene Vererbung

Einige Eigenschaften werden nicht automatisch von einem Element an die untergeordneten Elemente vererbt. Dies liegt daran, dass diese Eigenschaften normalerweise für das Element (oder die Auswahl der Elemente), auf das die Eigenschaft angewendet wird, eindeutig sein sollen. Häufig sind solche Eigenschaften `margin` , `padding` , `background` , `display` usw.

Manchmal ist jedoch trotzdem Vererbung erwünscht. Um dies zu erreichen, können wir die Anwendung `inherit` Wert auf die Eigenschaft , die vererbt werden soll. Der `inherit` Wert kann auf *jede* CSS - Eigenschaft und *jedes beliebigen* HTML - Element appied werden.

Nehmen Sie das folgende Stylesheet an:

```
#myContainer {
  color: red;
  padding: 5px;
}
#myContainer p {
  padding: inherit;
}
```

Dadurch wird `color: red` auf die Elemente `<h3>` und `<p>` angewendet, da die `color` Eigenschaft vererbt ist. Das `<p>` -Element erbt jedoch auch den `padding` von seinem übergeordneten Element, da dies angegeben wurde.

```
<div id="myContainer">
  <h3>Some header</h3>
  <p>Some paragraph</p>
</div>
```

Erbe online lesen: https://riptutorial.com/de/css/topic/3586/erbe

Kapitel 20: Farben

Syntax

- Farbe: #rgb
- Farbe: #rrggbb
- Farbe: rgb [a] (<rot>, <grün>, <blau> [, <alpha>])
- Farbe: hsl [a] (<Farbton>, <Sättigung%>, <Helligkeit%> [, <Alpha>])
- Farbe: colorkeyword / * grün, blau, gelb, orange, rot, .. etc * /

Examples

Farbe Schlüsselwörter

Die meisten Browser unterstützen die Verwendung von Farbschlüsselwörtern, um eine Farbe anzugeben. Um beispielsweise die `color` eines Elements auf Blau zu setzen, verwenden Sie das `blue` Schlüsselwort:

```
.some-class {
    color: blue;
}
```

Bei CSS-Schlüsselwörtern wird die Groß- und Kleinschreibung nicht `#0000FF` . `blue` , `Blue` und `BLUE` führen zu `#0000FF` .

Farbe Schlüsselwörter

Name der Farbe	Hex-Wert	RGB-Werte	Farbe
Alice blau	# F0F8FF	rgb (240.248.255)	
Altweiß	# FAEBD7	rgb (250,235,215)	
Aqua	# 00FFFF	rgb (0,255,255)	
Aquamarin	# 7FFFD4	rgb (127,255,212)	
Azurblau	# F0FFFF	rgb (240,255,255)	
Beige	# F5F5DC	rgb (245,245,220)	
Bisque	# FFE4C4	rgb (255,228,196)	

Name der Farbe	Hex-Wert	RGB-Werte	Farbe
Schwarz	# 000000	rgb (0,0,0)	
Blanchierte Mandel	#FFEBCD	rgb (255,235,205)	
Blau	# 0000FF	rgb (0,0,255)	
Blau Violett	# 8A2BE2	rgb (138,43,226)	
Braun	# A52A2A	rgb (165,42,42)	
Kräftiges Holz	# DEB887	rgb (222, 184, 135)	
Kadettenblau	# 5F9EA0	rgb (95,158,160)	
Chartreuse	# 7FFF00	rgb (127,255,0)	
Schokolade	# D2691E	rgb (210,105,30)	
Koralle	# FF7F50	rgb (255,127,80)	
Kornblumenblau	# 6495ED	rgb (100,149,237)	
Cornsilk	# FFF8DC	rgb (255,248,220)	
Crimson	# DC143C	rgb (220,20,60)	
Cyan	# 00FFFF	rgb (0,255,255)	
Dunkelblau	# 00008B	rgb (0,0,139)	
DarkCyan	# 008B8B	rgb (0,139,139)	
DarkGoldenRod	# B8860B	rgb (184,134,11)	
Dunkelgrau	# A9A9A9	rgb (169,169,169)	
Dunkelgrau	# A9A9A9	rgb (169,169,169)	
Dunkelgrün	# 006400	rgb (0,100,0)	
DarkKhaki	# BDB76B	rgb (189.183.107)	

Name der Farbe	Hex-Wert	RGB-Werte	Farbe
DarkMagenta	# 8B008B	rgb (139,0,139)	
DarkOliveGreen	# 556B2F	rgb (85,107,47)	
Dunkelorange	# FF8C00	rgb (255,140,0)	
DarkOrchid	# 9932CC	rgb (153,50,204)	
Dunkelrot	# 8B0000	rgb (139,0,0)	
DarkSalmon	# E9967A	rgb (233,150,122)	
DarkSeaGreen	# 8FBC8F	rgb (143,188,143)	
DarkSlateBlue	# 483D8B	rgb (72,61,139)	
DarkSlateGray	# 2F4F4F	rgb (47,79,79)	
DarkSlateGrey	# 2F4F4F	rgb (47,79,79)	
DarkTurquoise	# 00CED1	rgb (0,206,209)	
Dunkelviolett	# 9400D3	rgb (148,0,211)	
Dunkelrosa	# FF1493	rgb (255,20,147)	
DeepSkyBlue	# 00BFFF	rgb (0,191,255)	
DimGray	# 696969	rgb (105,105,105)	
DimGrey	# 696969	rgb (105,105,105)	
DodgerBlue	# 1E90FF	rgb (30.144.255)	
FireBrick	# B22222	rgb (178,34,34)	
FloralWhite	# FFFAF0	rgb (255,250,240)	
Waldgrün	# 228B22	rgb (34,139,34)	
Fuchsie	# FF00FF	rgb (255,0,255)	

Name der Farbe	Hex-Wert	RGB-Werte	Farbe
Gainsboro	#DCDCDC	rgb (220,220,220)	
GhostWhite	# F8F8FF	rgb (248,248,255)	
Gold	# FFD700	rgb (255,215,0)	
Goldrute	# DAA520	rgb (218,165,32)	
Grau	# 808080	rgb (128,128,128)	
Grau	# 808080	rgb (128,128,128)	
Grün	# 008000	rgb (0,128,0)	
Grün Gelb	# ADFF2F	rgb (173,255,47)	
Honigtau	# F0FFF0	rgb (240,255,240)	
HotPink	# FF69B4	rgb (255,105,180)	
IndianRed	# CD5C5C	rgb (205,92,92)	
Indigo	# 4B0082	rgb (75,0,130)	
Elfenbein	# FFFFF0	rgb (255,255,240)	
Khaki	# F0E68C	rgb (240,230,140)	
Lavendel	# E6E6FA	rgb (230.230.250)	
LavenderBlush	# FFF0F5	rgb (255,240,245)	
Rasengrün	# 7CFC00	rgb (124.252,0)	
ZitronenChiffon	#FFFACD	rgb (255,250,205)	
Hellblau	# ADD8E6	rgb (173,216,230)	
LightCoral	# F08080	rgb (240,128,128)	
LightCyan	# E0FFFF	rgb (224,255,255)	

Name der Farbe	Hex-Wert	RGB-Werte	Farbe
LightGoldenRodYellow	# FAFAD2	rgb (250,250,210)	
Hellgrau	# D3D3D3	rgb (211,211,211)	
Hellgrau	# D3D3D3	rgb (211,211,211)	
Hellgrün	# 90EE90	rgb (144.238.144)	
Hell-Pink	# FFB6C1	rgb (255.182.193)	
LightSalmon	# FFA07A	rgb (255,160,122)	
LightSeaGreen	# 20B2AA	rgb (32.178.170)	
LightSkyBlue	# 87CEFA	rgb (135,206,250)	
LightSlateGray	# 778899	rgb (119,136,153)	
LightSlateGrey	# 778899	rgb (119,136,153)	
LightSteelBlue	# B0C4DE	rgb (176,196,222)	
Hellgelb	# FFFFE0	rgb (255,255,224)	
Limette	# 00FF00	rgb (0,255,0)	
LimeGreen	# 32CD32	rgb (50,205,50)	
Leinen	# FAF0E6	rgb (250,240,230)	
Magenta	# FF00FF	rgb (255,0,255)	
Kastanienbraun	800000	rgb (128,0,0)	
MediumAquaMarine	# 66CDAA	rgb (102,205,170)	
Mittelblau	# 0000CD	rgb (0,0,205)	
MediumOrchid	# BA55D3	rgb (186,85,211)	
MediumPurple	# 9370DB	rgb (147,112,219)	

Name der Farbe	Hex-Wert	RGB-Werte	Farbe
MediumSeaGreen	# 3CB371	rgb (60,179,113)	
MediumSlateBlue	# 7B68EE	rgb (123,104,238)	
MediumSpringGreen	# 00FA9A	rgb (0,250,154)	
MediumTurquoise	# 48D1CC	rgb (72,209,204)	
MediumVioletRed	# C71585	rgb (199,21,133)	
Mitternachtsblau	# 191970	rgb (25, 25, 112)	
MintCream	# F5FFFA	rgb (245,255,250)	
MistyRose	# FFE4E1	rgb (255,228,225)	
Mokassin	# FFE4B5	rgb (255,228,181)	
NavajoWeiß	#FFDEAD	rgb (255,222,173)	
Marine	# 000080	rgb (0,0,128)	
OldLace	# FDF5E6	rgb (253,245,230)	
Olive	# 808000	rgb (128.128,0)	
Olivgrün	# 6B8E23	rgb (107,142,35)	
Orange	# FFA500	rgb (255,165,0)	
Orange Rot	# FF4500	rgb (255,69,0)	
Orchidee	# DA70D6	rgb (218,112,214)	
PaleGoldenRod	# EEE8AA	rgb (238,232,170)	
Blasses Grün	# 98FB98	rgb (152,251,152)	
PaleTurquoise	#AFEEEE	rgb (175,238,238)	
PaleVioletRed	# DB7093	rgb (219,112,147)	

Name der Farbe	Hex-Wert	RGB-Werte	Farbe
PapayaWhip	# FFEFD5	rgb (255,239,213)	
PeachPuff	# FFDAB9	rgb (255,218,185)	
Peru	# CD853F	rgb (205,133,63)	
Rosa	# FFC0CB	rgb (255,192,203)	
Pflaume	# DDA0DD	rgb (221,160,221)	
PowderBlue	# B0E0E6	rgb (176,224,230)	
Lila	# 800080	rgb (128,0,128)	
RebeccaPurple	# 663399	rgb (102,51,153)	
rot	# FF0000	rgb (255,0,0)	
RosyBrown	# BC8F8F	rgb (188,143,143)	
Königsblau	# 4169E1	rgb (65,105,225)	
Sattelbraun	# 8B4513	rgb (139,69,19)	
Lachs	# FA8072	rgb (250,128,114)	
SandyBrown	# F4A460	rgb (244,164,96)	
Meeresgrün	# 2E8B57	rgb (46,139,87)	
Muschel	# FFF5EE	rgb (255,245,238)	
Sienna	# A0522D	rgb (160,82,45)	
Silber	# C0C0C0	rgb (192,192,192)	
Himmelblau	# 87CEEB	rgb (135,206,235)	
SlateBlue	# 6A5ACD	rgb (106,90,205)	
Schiefer grau	# 708090	rgb (112.128.144)	

Name der Farbe	Hex-Wert	RGB-Werte	Farbe
Schiefergrau	# 708090	rgb (112.128.144)	
Schnee	#FFFAFA	rgb (255,250,250)	
Frühlingsgrün	# 00FF7F	rgb (0,255,127)	
Stahlblau	# 4682B4	rgb (70,130,180)	
Bräunen	# D2B48C	rgb (210,180,140)	
Teal	# 008080	rgb (0,128,128)	
Distel	# D8BFD8	rgb (216,191,216)	
Tomate	# FF6347	rgb (255,99,71)	
Türkis	# 40E0D0	rgb (64.224.208)	
Violett	# EE82EE	rgb (238,130,238)	
Weizen	# F5DEB3	rgb (245,222,179)	
Weiß	#FFFFFF	rgb (255,255,255)	
Weißer Rauch	# F5F5F5	rgb (245,245,245)	
Gelb	# FFFF00	rgb (255,255,0)	
Gelbgrün	# 9ACD32	rgb (154.205,50)	

Zusätzlich zu den genannten Farben, gibt es auch das Schlüsselwort `transparent` , die einen voll-transparent schwarz darstellt: `rgba(0,0,0,0)`

Hexadezimalwert

Hintergrund

CSS-Farben können auch als Hex-Triplett dargestellt werden, wobei die Mitglieder die roten, grünen und blauen Komponenten einer Farbe darstellen. Jeder dieser Werte steht für eine Zahl im Bereich von `00` bis `FF` oder von `0` bis `255` in Dezimalschreibweise. Hexidecimal-Werte können in

Groß- und / oder Kleinbuchstaben verwendet werden (dh `#3fc` = `#3FC` = `#33ffCC`). Der Browser interpretiert `#369` als `#336699` . Wenn dies nicht das ist, was Sie beabsichtigten, sondern `#306090` , wollten Sie dies explizit angeben.

Die Gesamtzahl der Farben, die mit Hex-Notation dargestellt werden können, beträgt 256 ^ 3 oder 16.777.216.

Syntax

```
color: #rrggbb;
color: #rgb
```

Wert	Beschreibung
`rr`	`00` - `FF` für den Rotanteil
`gg`	`00` - `FF` für die Grünmenge
`bb`	`00` - `FF` für die blaue Menge

```
.some-class {
    /* This is equivalent to using the color keyword 'blue' */
    color: #0000FF;
}

.also-blue {
    /* If you want to specify each range value with a single number, you can!
       This is equivalent to '#0000FF' (and 'blue') */
    color: #00F;
}
```

Die Hexadezimalschreibweise wird verwendet, um Farbwerte im RGB-Farbformat gemäß den 'Numerischen Farbwerten' des W3C anzugeben.

Im Internet gibt es eine Vielzahl von Tools, mit denen Sie nach hexadezimalen (oder einfach hexadezimalen) Farbwerten suchen können.

Suchen Sie nach " **Hex-Farbpalette** " oder " **Hex-Farbauswahl** " mit Ihrem bevorzugten Webbrowser, um eine Reihe von Optionen zu finden!

Hexadezimalwerte beginnen immer mit einem Nummernzeichen (#), sind bis zu sechs "Ziffern" lang und unterscheiden nicht zwischen Groß- und Kleinschreibung. `#FFC125` und `#ffc125` haben dieselbe Farbe.

rgb () Notation

RGB ist ein additives Farbmodell, das Farben als Mischungen aus rotem, grünem und blauem Licht darstellt. Im Wesentlichen ist die RGB-Darstellung das dezimale Äquivalent der Hexadezimal-Notation. Im Hexadezimalbereich reicht jede Zahl von 00-FF, was 0 bis 255 in

Dezimalzahlen und 0 bis 100% in Prozent entspricht.

```
.some-class {
    /* Scalar RGB, equivalent to 'blue'*/
    color: rgb(0, 0, 255);
}

.also-blue {
    /* Percentile RGB values*/
    color: rgb(0%, 0%, 100%);
}
```

Syntax

```
rgb(<red>, <green>, <blue>)
```

Wert	Beschreibung
`<red>`	eine ganze Zahl von 0 - 255 oder Prozentsatz von 0 - 100%
`<green>`	eine ganze Zahl von 0 - 255 oder Prozentsatz von 0 - 100%
`<blue>`	eine ganze Zahl von 0 - 255 oder Prozentsatz von 0 - 100%

hsl () Notation

HSL steht für **Farbton** ("welche Farbe"), **Sättigung** ("wie viel Farbe") und **Helligkeit** ("wie viel Weiß").

Der Farbton wird als Winkel von 0 ° bis 360 ° (ohne Einheiten) dargestellt, während Sättigung und Helligkeit in Prozent dargestellt werden.

```
p {
    color: hsl(240, 100%, 50%); /* Blue */
}
```

Syntax

```
color: hsl(<hue>, <saturation>%, <lightness>%);
```

Wert	Beschreibung
`<hue>`	um den Farbkreis herum (ohne Einheiten) angegeben, wobei 0 ° rot, 60 ° gelb, 120 ° grün, 180 ° cyan, 240 ° blau, 300 ° magenta und 360 ° rot sind
`<saturation>`	in Prozent angegeben, wobei 0% vollständig entsättigt (Graustufen) und 100% vollständig gesättigt sind (kräftig gefärbt)

Wert	Beschreibung
`<lightness>`	in Prozent angegeben, wobei 0% vollständig schwarz und 100% vollständig weiß sind

Anmerkungen

- Eine Sättigung von 0% erzeugt immer eine Graustufenfarbe; Das Ändern des Farbtons hat keine Auswirkung.
- Eine Helligkeit von 0% erzeugt immer Schwarz, und 100% erzeugt immer Weiß; Das Ändern des Farbtons oder der Sättigung hat keine Auswirkung.

currentColor

`currentColor` gibt den berechneten Farbwert des aktuellen Elements zurück.

Verwenden Sie in demselben Element

Hier wird currentColor zu Rot ausgewertet, da die `color` auf `red` :

```
div {
   color: red;
   border: 5px solid currentColor;
   box-shadow: 0 0 5px currentColor;
}
```

In diesem Fall ist die Angabe von currentColor für den Rahmen höchstwahrscheinlich redundant, da durch das Auslassen identische Ergebnisse erzielt werden. Verwenden Sie currentColor nur innerhalb der border-Eigenschaft innerhalb desselben Elements, wenn es ansonsten durch einen spezifischeren Selektor überschrieben würde.

Da es sich um die berechnete Farbe handelt, ist der Rand im folgenden Beispiel grün, da die zweite Regel die erste Regel überschreibt:

```
div {
   color: blue;
   border: 3px solid currentColor;
   color: green;
}
```

Vom übergeordneten Element geerbt

Die Farbe des übergeordneten Elements wird vererbt. Hier wird currentColor mit 'blue' bewertet, sodass die Rahmenfarbe des untergeordneten Elements blau ist.

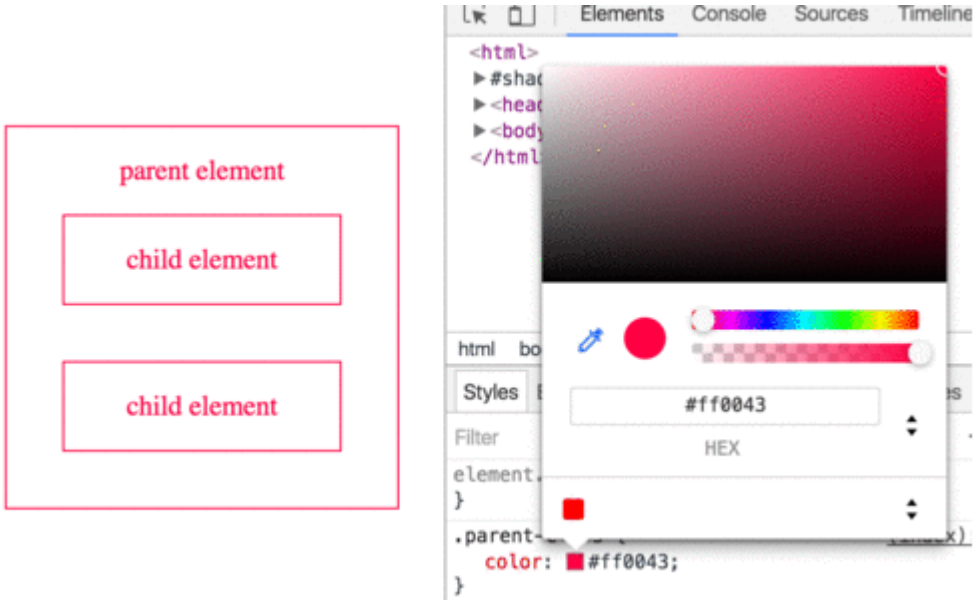
```
.parent-class {
```

```
    color: blue;
}

.parent-class .child-class {
    border-color: currentColor;
}
```

currentColor kann auch von anderen Regeln verwendet werden, die normalerweise nicht von der color -Eigenschaft erben würden, wie z. B. Hintergrundfarbe. Das folgende Beispiel zeigt die untergeordneten Elemente, deren Hintergrundfarbe die übergeordnete Farbe ist:

```
.parent-class {
    color: blue;
}

.parent-class .child-class {
    background-color: currentColor;
}
```

Mögliches Ergebnis:



rgba () Notation

Ähnlich der rgb () - Notation , jedoch mit einem zusätzlichen Alpha-Wert (Opazität).

```
.red {
    /* Opaque red */
    color: rgba(255, 0, 0, 1);
}

.red-50p {
    /* Half-translucent red. */
    color: rgba(255, 0, 0, .5);
}
```

Syntax

```
rgba(<red>, <green>, <blue>, <alpha>);
```

Wert	Beschreibung
`<red>`	eine ganze Zahl von 0 - 255 oder Prozentsatz von 0 - 100%
`<green>`	eine ganze Zahl von 0 - 255 oder Prozentsatz von 0 - 100%
`<blue>`	eine ganze Zahl von 0 - 255 oder Prozentsatz von 0 - 100%
`<alpha>`	eine Zahl von 0 - 1, wobei 0,0 vollständig transparent und 1,0 vollständig undurchsichtig ist

hsla () Notation

Ähnlich der hsl () -Notation , jedoch mit zusätzlichem Alpha-Wert (Opazität).

```
hsla(240, 100%, 50%, 0)     /* transparent */
hsla(240, 100%, 50%, 0.5)   /* half-translucent blue */
hsla(240, 100%, 50%, 1)     /* fully opaque blue */
```

Syntax

```
hsla(<hue>, <saturation>%, <lightness>%, <alpha>);
```

Wert	Beschreibung
`<hue>`	um den Farbkreis herum (ohne Einheiten) angegeben, wobei 0 ° rot, 60 ° gelb, 120 ° grün, 180 ° cyan, 240 ° blau, 300 ° magenta und 360 ° rot sind
`<saturation>`	Prozentsatz, wenn 0% vollständig entsättigt ist (Graustufen) und 100% vollständig gesättigt (kräftig gefärbt)
`<lightness>`	Prozentsatz, wobei 0% vollständig schwarz und 100% vollständig weiß sind
`<alpha>`	eine Zahl von 0 - 1, wobei 0 vollständig transparent und 1 vollständig undurchsichtig ist

Farben online lesen: https://riptutorial.com/de/css/topic/644/farben

Kapitel 21: Flexibles Boxenlayout (Flexbox)

Einführung

Das Flexible Box-Modul oder kurz „Flexbox“ ist ein für Benutzeroberflächen entwickeltes Boxmodell. Mit diesem können Benutzer den Platz zwischen Elementen in einem Container so ausrichten und verteilen, dass sich Elemente vorhersehbar verhalten, wenn das Seitenlayout unterschiedliche, unbekannte Elemente berücksichtigen muss Bildschirmgrößen. Ein Flex-Container erweitert Elemente, um den verfügbaren Platz zu füllen, und verkleinert sie, um ein Überlaufen zu verhindern.

Syntax

- Anzeige: Flex;
- Flex-Richtung: Reihe | Zeilenumkehrung | Spalte | Spaltenumkehrung;
- Flex-Wrap: Nowrap | wickeln | Wrap-Reverse;
- Flex-Flow: <'Flex-Richtung'> || <'flex-wrap'>
- Inhalt rechtfertigen: Flex-Start | Flex-Ende | Zentrum | Leerzeichen | Raum um;
- Ausrichtungsartikel: Flex-Start | Flex-Ende | Zentrum | Grundlinie | strecken;
- Inhalt ausrichten: Flex-Start | Flex-Ende | Zentrum | Leerzeichen | Raum um | strecken;
- Reihenfolge: <Ganzzahl>;
- Flex-Grow: <Nummer>; / * default 0 * /
- Flex-Shrink: <Nummer>; / * default 1 * /
- Flex-Basis: <Länge> | Auto; / * default auto * /
- flex: keine | [<'flex-grow'> <'flex-shrink'>? || <'flex-basis'>]
- align-self: auto | Flex-Start | Flex-Ende | Zentrum | Grundlinie | strecken;

Bemerkungen

Vender-Präfixe

- Anzeige: -webkit-box; / * Chrome <20 * /
- Anzeige: -webkit-flex; / * Chrome 20+ * /
- Anzeige: -Moz-Box; /* Feuerfuchs */
- Anzeige: -ms-flexbox; / * IE * /
- Anzeige: Flex; / * Moderne Browser * /

Ressourcen

- Ein vollständiges Handbuch zu Flexbox
- Gelöst von Flexbox
- Was ist die Flexbox ?!

- Flexbox in 5 Minuten
- Flexbugs

Examples

Klebrige Fußzeile mit variabler Höhe

Dieser Code erstellt eine klebrige Fußzeile. Wenn der Inhalt das Ende des Ansichtsfensters nicht erreicht, bleibt die Fußzeile am unteren Rand des Ansichtsfensters. Wenn der Inhalt über den unteren Rand des Ansichtsfensters hinausragt, wird auch die Fußzeile aus dem Ansichtsfenster verschoben. *Zeige Ergebnis*

HTML:

```
<div class="header">
  <h2>Header</h2>
</div>

<div class="content">
  <h1>Content</h1>
  <p>Lorem ipsum dolor sit amet, consectetur adipiscing elit. Integer nec odio. Praesent
libero. Sed cursus ante dapibus diam. Sed nisi. Nulla quis sem at nibh elementum imperdiet.
Duis sagittis ipsum. Praesent mauris. Fusce nec tellus sed augue semper porta. Mauris massa.
Vestibulum lacinia arcu eget nulla. Class aptent taciti sociosqu ad litora torquent per
conubia nostra, per inceptos himenaeos. Curabitur sodales ligula in libero. </p>
</div>

<div class="footer">
  <h4>Footer</h4>
</div>
```

CSS:

```
html, body {
  height: 100%;
}

body {
  display: flex;
  flex-direction: column;
}

.content {
  /* Include `0 auto` for best browser compatibility. */
  flex: 1 0 auto;
}

.header, .footer {
  background-color: grey;
  color: white;
  flex: none;
}
```

Holy Grail Layout mit Flexbox

Holy Grail Layout ist ein Layout mit einer festen Höhe Kopf und Fußzeile und einem Zentrum mit 3 Spalten. Die 3 Spalten enthalten ein Sidenav mit fester Breite, ein Fluidzentrum und eine Spalte für andere Inhalte wie Anzeigen (das Fluidzentrum erscheint zuerst im Markup). Mit CSS Flexbox kann dies mit einem sehr einfachen Markup erreicht werden:

HTML-Markup:

```
<div class="container">
  <header class="header">Header</header>
  <div class="content-body">
    <main class="content">Content</main>
    <nav class="sidenav">Nav</nav>
    <aside class="ads">Ads</aside>
  </div>
  <footer class="footer">Footer</footer>
</div>
```

CSS:

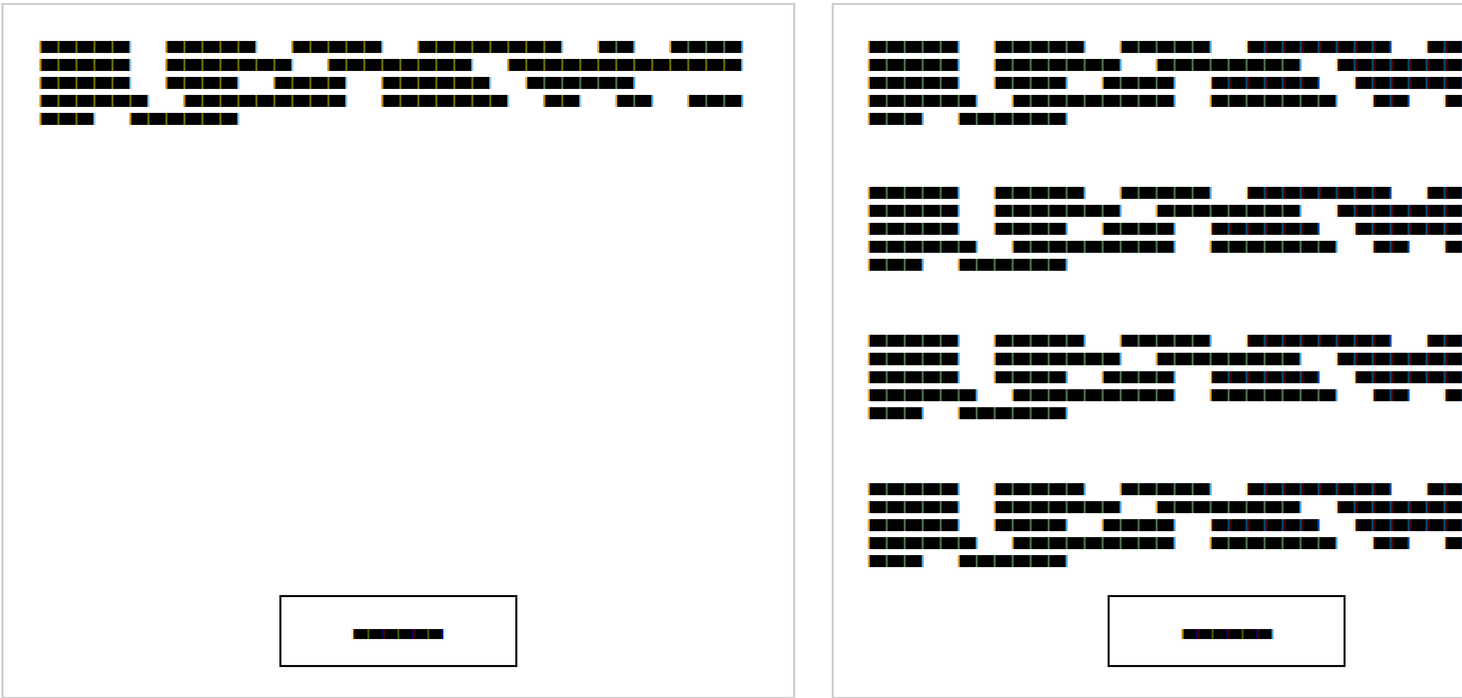
```
body {
  margin: 0;
  padding: 0;
}

.container {
  display: flex;
  flex-direction: column;
  height: 100vh;
}

.header {
  flex: 0 0 50px;
}

.content-body {
  flex: 1 1 auto;

  display: flex;
  flex-direction: row;
}

.content-body .content {
  flex: 1 1 auto;
  overflow: auto;
}

.content-body .sidenav {
  order: -1;
  flex: 0 0 100px;
  overflow: auto;
}

.content-body .ads {
  flex: 0 0 100px;
  overflow: auto;
}

.footer {
```

```
  flex: 0 0 50px;
}
```

Demo

Perfekt abgestimmte Knöpfe in Karten mit Flexbox

Heutzutage ist es ein regelmäßiges Muster im Design, die Aufrufe **der Aktionen** in den darin enthaltenen Karten vertikal wie folgt auszurichten:



Dies kann durch einen speziellen Trick mit der `flexbox`

HTML

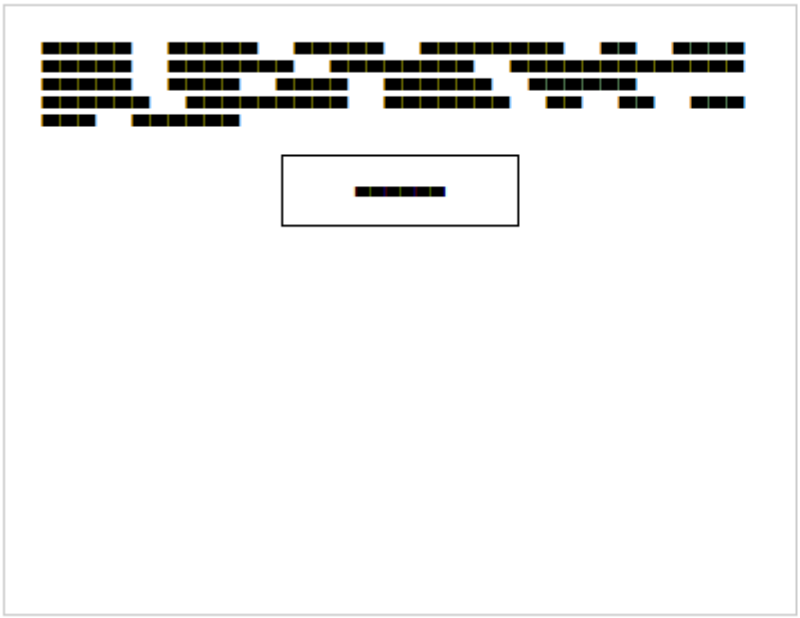
```
<div class="cards">
  <div class="card">
    <p>Lorem ipsum Magna proident ex anim dolor ullamco pariatur reprehenderit culpa esse enim
mollit labore dolore voluptate ullamco et ut sed qui minim.</p>
    <p><button>Action</button></p>
  </div>
  <div class="card">
    <p>Lorem ipsum Magna proident ex anim dolor ullamco pariatur reprehenderit culpa esse enim
mollit labore dolore voluptate ullamco et ut sed qui minim.</p>
    <p>Lorem ipsum Magna proident ex anim dolor ullamco pariatur reprehenderit culpa esse enim
mollit labore dolore voluptate ullamco et ut sed qui minim.</p>
    <p>Lorem ipsum Magna proident ex anim dolor ullamco pariatur reprehenderit culpa esse enim
mollit labore dolore voluptate ullamco et ut sed qui minim.</p>
    <p>Lorem ipsum Magna proident ex anim dolor ullamco pariatur reprehenderit culpa esse enim
mollit labore dolore voluptate ullamco et ut sed qui minim.</p>
    <p><button>Action</button></p>
  </div>
</div>
```

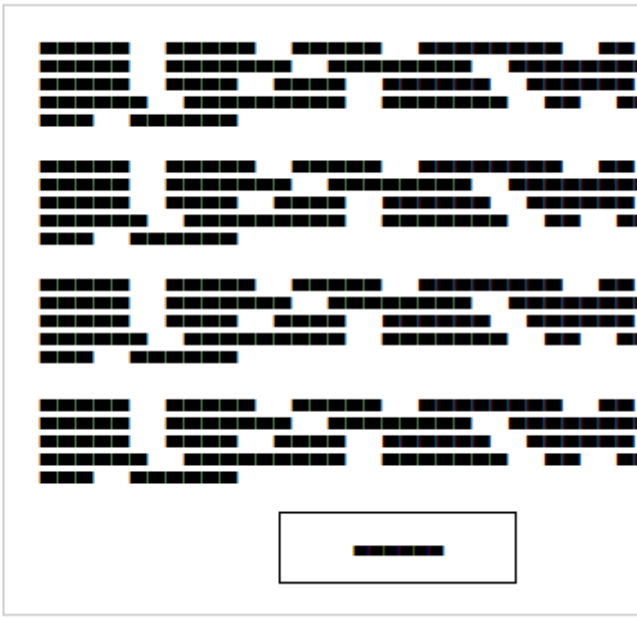
Zuerst verwenden wir CSS, um die `display: flex;` anzuwenden `display: flex;` zum Behälter. Dadurch werden 2 Spalten mit gleicher Höhe erstellt, in denen der Inhalt natürlich fließt

CSS

```
.cards {
    display: flex;
}
.card {
    border: 1px solid #ccc;
    margin: 10px 10px;
    padding: 0 20px;
}
button {
    height: 40px;
    background: #fff;
    padding: 0 40px;
    border: 1px solid #000;
}
p:last-child {
    text-align: center;
}
```

Das Layout ändert sich und wird so:





Um die Schaltflächen an den unteren Rand des Blocks zu verschieben, müssen Sie die `display: flex;` anwenden `display: flex;` auf die Karte selbst, wobei die Richtung auf `column` . Danach sollten wir das letzte Element in der Karte auswählen und den `margin-top` auf `auto` . Dadurch wird der letzte Absatz an den unteren Rand der Karte geschoben und das gewünschte Ergebnis erzielt.

Finales CSS:

```
.cards {
    display: flex;
}
```

```
.card {
    border: 1px solid #ccc;
    margin: 10px 10px;
    padding: 0 20px;
    display: flex;
    flex-direction: column;
}
button {
    height: 40px;
    background: #fff;
    padding: 0 40px;
    border: 1px solid #000;
}
p:last-child {
    text-align: center;
    margin-top: auto;
}
```

Dynamische vertikale und horizontale Zentrierung (Elemente ausrichten, Inhalt ausrichten)

Einfaches Beispiel (Zentrieren eines einzelnen Elements)

HTML

```
<div class="aligner">
  <div class="aligner-item">…</div>
</div>
```

CSS

```
.aligner {
  display: flex;
  align-items: center;
  justify-content: center;
}

.aligner-item {
  max-width: 50%; /*for demo. Use actual width instead.*/
}
```

Hier ist eine Demo .

Argumentation

Eigentum	Wert	Beschreibung
`align-items`	`center`	Dies zentriert die Elemente entlang der Achse andere als der angegebenen einer nach `flex-direction` , das heißt, vertikale Zentrierung für eine horizontale FlexBox und horizontale Zentrieren für eine vertikale FlexBox.
`justify-content`	`center`	Dies zentriert die Elemente entlang der durch die `flex-direction` angegebenen Achse. Dh für eine horizontale Flexbox (`flex-direction: row`) zentriert diese horizontal und für eine vertikale Flexbox (`flex-direction: column`) Flexbox vertikal.

Beispiele für individuelle Eigenschaften

Alle folgenden Stile werden auf dieses einfache Layout angewendet:

```
<div id="container">
  <div></div>
  <div></div>
  <div></div>
</div>
```

wo `#container` ist die `flex-box` .

Beispiel: `justify-content: center` Zentriert auf einer horizontalen Flexbox

CSS:

```
div#container {
  display: flex;
  flex-direction: row;
  justify-content: center;
}
```

Ergebnis:

Hier ist eine Demo .

Beispiel: `justify-content: center` Zentriert auf einer vertikalen Flexbox

CSS:

```
div#container {
  display: flex;
  flex-direction: column;
  justify-content: center;
}
```

Ergebnis:

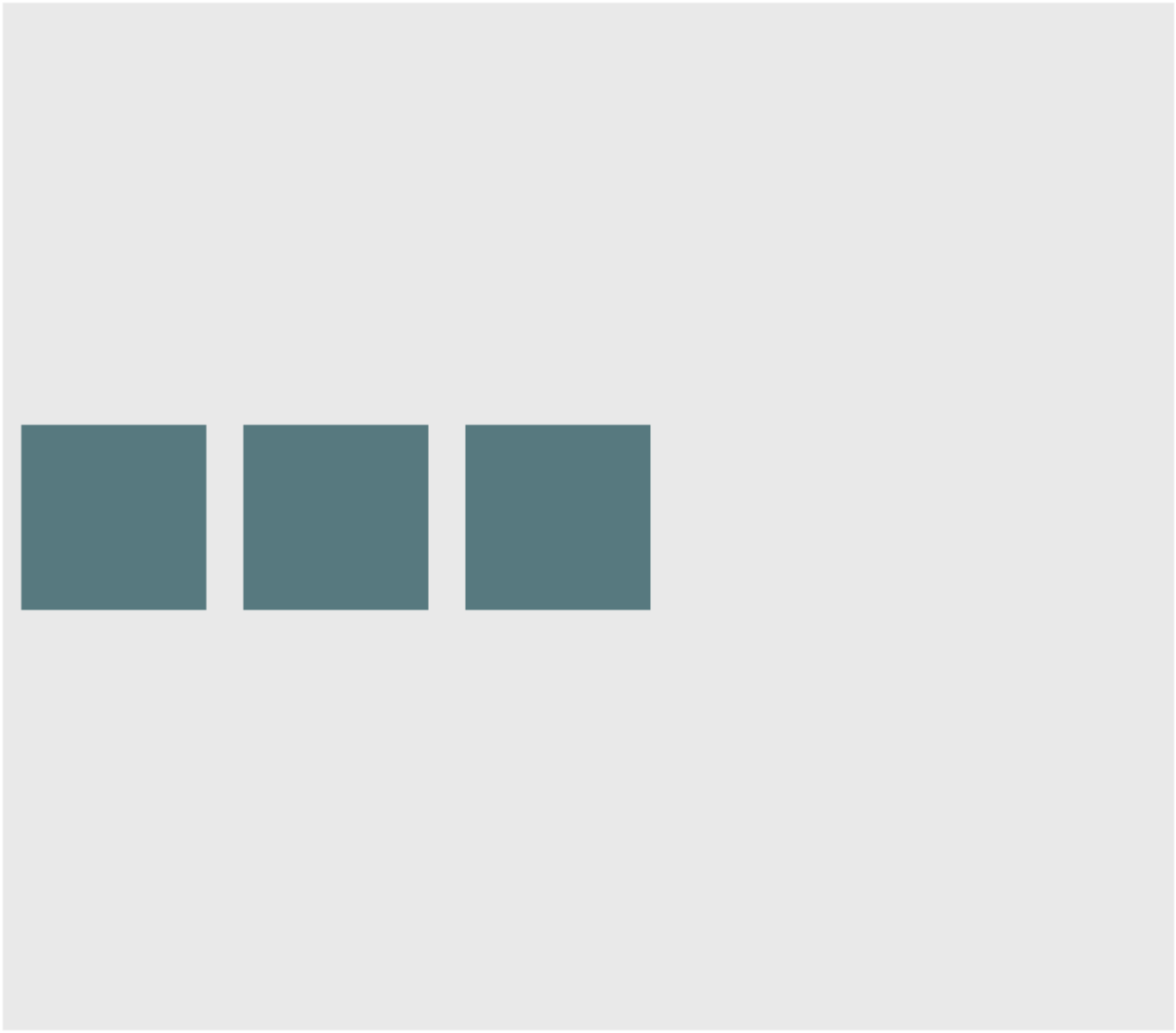
Hier ist eine Demo .

Beispiel: `align-content: center` einer horizontalen Flexbox zentrieren

CSS:

```
div#container {
  display: flex;
  flex-direction: row;
  align-items: center;
}
```

Ergebnis:



Hier ist eine Demo .

Beispiel: `align-content: center` Zentriert auf einer vertikalen Flexbox

CSS:

```
div#container {
  display: flex;
  flex-direction: column;
  align-items: center;
}
```

Ergebnis:

Hier ist eine Demo .

Beispiel: Kombination zum Zentrieren von beiden auf der horizontalen Flexbox

```
div#container {
  display: flex;
  flex-direction: row;
  justify-content: center;
  align-items: center;
}
```

Ergebnis:

Hier ist eine Demo .

Beispiel: Kombination zum Zentrieren beider auf vertikaler Flexbox

```
div#container {
  display: flex;
  flex-direction: column;
  justify-content: center;
  align-items: center;
}
```

Ergebnis:

Hier ist eine Demo .

Gleiche Höhe bei verschachtelten Containern

Dieser Code stellt sicher, dass alle geschachtelten Container immer die gleiche Höhe haben. Dies geschieht, indem sichergestellt wird, dass alle geschachtelten Elemente dieselbe Höhe wie das enthaltende parrent div haben. Siehe Arbeitsbeispiel : https://jsfiddle.net/3wwh7ewp/

Dieser Effekt wird dadurch erreicht, dass die Eigenschaft `align-items` standardmäßig auf `stretch` wird.

HTML

```
<div class="container">
  <div style="background-color: red">
    Some <br />
    data <br />
    to make<br />
    a height <br />
 </div>
```

```
  <div style="background-color: blue">
    Fewer <br />
    lines <br />
 </div>
</div>
```

CSS

```
.container {
    display: flex;
    align-items: stretch; // Default value
}
```

Hinweis: Funktioniert nicht bei IE-Versionen unter 10

Passen Sie Elemente optimal an ihren Behälter an

Eine der schönsten Eigenschaften von flexbox ist die optimale Anpassung von Behältern an das übergeordnete Element.

Live-Demo

HTML:

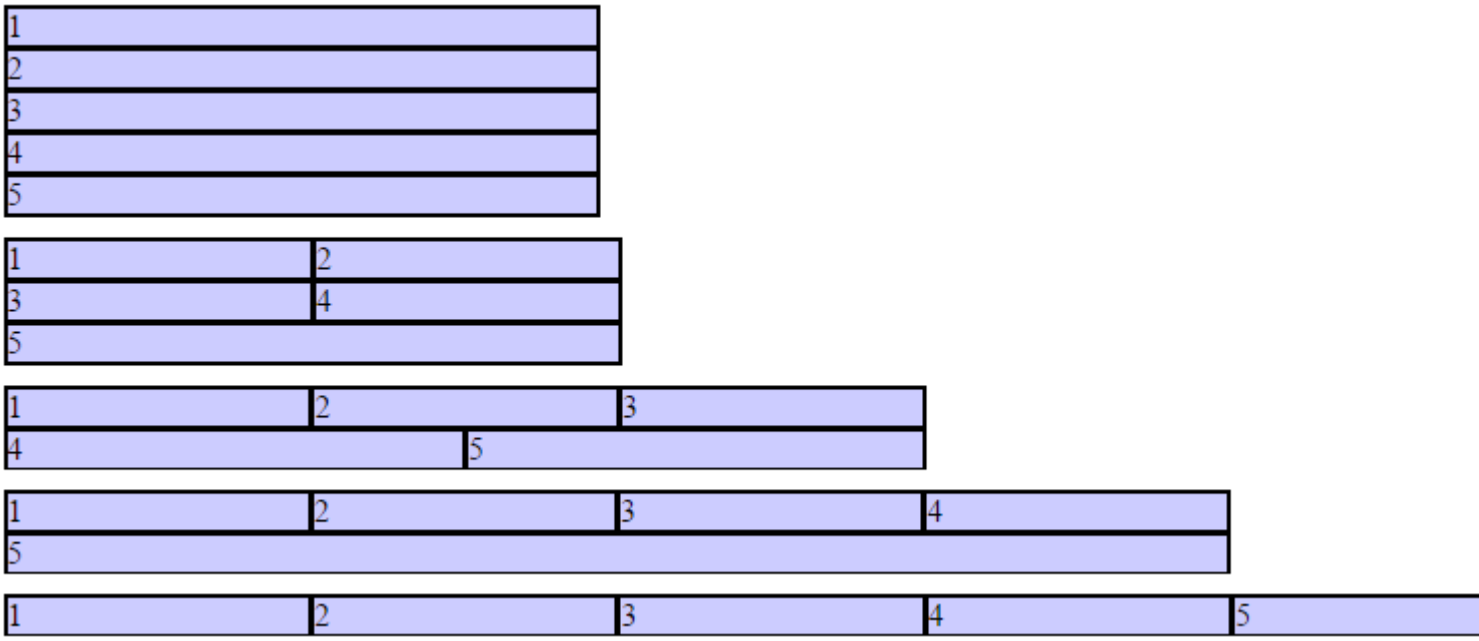
```
<div class="flex-container">
  <div class="flex-item">1</div>
  <div class="flex-item">2</div>
  <div class="flex-item">3</div>
  <div class="flex-item">4</div>
  <div class="flex-item">5</div>
</div>
```

CSS:

```
.flex-container {
    background-color: #000;
    height: 100%;
    display:flex;
    flex-direction: row;
    flex-wrap: wrap;
    justify-content: flex-start;
    align-content: stretch;
    align-items: stretch;
}

.flex-item {
    background-color: #ccf;
    margin: 0.1em;
    flex-grow: 1;
    flex-shrink: 0;
    flex-basis: 200px; /* or % could be used to ensure a specific layout */
}
```

Ergebnis:

Spalten passen sich an, wenn die Bildschirmgröße geändert wird.



Flexibles Boxenlayout (Flexbox) online lesen: https://riptutorial.com/de/css/topic/445/flexibles-boxenlayout--flexbox-

Kapitel 22: Formen für Schwimmer

Syntax

- Form außerhalb: keine | [<Grundform> || <Formfeld>] | <image>
- Formrand: <Länge> | <Prozentsatz>
- Form-Bild-Schwelle: <Anzahl>

Parameter

Parameter	Einzelheiten
keiner	Der Wert `none` bedeutet, dass der Float-Bereich (der Bereich, in dem Inhalt um ein Float-Element gewickelt wird) nicht betroffen ist. Dies ist der Standardwert / Anfangswert.
Grundform	Bezieht sich auf eins zwischen `inset()` , `circle()` , `ellipse()` oder `polygon()` . Mit einer dieser Funktionen und ihren Werten wird die Form definiert.
Formkasten	Bezieht sich auf eine unter `margin-box` , `border-box` , `padding-box` , `content-box` . Wenn nur <Formfeld> (ohne <Grundform>) bereitgestellt wird, ist dieses Feld *die* Form. Wenn es zusammen mit der <Grundform> verwendet wird, dient dies als Referenzfeld.
Bild	Wenn ein Bild als Wert bereitgestellt wird, wird die Form basierend auf dem Alphakanal des angegebenen Bildes berechnet.

Bemerkungen

Die Browser-Unterstützung für das CSS Shapes-Modul ist zu diesem Zeitpunkt sehr begrenzt.

Es wird in Chrome v37 + und Opera 24+ ohne Browser- / Herstellerpräfixe unterstützt. Safari unterstützt es ab Version 7.1, jedoch mit dem Präfix `-webkit-` .

Es wird noch nicht in IE, Edge und Firefox unterstützt.

Examples

Außenform mit Grundform - Kreis ()

Mit der `shape-outside` CSS-Eigenschaft können Sie Formwerte für den Float-Bereich definieren, sodass der Inline-Inhalt anstelle der Box des Float-Objekts um die Form verläuft.

CSS

```
img:nth-of-type(1) {
  shape-outside: circle(80px at 50% 50%);
  float: left;
  width: 200px;
}
img:nth-of-type(2) {
  shape-outside: circle(80px at 50% 50%);
  float: right;
  width: 200px;
}
p {
  text-align: center;
  line-height: 30px; /* purely for demo */
}
```

HTML

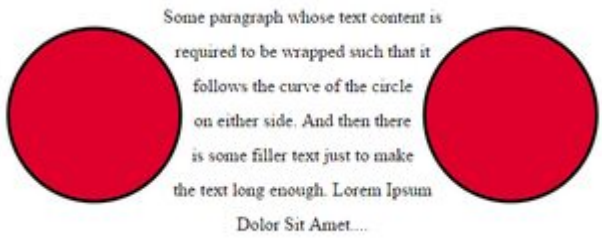
```
<img src="http://images.clipartpanda.com/circle-clip-art-circlergb.jpg">
<img src="http://images.clipartpanda.com/circle-clip-art-circlergb.jpg">
<p>Some paragraph whose text content is required to be wrapped such that it follows the curve
of the circle on either side. And then there is some filler text just to make the text long
enough. Lorem Ipsum Dolor Sit Amet....</p>
```

Im obigen Beispiel handelt es sich bei beiden Bildern eigentlich um quadratische Bilder. Wenn der Text ohne die Eigenschaft `shape-outside` der `shape-outside` platziert wird, wird er auf beiden Seiten nicht um den Kreis herumfließen. Es fließt nur um die umschließende Box des Bildes. Mit `shape-outside` der Float-Bereich als *Kreis* definiert und der Inhalt wird um diesen *imaginären Kreis* , der mit `shape-outside` erstellt wird, `shape-outside` .

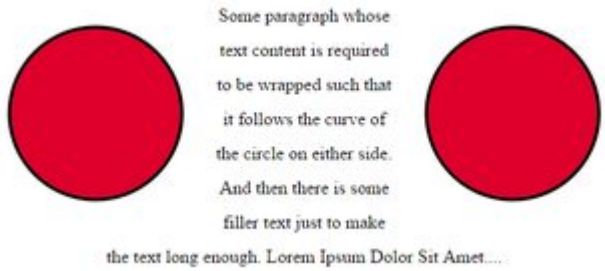
Der *imaginäre Kreis* , der zum Neudefinieren des Floatbereichs verwendet wird, ist ein Kreis mit einem Radius von 80px, der vom Mittelpunkt des Referenzfelds des Bilds gezeichnet wird.

Nachfolgend finden Sie einige Screenshots, die veranschaulichen, wie der Inhalt umgebogen wird, wenn `shape-outside` verwendet wird und wenn er nicht verwendet wird.

Ausgabe mit `shape-outside`



Ausgabe ohne `shape-outside`



Formrand

Die `shape-margin` CSS - Eigenschaft fügt eine *Marge* `shape-outside` .

CSS

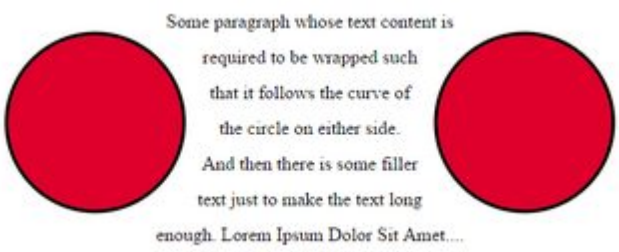
```
img:nth-of-type(1) {
  shape-outside: circle(80px at 50% 50%);
  shape-margin: 10px;
  float: left;
  width: 200px;
}
img:nth-of-type(2) {
  shape-outside: circle(80px at 50% 50%);
  shape-margin: 10px;
  float: right;
  width: 200px;
}
p {
  text-align: center;
  line-height: 30px; /* purely for demo */
}
```

HTML

```
<img src="http://images.clipartpanda.com/circle-clip-art-circlergb.jpg">
<img src="http://images.clipartpanda.com/circle-clip-art-circlergb.jpg">
<p>Some paragraph whose text content is required to be wrapped such that it follows the curve
of the circle on either side. And then there is some filler text just to make the text long
enough. Lorem Ipsum Dolor Sit Amet....</p>
```

In diesem Beispiel wird ein 10px Rand um die **Form** unter Verwendung hinzugefügt `shape-margin` . Dies schafft etwas mehr Platz zwischen dem *imaginären Kreis* , der den Floatbereich definiert, und dem tatsächlichen Inhalt, der herumfließt.

Ausgabe:



Formen für Schwimmer online lesen: https://riptutorial.com/de/css/topic/2034/formen-fur-schwimmer

Kapitel 23: Funktionen

Syntax

- `<calc()> = calc( <calc-sum> )`
- `<calc-sum> = <calc-product> [ [ '+' | '-' ] <calc-product> ]*`
- `<calc-product> = <calc-value> [ '*' <calc-value> | '/' <number> ]*`
- `<calc-value> = <number> | <dimension> | <percentage> | ( <calc-sum> )`

Bemerkungen

Für `calc()` ist um die Operatoren " `-` " und " `+` " ein Leerzeichen erforderlich, nicht jedoch die Operatoren " `*` " oder " `/` ".

Alle Einheiten müssen vom selben Typ sein. Der Versuch, eine Höhe beispielsweise mit einer Zeitdauer zu multiplizieren, ist ungültig.

Examples

calc () - Funktion

Akzeptiert einen mathematischen Ausdruck und gibt einen numerischen Wert zurück.

Dies ist besonders nützlich, wenn Sie mit verschiedenen Einheitentypen arbeiten (z. B. einen px-Wert von einem Prozentwert abziehen), um den Wert eines Attributs zu berechnen.

Alle Operatoren `+` , `-` , `/` und `*` können verwendet werden. Klammern können hinzugefügt werden, um bei Bedarf die Reihenfolge der Operationen festzulegen.

Verwenden Sie `calc()` , um die Breite eines div-Elements zu berechnen:

```
#div1 {
    position: absolute;
    left: 50px;
    width: calc(100% - 100px);
    border: 1px solid black;
    background-color: yellow;
    padding: 5px;
    text-align: center;
}
```

Verwenden Sie `calc()` , um die Position eines Hintergrundbildes zu bestimmen:

```
background-position: calc(50% + 17px) calc(50% + 10px), 50% 50%;
```

Verwenden Sie `calc()` , um die Höhe eines Elements zu bestimmen:

```
height: calc(100% - 20px);
```

Funktion attr ()

Gibt den Wert eines Attributs des ausgewählten Elements zurück.

Nachfolgend finden Sie ein blockquote-Element, das ein Zeichen innerhalb eines `data-*` -Attributs enthält, das CSS (z. B. innerhalb von `::before` und `::after` pseudo-element) mithilfe dieser Funktion verwenden kann.

```
<blockquote data-mark='"'></blockquote>
```

Im folgenden CSS-Block wird das Zeichen vor und nach dem Text im Element angehängt:

```
blockquote[data-mark]::before,
blockquote[data-mark]::after {
    content: attr(data-mark);
}
```

linearer Farbverlauf ()

Erzeugt ein Bild, das einen linearen Farbverlauf darstellt.

```
linear-gradient( 0deg, red, yellow 50%, blue);
```

Dadurch entsteht ein Farbverlauf von unten nach oben, wobei die Farben bei Rot beginnen, dann bei 50% und dann in Blau.

Radialgradient () Funktion

Erzeugt ein Bild, das einen Farbverlauf aus der Mitte des Farbverlaufs darstellt

```
radial-gradient(red, orange, yellow) /*A gradient coming out from the middle of the
gradient, red at the center, then orange, until it is finally yellow at the edges*/
```

var () Funktion

Mit der Funktion var () kann auf CSS-Variablen zugegriffen werden.

```
/* set a variable */
:root {
    --primary-color: blue;
}

/* access variable */
selector {
    color: var(--primary-color);
}
```

Diese Funktion befindet sich derzeit in der Entwicklung. Überprüfen Sie caniuse.com auf die neueste Browserunterstützung.

Funktionen online lesen: https://riptutorial.com/de/css/topic/2214/funktionen

Kapitel 24: Funktionsabfragen

Syntax

- @supports [Bedingung] {/ * CSS-Regeln zum Anwenden von * /}

Parameter

Parameter	Einzelheiten
`(property: value)`	Wertet true aus, wenn der Browser die CSS-Regel verarbeiten kann. Die Klammern um die Regel sind erforderlich.
`and`	Gibt nur dann true zurück, wenn sowohl die vorherige als auch die nächste Bedingung erfüllt sind.
`not`	Negiert die nächste Bedingung
`or`	Gibt true zurück, wenn entweder die vorherige oder die nächste Bedingung wahr ist.
`(...)`	Gruppen bedingungen

Bemerkungen

Die Funktionserkennung mit `@supports` wird in Edge, Chrome, Firefox, Opera und Safari 9 und `@supports` unterstützt.

Examples

Grundlegende Verwendung von @supports

```
@supports (display: flex) {
  /* Flexbox is available, so use it */
  .my-container {
    display: flex;
  }
}
```

In der Syntax ist `@supports` sehr ähnlich zu `@media` , aber anstelle von Bildschirmgröße und -ausrichtung `@supports` , ob der Browser eine bestimmte CSS-Regel verarbeiten kann.

`@supports (flex)` , dass die Regel nicht `@supports (display: flex)` ist, `@supports (display: flex)` .

Erkennungsfunktion für Verkettungsfunktionen

Verwenden Sie den Operator `and` um mehrere Features gleichzeitig zu erkennen.

```
@supports (transform: translateZ(1px)) and (transform-style: preserve-3d) and (perspective:
1px) {
  /* Probably do some fancy 3d stuff here */
}
```

Es gibt auch einen `or` Operator und einen `not` Operator:

```
@supports (display: flex) or (display: table-cell) {
  /* Will be used if the browser supports flexbox or display: table-cell */
}
@supports not (-webkit-transform: translate(0, 0, 0)) {
  /* Will *not* be used if the browser supports -webkit-transform: translate(...) */
}
```

`@supports` für das ultimative `@supports` Erlebnis, logische Ausdrücke mit Klammern zu gruppieren:

```
@supports ((display: block) and (zoom: 1)) or ((display: flex) and (not (display: table-
cell))) or (transform: translateX(1px)) {
  /* ... */
}
```

Dies funktioniert, wenn der Browser

1. Unterstützt die `display: block` AND `zoom: 1` oder
2. Unterstützt `display: flex` UND NICHT `display: table-cell` oder
3. Unterstützt `transform: translateX(1px)` .

Funktionsabfragen online lesen: https://riptutorial.com/de/css/topic/5024/funktionsabfragen

Kapitel 25: Gitter

Einführung

Das Rasterlayout ist ein neues und leistungsfähiges CSS-Layoutsystem, mit dem der Inhalt einer Webseite auf einfache Weise in Zeilen und Spalten unterteilt werden kann.

Bemerkungen

Das CSS-Grid-Layout-Modul Level 1 ist seit dem 9. September 2016 eine W3C-Kandidatenempfehlung. Es befindet sich in der Testphase (https://www.w3.org/Style/CSS/current-work) .

Seit dem 3. Juli 2017 unterstützen die Browser Microsoft Internet Explorer 10 und 11 und Edge nur eine ältere Version der Spezifikation, die ein Herstellerpräfix verwendet.

Examples

Basisbeispiel

Eigentum	Mögliche Werte
Anzeige	Raster / Inline-Raster

Das CSS-Grid ist als Anzeigeeigenschaft definiert. Es gilt nur für ein übergeordnetes Element und dessen unmittelbar untergeordnete Elemente.

Betrachten Sie das folgende Markup:

```
<section class="container">
  <div class="item1">item1</div>
  <div class="item2">item2</div>
  <div class="item3">item3</div>
  <div class="item4">item4</div>
</section>
```

Die einfachste Möglichkeit, die Markup-Struktur oben als Gitter zu definieren, besteht darin, die `display` einfach auf `grid` :

```
.container {
  display: grid;
}
```

Dies führt jedoch immer dazu, dass alle untergeordneten Elemente übereinander kollabieren. Dies liegt daran, dass die Kinder derzeit nicht wissen, wie sie sich innerhalb des Gitters positionieren sollen. Aber wir können es ihnen explizit sagen.

Zuerst müssen wir das Gitterelement sagen `.container` , wie viele Zeilen und Spalten wird seine Struktur bilden und wir können dies die Verwendung tun `grid-columns` und `grid-rows` Eigenschaften (man beachte die Pluralisierung):

```
.container {
  display: grid;
  grid-columns: 50px 50px 50px;
  grid-rows: 50px 50px;
}
```

Das hilft uns aber immer noch nicht viel, da wir jedem untergeordneten Element eine Reihenfolge geben müssen. Wir können dies tun, indem wir die Werte der `grid-row` und `grid-column` , die angeben, wo sie sich im Raster befinden:

```
.container .item1 {
  grid-column: 1;
  grid-row: 1;
}
.container .item2 {
  grid-column: 2;
  grid-row: 1;
}
.container .item3 {
  grid-column: 1;
  grid-row: 2;
}
.container .item4 {
  grid-column: 2;
  grid-row: 2;
}
```

Indem jedem Artikel ein Spalten- und Zeilenwert zugewiesen wird, wird die Artikelreihenfolge innerhalb des Containers angegeben.

Sehen Sie sich ein Arbeitsbeispiel auf JSFiddle an . Sie müssen dies in IE10, IE11 oder Edge anzeigen, damit es funktioniert, da dies derzeit die einzigen Browser sind, die Grid Layout (mit dem Herstellerpräfix `-ms-`) unterstützen, oder ein Flag in Chrome, Opera und Firefox entsprechend der caniuse aktivieren mit ihnen testen.

Gitter online lesen: https://riptutorial.com/de/css/topic/2152/gitter

Kapitel 26: Hintergründe

Einführung

Mit CSS können Sie Farben, Farbverläufe und Bilder als Hintergrund eines Elements festlegen.

Es ist möglich, verschiedene Kombinationen von Bildern, Farben und Verläufen festzulegen und deren Größe, Positionierung und Wiederholung (unter anderem) anzupassen.

Syntax

- Hintergrundfarbe: Farbe | transparent | initial | erben;
- Hintergrundbild: URL | Keine | initial | erben;
- Hintergrundposition: Wert;
- Hintergrundgröße: <bg-size> [<bg-size>]
- <bg-size>: auto | Länge | Abdeckung | enthalten | initial | erben;
- Hintergrundwiederholung: Wiederholung | Wiederholung-x | wiederholen-y | keine Wiederholung | initial | erben;
- Hintergrundursprung: Polsterbox | Randfeld | Inhaltsfeld | initial | erben;
- Hintergrund-Clip: Border-Box | Polsterbox | Inhaltsfeld | initial | erben;
- Hintergrundanhang: scrollen | fixiert | lokal | initial | erben;
- Hintergrund: bg-color bg-Bildposition / bg-Größe bg-Wiederholung bg-ursprung bg-clip bg-anhang initial | erben;

Bemerkungen

- CSS3-Farbverläufe funktionieren nicht für Versionen von Internet Explorer, die weniger als 10 sind.

Examples

Hintergrundfarbe

Die `background-color` Eigenschaft setzt die Hintergrundfarbe eines Elements eines Farbwertes oder durch Schlüsselwörter, wie die Verwendung von `transparent` , `inherit` oder `initial` .

- **transparent** : Gibt an, dass die Hintergrundfarbe transparent sein soll. Dies ist die Standardeinstellung.
- **erben** , erbt diese Eigenschaft von ihrem übergeordneten Element.
- **initial** setzt diese Eigenschaft auf ihren Standardwert.

Dies kann auf alle Elemente und `::first-letter` / `::first-line` Pseudo-Elemente angewendet werden .

Farben in CSS können mit verschiedenen Methoden angegeben werden .

Farbnamen

CSS

```
div {
  background-color: red;  /* red */
}
```

HTML

```
<div>This will have a red background</div>
```

- Das oben verwendete Beispiel ist eine von mehreren Möglichkeiten, wie CSS eine einzelne Farbe darstellen muss.

Hex-Farbcodes

Der Hex-Code wird verwendet, um RGB-Komponenten einer Farbe in der Hexadezimal-Notation 16 anzugeben. # ff0000 ist beispielsweise hellrot, wobei der Rotanteil der Farbe 256 Bit (ff) beträgt und der entsprechende Grün- und Blauanteil der Farbe 0 (00) ist.

Wenn beide Werte in jeder der drei RGB-Paarungen (R, G und B) gleich sind, kann der Farbcode auf drei Zeichen verkürzt werden (die erste Ziffer jeder Paarung). `#ff0000` kann auf `#f00` und `#ffffff` auf `#fff` verkürzt `#fff` .

Die Hex-Schreibweise unterscheidet nicht zwischen Groß- und Kleinschreibung.

```
body {
  background-color: #de1205; /* red */
}

.main {
  background-color: #00f; /* blue */
}
```

RGB / RGBa

Eine andere Möglichkeit, eine Farbe zu deklarieren, ist die Verwendung von RGB oder RGBa.

RGB steht für Rot, Grün und Blau und erfordert drei separate Werte zwischen 0 und 255 (in Klammern gesetzt), die den Dezimalfarbwerten für Rot, Grün und Blau entsprechen.

Mit RGBa können Sie einen zusätzlichen Alpha-Parameter zwischen 0,0 und 1,0 hinzufügen, um die Deckkraft zu definieren.

```
header {
  background-color: rgb(0, 0, 0); /* black */
}

footer {
  background-color: rgba(0, 0, 0, 0.5); /* black with 50% opacity */
}
```

HSL / HSLa

Eine andere Möglichkeit, eine Farbe zu deklarieren, ist die Verwendung von HSL oder HSLa und ähnelt RGB und RGBa.

HSL steht für Farbton, Sättigung und Helligkeit und wird häufig auch als HLS bezeichnet:

- Der Farbton ist ein Grad des Farbkreises (von 0 bis 360).
- Die Sättigung beträgt einen Prozentsatz zwischen 0% und 100%.
- Die Helligkeit ist auch ein Prozentsatz zwischen 0% und 100%.

Mit HSLa können Sie einen zusätzlichen Alpha-Parameter zwischen 0,0 und 1,0 hinzufügen, um die Deckkraft zu definieren.

```
li a {
  background-color: hsl(120, 100%, 50%); /* green */
}

#p1 {
  background-color: hsla(120, 100%, 50%, .3); /* green with 30% opacity */
}
```

Interaktion mit Hintergrundbild

Die folgenden Aussagen sind alle gleichwertig:

```
body {
  background: red;
  background-image: url(partiallytransparentimage.png);
}

body {
  background-color: red;
  background-image: url(partiallytransparentimage.png);
}

body {
  background-image: url(partiallytransparentimage.png);
  background-color: red;
}

body {
  background: red url(partiallytransparentimage.png);
}
```

Sie alle führen dazu, dass die rote Farbe unter dem Bild angezeigt wird, wenn die Bildteile transparent sind oder das Bild nicht angezeigt wird (möglicherweise als Folge einer `background-repeat`).

Beachten Sie, dass Folgendes nicht gleichwertig ist:

```
body {
  background-image: url(partiallytransparentimage.png);
  background: red;
}
```

Hier setzt der Wert des `background` Ihr `background-image` außer Kraft.

Weitere Informationen zur `background` finden Sie unter Background Shorthand

Hintergrundbild

Die `background-image` Eigenschaft wird verwendet, um ein Hintergrundbild anzugeben, das auf alle übereinstimmenden Elemente angewendet werden soll. Standardmäßig ist dieses Bild so angeordnet, dass es das gesamte Element mit Ausnahme des Randes abdeckt.

```
.myClass {
  background-image: url('/path/to/image.jpg');
}
```

Um mehrere Bilder als `background-image` , definieren Sie eine durch Kommas getrennte `url()`

```
.myClass {
  background-image: url('/path/to/image.jpg'),
                    url('/path/to/image2.jpg');
}
```

Die Bilder werden entsprechend ihrer Reihenfolge gestapelt, wobei sich das erste deklarierte Bild auf den anderen befindet und so weiter.

Wert	Ergebnis
`url('/path/to/image.jpg')`	Geben Sie den Pfad bzw. die Pfade des Hintergrundbilds oder eine mit dem URI-Schema für Daten festgelegte Bildressource an (Apostrophe können weggelassen werden), mehrere durch Kommas getrennte Vielfache
`none`	Kein Hintergrundbild
`initial`	Standardwert
`inherit`	Erbt den Wert des Elternteils

Mehr CSS für Hintergrundbild

Diese folgenden Attribute sind sehr nützlich und auch fast unverzichtbar.

```
background-size:     xpx ypx | x% y%;
background-repeat:   no-repeat | repeat | repeat-x | repeat-y;
background-position: left offset (px/%) right offset (px/%) | center center | left top | right
bottom;
```

Hintergrundsteigungen

Farbverläufe sind neue Bildtypen, die in CSS3 hinzugefügt werden. Als Bild werden Farbverläufe mit der `background-image` Eigenschaft oder der `background` Abkürzung festgelegt.

Es gibt zwei Arten von Gradientenfunktionen, linear und radial. Jeder Typ hat eine sich nicht wiederholende Variante und eine sich wiederholende Variante:

- `linear-gradient()`
- `repeating-linear-gradient()`
- `radial-gradient()`
- `repeating-radial-gradient()`

linearer Gradient ()

Ein `linear-gradient` hat die folgende Syntax

```
background: linear-gradient( <direction>?, <color-stop-1>, <color-stop-2>, ...);
```

Wert	Bedeutung
`<direction>`	Könnte ein Argument wie `to top` , `to bottom` , `to right` oder `to left` ; oder ein Winkel wie `0deg` , `90deg` .... Der Winkel beginnt von oben und dreht sich im Uhrzeigersinn. Kann in Grad , Grad , Rad oder Drehung angegeben werden . Wenn weggelassen, fließt der Farbverlauf von oben nach unten
`<color-stop-list>`	Liste der Farben, optional gefolgt von einem Prozentsatz oder einer Länge , um sie anzuzeigen. Zum Beispiel `yellow 10%` , `rgba(0,0,0,.5) 40px` , `#fff 100%` ...

Dies erzeugt beispielsweise einen linearen Farbverlauf, der von rechts beginnt und von Rot nach Blau übergeht

```
.linear-gradient {
  background: linear-gradient(to left, red, blue); /* you can also use 270deg */
}
```

Sie können einen `diagonal` erstellen, indem Sie sowohl eine horizontale als auch eine vertikale Startposition angeben.

```
.diagonal-linear-gradient {
  background: linear-gradient(to left top, red, yellow 10%);
}
```

Sie können eine beliebige Anzahl von Farbstopps in einem Verlauf angeben, indem Sie sie durch Kommas voneinander trennen. In den folgenden Beispielen wird ein Verlauf mit 8 Farbstopps erstellt

```
.linear-gradient-rainbow {
  background: linear-gradient(to left, red, orange, yellow, green, blue, indigo, violet)
}
```

Radialgradient ()

```
.radial-gradient-simple {
  background: radial-gradient(red, blue);
}

.radial-gradient {
  background: radial-gradient(circle farthest-corner at top left, red, blue);
}
```

Wert	Bedeutung
`circle`	Form des Verlaufs Werte sind `circle` oder `ellipse` , Standard ist `ellipse` .
`farthest-corner`	Schlüsselwörter, die beschreiben, wie groß die Endform sein muss. Werte sind am `closest-side` , `farthest-side` , am `closest-corner` , `farthest-corner`
`top left`	Legt die Position des Verlaufszentrums wie die `background-position` .

Farbverläufe wiederholen

Wiederholende Verlaufsfunktionen verwenden dieselben Argumente wie die obigen Beispiele, kippen den Verlauf jedoch über den Hintergrund des Elements.

```
.bullseye {
  background: repeating-radial-gradient(red, red 10%, white 10%, white 20%);
}
.warning {
  background: repeating-linear-gradient(-45deg, yellow, yellow 10%, black 10%, black 20% );
}
```

Wert	Bedeutung
`-45deg`	Winkeleinheit . Der Winkel beginnt von oben und dreht sich im Uhrzeigersinn.

Wert	Bedeutung
	Kann in Grad , Grad , Rad oder Drehung angegeben werden .
`to left`	Richtung des Farbverlaufs, Standard ist `to bottom` . Syntax: `to [y-axis(top OR bottom)] [x-axis(left OR right)]` dh `to top right`
`yellow 10%`	Farbe, optional gefolgt von einem Prozentsatz oder einer Länge, um ihn anzuzeigen. Zwei oder mehrmals wiederholt.

Beachten Sie, dass die Farbcodes HEX, RGB, RGBa, HSL und HSLa anstelle von Farbnamen verwendet werden können. Zur Veranschaulichung wurden Farbnamen verwendet. Beachten Sie auch, dass die Radialgradientensyntax viel komplexer ist als der Lineargradient. Hier wird eine vereinfachte Version gezeigt. Eine vollständige Erklärung und Spezifikationen finden Sie in den MDN-Dokumenten

Hintergrund-Kürzel

Die `background` Eigenschaft kann verwendet werden, um eine oder mehrere hintergrundbezogene Eigenschaften festzulegen:

Wert	Beschreibung	CSS Ver.
`background-image`	Zu verwendendes Hintergrundbild	1+
`background-color`	Hintergrundfarbe zum Anwenden	1+
`background-position`	Position des Hintergrundbildes	1+
`background-size`	Größe des Hintergrundbildes	3+
`background-repeat`	So wiederholen Sie das Hintergrundbild	1+
`background-origin`	Wie der Hintergrund positioniert wird (wird ignoriert, wenn der `background-attachment fixed`)	3+
`background-clip`	Wie wird der Hintergrund relativ zum `content-box` , `border-box` oder `padding-box`	3+
`background-attachment`	Wie sich das Hintergrundbild verhält, ob es mit dem enthaltenen Block scrollen oder eine feste Position innerhalb des Ansichtsfensters hat	1+
`initial`	Legt den Wert der Eigenschaft auf Standard fest	3+
`inherit`	Übernimmt den Eigenschaftswert vom übergeordneten Element	2+

Die Reihenfolge der Werte spielt keine Rolle und jeder Wert ist optional

Syntax

Die Syntax der Hintergrund-Abkürzungsdeklaration lautet:

```
background: [<background-image>] [<background-color>]  [<background-position>]/[<background-size>] [<background-repeat>] [<background-origin>] [<background-clip>] [<background-attachment>] [<initial|inherit>];
```

Beispiele

```
background: red;
```

Einfach eine `background-color` mit dem `red` Wert einstellen.

```
background: border-box red;
```

Einen `background-clip` auf ein Rahmenfeld und eine `background-color` auf Rot setzen.

```
background: no-repeat center url("somepng.jpg");
```

Stellt eine `background-repeat` auf Nicht wiederholen, einen `background-origin` auf Mitte und ein `background-image` auf ein Bild ein.

```
background: url('pattern.png') green;
```

In diesem Beispiel wird die `background-color` des Elements mit `pattern.png` auf `green` `pattern.png` , sofern verfügbar, überlagert und so oft wiederholt, bis das Element gefüllt ist. Wenn `pattern.png` Transparenz enthält, wird die `green` Farbe dahinter sichtbar.

```
background: #000000 url("picture.png") top left / 600px auto no-repeat;
```

In diesem Beispiel haben wir einen schwarzen Hintergrund mit einem Bild 'picture.png' oben, das Bild wiederholt sich in keiner der Achsen und befindet sich in der oberen linken Ecke. Das `/` hinter der Position soll die Größe des Hintergrundbildes enthalten können, das in diesem Fall auf `600px` Breite und Auto für die Höhe eingestellt ist. Dieses Beispiel könnte gut mit einem Funktionsbild funktionieren, das zu einer Volltonfarbe überblenden kann.

> **HINWEIS: Die** Verwendung der Kurzschreibe-Hintergrundeigenschaft setzt alle zuvor festgelegten Hintergrundeigenschaftswerte zurück, auch wenn kein Wert angegeben wird. Wenn Sie nur einen zuvor festgelegten Hintergrundeigenschaftswert ändern möchten, verwenden Sie stattdessen eine Langzeiteigenschaft.

Hintergrundposition

Mit `background-position` Eigenschaft " `background-position` wird die Startposition für ein Hintergrundbild oder einen Farbverlauf festgelegt

```
.myClass {
  background-image: url('path/to/image.jpg');
  background-position: 50% 50%;
}
```

Die Position wird mit einer **X-** und **Y** -Koordinate festgelegt und mit einer der in CSS verwendeten Einheiten festgelegt.

Einheit	Beschreibung
Wert % *Wert* %	Ein Prozentsatz für den horizontalen Versatz ist relativ zu *(Breite des Hintergrundpositionierungsbereichs - Breite des Hintergrundbildes)* . Ein Prozentsatz für den vertikalen Versatz ist relativ zu *(Höhe des Hintergrundpositionierungsbereichs - Höhe des Hintergrundbildes)* Die Größe des Bildes ist die Größe, die durch die `background-size` vorgegeben wird.
Wert px *Wert* px	Versetzt das Hintergrundbild um eine in Pixel angegebene Länge relativ zur oberen linken Ecke des Hintergrundpositionierungsbereichs

CSS-Einheiten können mit verschiedenen Methoden angegeben werden (siehe hier).

Langhändige Hintergrundpositionseigenschaften

Neben der Abkürzungseigenschaft oben können Sie auch die Hintergrundeigenschaften long- `background-position-x` und `background-position-y` . Damit können Sie die x- oder y-Positionen separat steuern.

> **HINWEIS:** Dies wird in allen Browsern mit Ausnahme von Firefox (Versionen 31-48) 2 unterstützt . Firefox 49, der im September 2016 veröffentlicht *wird* , *wird* diese Eigenschaften unterstützen. Bis dahin gibt es einen Firefox-Hack in dieser Stack Overflow-Antwort.

Hintergrundbefestigung

Die Eigenschaft Hintergrundanhang legt fest, ob ein Hintergrundbild fixiert ist, oder scrollt mit dem Rest der Seite.

```
body {
  background-image: url('img.jpg');
  background-attachment: fixed;
```

```
}
```

Wert	Beschreibung
scrollen	Der Hintergrund rollt zusammen mit dem Element. Dies ist die Standardeinstellung.
Fest	Der Hintergrund ist bezüglich des Darstellungsbereichs fest.
lokal	Der Hintergrund rollt zusammen mit dem Inhalt des Elements.
Initiale	Setzt diese Eigenschaft auf ihren Standardwert.
erben	Übernimmt diese Eigenschaft von ihrem übergeordneten Element.

Beispiele

Hintergrundanhang: scrollen

Das Standardverhalten, wenn der Körper gescrollt wird, rollt der Hintergrund mit:

```
body {
  background-image: url('image.jpg');
  background-attachment: scroll;
}
```

Hintergrundbefestigung: behoben

Das Hintergrundbild wird fixiert und bewegt sich nicht, wenn der Körper gescrollt wird:

```
body {
  background-image: url('image.jpg');
  background-attachment: fixed;
}
```

Hintergrundanhang: lokal

Das Hintergrundbild des div wird gescrollt, wenn der Inhalt des divs gescrollt wird.

```
div {
  background-image: url('image.jpg');
  background-attachment: local;
}
```

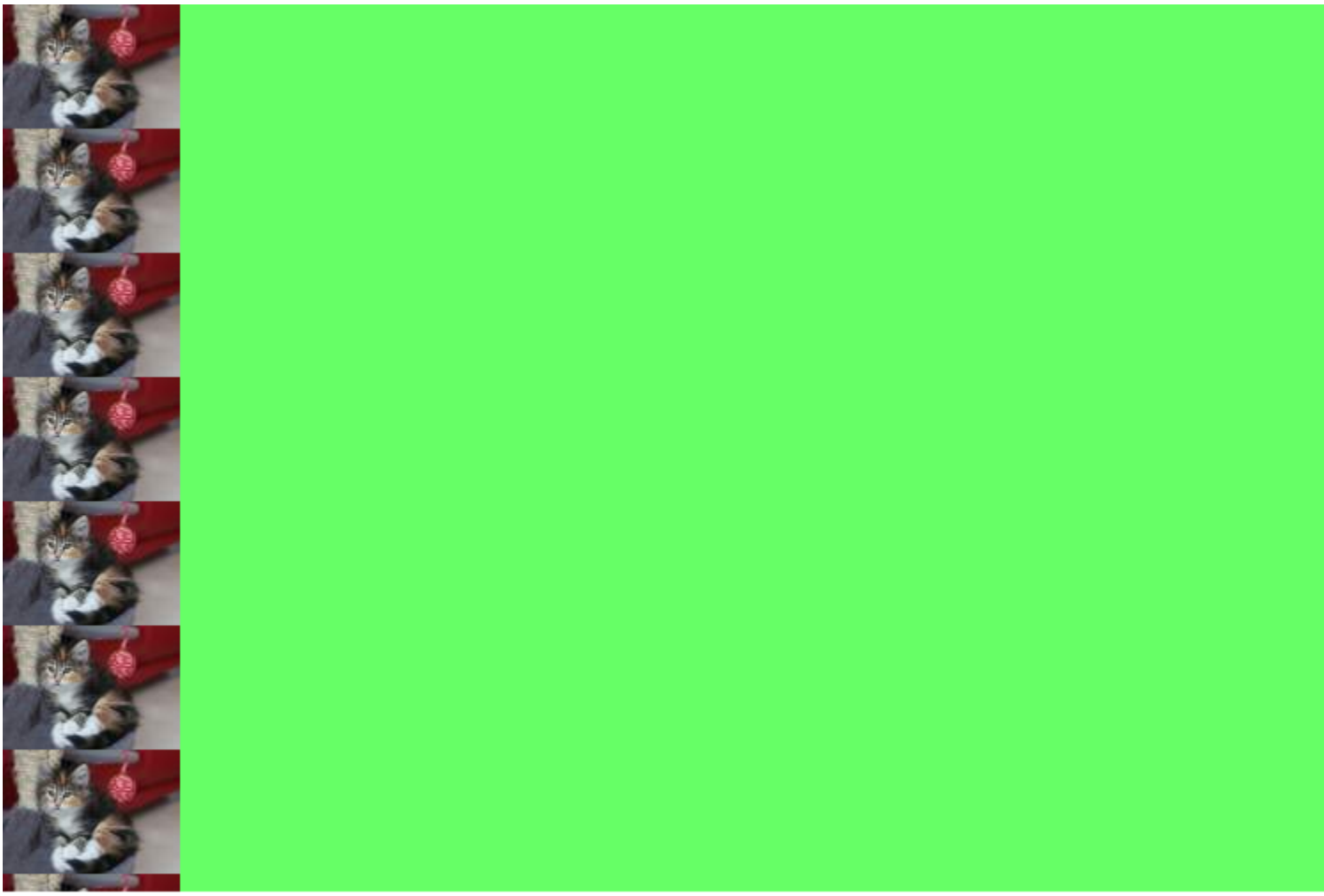
Hintergrund Wiederholung

Die Background-Repeat-Eigenschaft legt fest, ob / wie ein Hintergrundbild wiederholt wird.

Standardmäßig wird ein Hintergrundbild sowohl vertikal als auch horizontal wiederholt.

```
div {
  background-image: url("img.jpg");
  background-repeat: repeat-y;
}
```

Eine `background-repeat: repeat-y` sieht folgendermaßen aus:



Hintergrundfarbe mit Deckkraft

Wenn Sie für ein Element `opacity` festlegen, wirkt sich dies auf alle untergeordneten Elemente aus. Um eine Deckkraft nur auf dem Hintergrund eines Elements festzulegen, müssen Sie RGBA-Farben verwenden. Das folgende Beispiel hat einen schwarzen Hintergrund mit einer Deckkraft von 0,6.

```
/* Fallback for web browsers that don't support RGBa */
background-color: rgb(0, 0, 0);

/* RGBa with 0.6 opacity */
background-color: rgba(0, 0, 0, 0.6);

/* For IE 5.5 - 7*/
filter: progid:DXImageTransform.Microsoft.gradient(startColorstr=#99000000,
endColorstr=#99000000);
```

```
/* For IE 8*/
-ms-filter: "progid:DXImageTransform.Microsoft.gradient(startColorstr=#99000000,
endColorstr=#99000000)";
```

Mehrere Hintergrundbilder

In CSS3 können wir mehrere Hintergrund in demselben Element stapeln.

```
#mydiv {
  background-image: url(img_1.png), /* top image */
                    url(img_2.png), /* middle image */
                    url(img_3.png); /* bottom image */
  background-position: right bottom,
                       left top,
                       right top;
  background-repeat: no-repeat,
                     repeat,
                     no-repeat;
}
```

Die Bilder werden übereinander gestapelt, wobei der erste Hintergrund oben und der letzte Hintergrund hinten liegt. `img_1` wird oben `img_2` , `img_2` und `img_3` sind unten.

Wir können dazu auch die Abkürzungseigenschaft für den Hintergrund verwenden:

```
#mydiv {
  background: url(img_1.png) right bottom no-repeat,
              url(img_2.png) left top repeat,
              url(img_3.png) right top no-repeat;
}
```

Wir können auch Bilder und Farbverläufe stapeln:

```
#mydiv {
  background: url(image.png) right bottom no-repeat,
              linear-gradient(to bottom, #fff 0%,#000 100%);
}
```

- Demo

Die Hintergrundursprungseigenschaft

Die Hintergrundursprungseigenschaft gibt an, wo das Hintergrundbild positioniert wird.

Hinweis: Wenn für die `background-attachment fixed` , hat diese Eigenschaft keine Auswirkungen.

Standardwert: `padding-box`

Mögliche Werte:

- `padding-box` - Die Position ist relativ zur Polsterbox
- `border-box`

- Die Position ist relativ zum Rahmen

- `content-box` - Die Position ist relativ zum Inhaltsfeld
- `initial`
- `inherit`

CSS

```
.example {
  width: 300px;
  border: 20px solid black;
  padding: 50px;
  background: url(https://static.pexels.com/photos/6440/magazines-desk-work-workspace-
medium.jpg);
  background-repeat: no-repeat;
}

.example1 {}

.example2 { background-origin: border-box; }

.example3 { background-origin: content-box; }
```

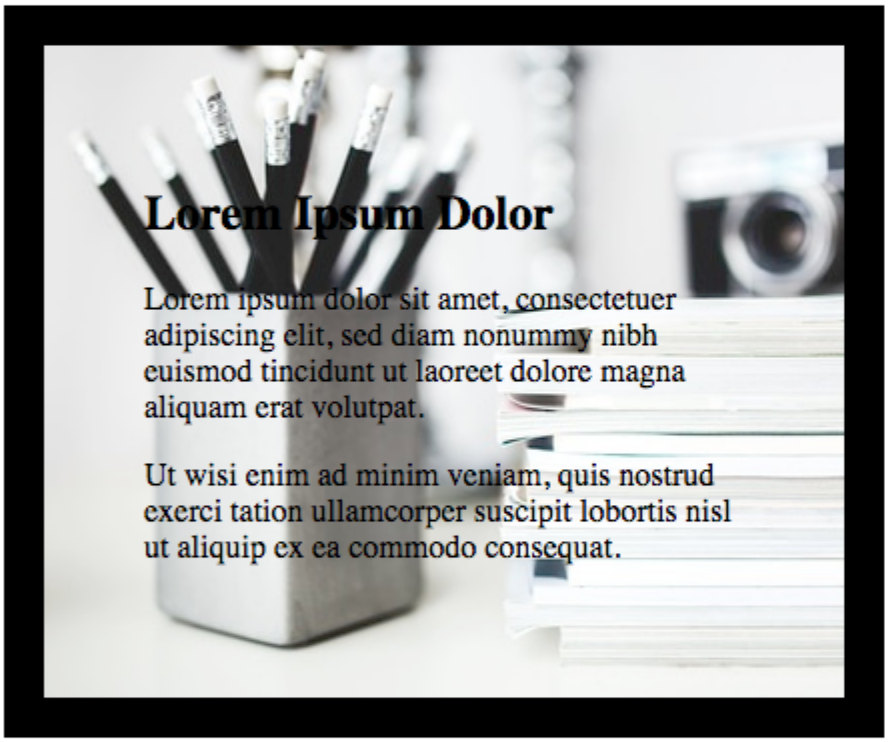
HTML

```
<p>No background-origin (padding-box is default):</p>

<div class="example example1">
  <h2>Lorem Ipsum Dolor</h2>
  <p>Lorem ipsum dolor sit amet, consectetuer adipiscing elit, sed diam nonummy nibh euismod
tincidunt ut laoreet dolore magna aliquam erat volutpat.</p>
  <p>Ut wisi enim ad minim veniam, quis nostrud exerci tation ullamcorper suscipit lobortis
nisl ut aliquip ex ea commodo consequat.</p>
</div>

<p>background-origin: border-box:</p>
<div class="example example2">
  <h2>Lorem Ipsum Dolor</h2>
  <p>Lorem ipsum dolor sit amet, consectetuer adipiscing elit, sed diam nonummy nibh euismod
tincidunt ut laoreet dolore magna aliquam erat volutpat.</p>
  <p>Ut wisi enim ad minim veniam, quis nostrud exerci tation ullamcorper suscipit lobortis
nisl ut aliquip ex ea commodo consequat.</p>
</div>

<p>background-origin: content-box:</p>
<div class="example example3">
  <h2>Lorem Ipsum Dolor</h2>
  <p>Lorem ipsum dolor sit amet, consectetuer adipiscing elit, sed diam nonummy nibh euismod
tincidunt ut laoreet dolore magna aliquam erat volutpat.</p>
  <p>Ut wisi enim ad minim veniam, quis nostrud exerci tation ullamcorper suscipit lobortis
nisl ut aliquip ex ea commodo consequat.</p>
</div>
```

Ergebnis:

No background-origin (padding-box is default):

background-origin: border-box:



background-origin: content-box:

Eigenschaft gibt den Malbereich des Hintergrunds an.

Standardwert: `border-box`

Werte

- `border-box` ist der Standardwert. Dadurch kann sich der Hintergrund bis zur Außenkante des Elementrahmens erstrecken.
- `padding-box` schneidet den Hintergrund an der Außenkante der Auffüllung des Elements und lässt ihn nicht in den Rand hineinragen;
- `content-box` schneidet den Hintergrund am Rand der Inhaltsbox.
- `inherit` wendet die Einstellung des übergeordneten Elements auf das ausgewählte Element an.

CSS

```
.example {
  width: 300px;
  border: 20px solid black;
  padding: 50px;
  background: url(https://static.pexels.com/photos/6440/magazines-desk-work-workspace-
medium.jpg);
  background-repeat: no-repeat;
}

.example1 {}

.example2 { background-origin: border-box; }

.example3 { background-origin: content-box; }
```

HTML

```
<p>No background-origin (padding-box is default):</p>

<div class="example example1">
  <h2>Lorem Ipsum Dolor</h2>
  <p>Lorem ipsum dolor sit amet, consectetuer adipiscing elit, sed diam nonummy nibh euismod
tincidunt ut laoreet dolore magna aliquam erat volutpat.</p>
  <p>Ut wisi enim ad minim veniam, quis nostrud exerci tation ullamcorper suscipit lobortis
nisl ut aliquip ex ea commodo consequat.</p>
</div>

<p>background-origin: border-box:</p>
<div class="example example2">
  <h2>Lorem Ipsum Dolor</h2>
  <p>Lorem ipsum dolor sit amet, consectetuer adipiscing elit, sed diam nonummy nibh euismod
tincidunt ut laoreet dolore magna aliquam erat volutpat.</p>
  <p>Ut wisi enim ad minim veniam, quis nostrud exerci tation ullamcorper suscipit lobortis
nisl ut aliquip ex ea commodo consequat.</p>
</div>

<p>background-origin: content-box:</p>
<div class="example example3">
  <h2>Lorem Ipsum Dolor</h2>
  <p>Lorem ipsum dolor sit amet, consectetuer adipiscing elit, sed diam nonummy nibh euismod
```

```
tincidunt ut laoreet dolore magna aliquam erat volutpat.</p>
  <p>Ut wisi enim ad minim veniam, quis nostrud exerci tation ullamcorper suscipit lobortis
nisl ut aliquip ex ea commodo consequat.</p>
</div>
```

Hintergrundgröße

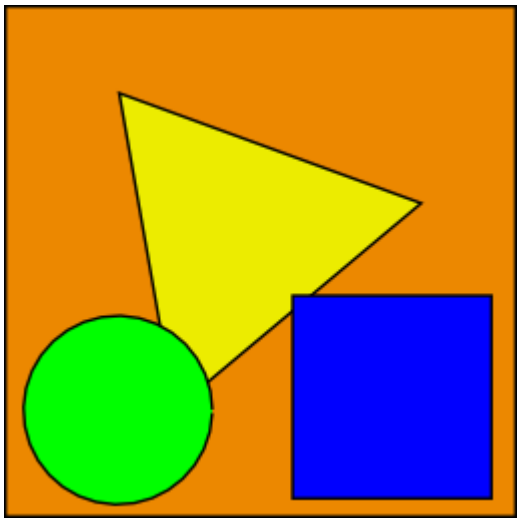
Gesamtübersicht

Die `background-size` Eigenschaft ermöglicht es , die Skalierung des steuern `background-image` . Es werden bis zu zwei Werte benötigt, die den Maßstab / die Größe des resultierenden Bildes in vertikaler und horizontaler Richtung bestimmen. Wenn die Eigenschaft fehlt, gilt ihre `width` und `height` als `auto` .

`auto` behält das Bildseitenverhältnis des Bildes bei, wenn es bestimmt werden kann. Die Höhe ist optional und kann als `auto` . Daher würden bei einem Bild mit 256 x 256 Pixeln alle folgenden Einstellungen für die `background-size` einem Bild mit einer Höhe und Breite von 50 px führen:

```
background-size: 50px;
background-size: 50px auto; /* same as above */
background-size: auto 50px;
background-size: 50px 50px;
```

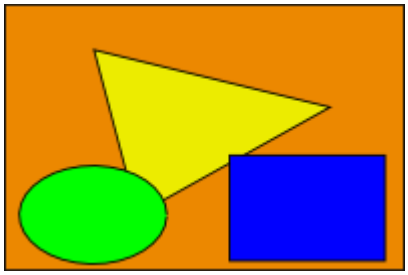
Wenn wir also mit dem folgenden Bild beginnen (welches die erwähnte Größe von 256 px × 256 px hat),



Auf dem Bildschirm des Benutzers werden am Ende des Elements 50 x 50 x 50 Pixel angezeigt, die im Hintergrund unseres Elements enthalten sind:

Man kann auch Prozentwerte verwenden, um das Bild in Bezug auf das Element zu skalieren. Das folgende Beispiel würde ein gezeichnetes Bild mit 200 x 133 x 133 Pixel ergeben:

```
#withbackground {
    background-image: url(to/some/background.png);

    background-size: 100% 66%;

    width: 200px;
    height: 200px;

    padding: 0;
    margin: 0;
}
```



Das Verhalten hängt vom `background-origin` .

Das Seitenverhältnis beibehalten

Das letzte Beispiel im vorherigen Abschnitt hat sein ursprüngliches Seitenverhältnis verloren. Der Kreis wurde zu einer Ellipse, das Quadrat zu einem Rechteck, das Dreieck zu einem anderen Dreieck.

Der Längen- oder Prozentwertansatz ist nicht flexibel genug, um das Seitenverhältnis zu jeder Zeit beizubehalten. `auto` hilft nicht, da Sie möglicherweise nicht wissen, welche Dimension Ihres Elements größer ist. Jedoch können bestimmte Bereiche ein Bild zu bedecken (und das Verhältnis richtigen Seiten) vollständig oder ein Bild mit dem richtigen Seitenverhältnis vollständig in einem Hintergrundbereich zu enthalten, wobei die Werte, `contain` und `cover` die zusätzliche Funktionalität bereitzustellen.

Eggsplanation für `contain` und `cover`

Entschuldigung für das schlechte Wortspiel, aber wir werden ein Bild des Tages von Biswarup Ganguly zur Demonstration verwenden. Nehmen wir an, dies ist Ihr Bildschirm und der graue Bereich befindet sich außerhalb Ihres sichtbaren Bildschirms. Zur Demonstration gehen wir von einem 16 × 9-Verhältnis aus.

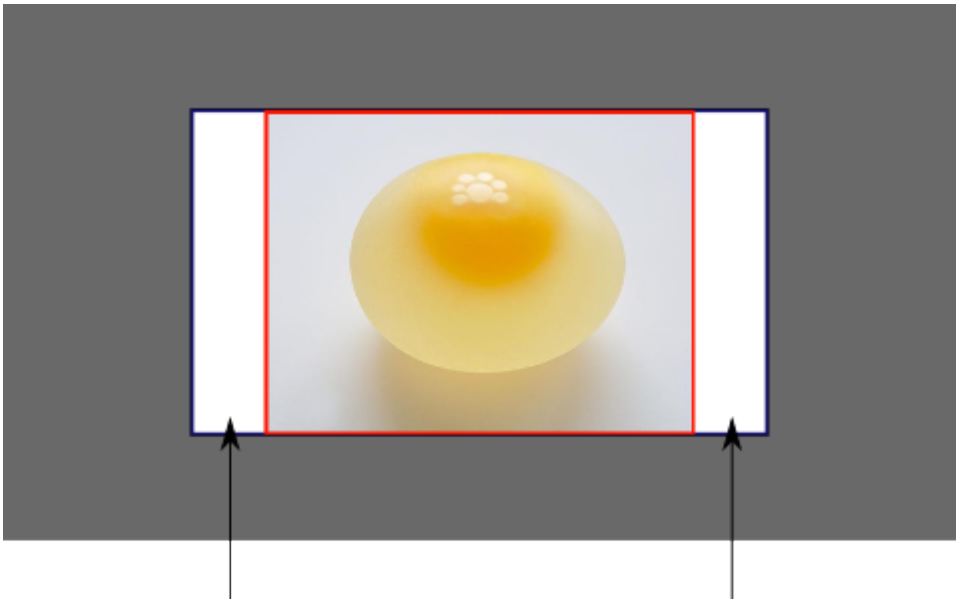
Wir möchten das oben genannte Bild des Tages als Hintergrund verwenden. Aus irgendeinem Grund haben wir das Bild jedoch auf 4x3 zugeschnitten. Wir können die Eigenschaft für die `background-size` auf eine feste Länge festlegen, wir werden uns jedoch auf das `contain` und `cover` . Ich gehe auch davon aus, dass wir die Breite und / oder Höhe des `body` nicht beeinträchtigt haben.

contain

```
contain
```

> Skalieren Sie das Bild unter Beibehaltung des intrinsischen Seitenverhältnisses (sofern vorhanden) auf die größte Größe, sodass sowohl Breite als auch Höhe in den Hintergrundpositionierungsbereich passen können.

Dadurch wird sichergestellt, dass das Hintergrundbild immer vollständig im Hintergrundpositionierungsbereich enthalten ist. In diesem Fall könnte jedoch ein leerer Bereich mit Ihrer `background-color` gefüllt sein:

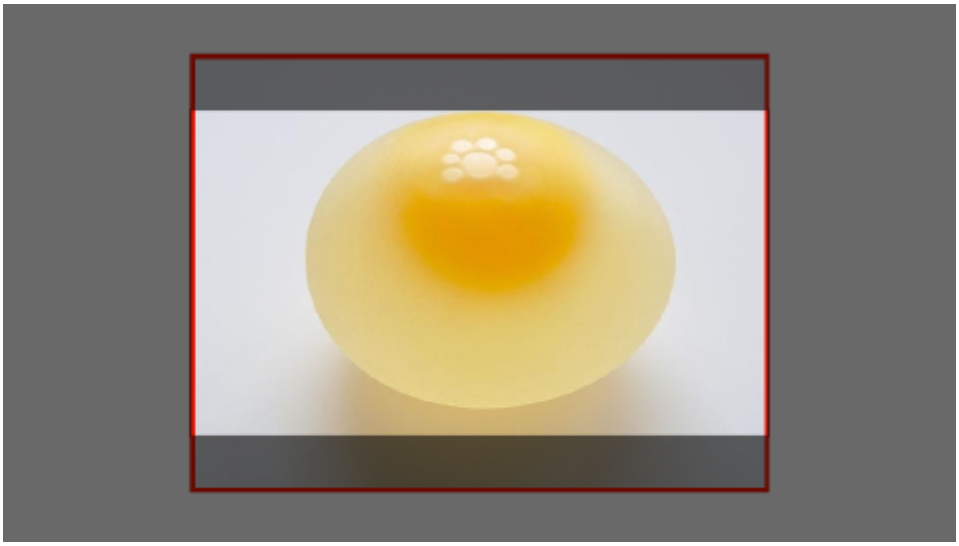


cover

```
cover
```

Skalieren Sie das Bild unter Beibehaltung des intrinsischen Seitenverhältnisses (sofern vorhanden) auf die kleinste Größe, sodass sowohl Breite als auch Höhe den Hintergrundpositionierungsbereich vollständig abdecken können.

Dadurch wird sichergestellt, dass das Hintergrundbild alles abdeckt. Es wird keine sichtbare `background-color` angezeigt. Je nach Bildschirmverhältnis kann jedoch ein großer Teil Ihres Bildes abgeschnitten werden:



Demonstration mit aktuellem Code

```
div > div {
  background-image: url(http://i.stack.imgur.com/r5CAq.jpg);
  background-repeat: no-repeat;
  background-position: center center;
  background-color: #ccc;
  border: 1px solid;
  width: 20em;
  height: 10em;
}
div.contain {
  background-size: contain;
}
div.cover {
  background-size: cover;
}
/*****************************************
 Additional styles for the explanation boxes
*****************************************/

div > div {
  margin: 0 1ex 1ex 0;
  float: left;
}
div + div {
  clear: both;
  border-top: 1px dashed silver;
  padding-top:1ex;
```

```
}
div > div::after {
  background-color: #000;
  color: #fefefe;
  margin: 1ex;
  padding: 1ex;
  opacity: 0.8;
  display: block;
  width: 10ex;
  font-size: 0.7em;
  content: attr(class);
}
```

```
<div>
  <div class="contain"></div>
  <p>Note the grey background. The image does not cover the whole region, but it's fully
<em>contained</em>.
  </p>
</div>
<div>
  <div class="cover"></div>
  <p>Note the ducks/geese at the bottom of the image. Most of the water is cut, as well as a
part of the sky. You don't see the complete image anymore, but neither do you see any
background color; the image <em>covers</em> all of the <code>&lt;div&gt;</code>.</p>
</div>
```

Note the grey background. The image does not cover the whole region, but it's fully *contained*.

Note the ducks/geese at the bottom of the image. Most of the water is cut, as well as a part of the sky. You don't see the complete image anymore, but neither do you see any background color; the image *covers* all of the `<div>`.

background-blend-mode-Eigenschaft

```
.my-div {
    width: 300px;
    height: 200px;
    background-size: 100%;
    background-repeat: no-repeat;
    background-image: linear-gradient(to right, black 0%,white 100%),
url('https://static.pexels.com/photos/54624/strawberry-fruit-red-sweet-54624-medium.jpeg');
    background-blend-mode:saturation;
```

```
}
```

```
<div class="my-div">Lorem ipsum</div>
```

Das Ergebnis finden Sie hier: https://jsfiddle.net/MadalinaTn/y69d28Lb/

CSS-Syntax: Hintergrund-Mischmodus: Normal | multiplizieren | Bildschirm | Überlagerung | verdunkeln | aufhellen | Farbe ausweichen | Sättigung | Farbe | Helligkeit;

Hintergründe online lesen: https://riptutorial.com/de/css/topic/296/hintergrunde

Kapitel 27: Inline-Block-Layout

Examples

Berechtigte Navigationsleiste

Die horizontal ausgerichtete Navigationsleiste (Menüleiste) enthält einige Elemente, die gerechtfertigt sein sollten. Das erste (linke) Element hat keinen linken Rand innerhalb des Containers, das letzte (rechte) Element hat keinen rechten Rand innerhalb des Containers. Der Abstand zwischen den Artikeln ist unabhängig von der Breite der einzelnen Artikel gleich.

HTML

```
<nav>
    <ul>
        <li>abc</li>
        <li>abcdefghijkl</li>
        <li>abcdef</li>
    </ul>
</nav>
```

CSS

```
nav {
    width: 100%;
    line-height: 1.4em;
}
ul {
    list-style: none;
    display: block;
    width: 100%;
    margin: 0;
    padding: 0;
    text-align: justify;
    margin-bottom: -1.4em;
}
ul:after {
    content: "";
    display: inline-block;
    width: 100%;
}
li {
    display: inline-block;
}
```

Anmerkungen

- Die Tags `nav` , `ul` und `li` wurden aufgrund ihrer semantischen Bedeutung von "Liste der Navigationselemente (Menüelemente)" ausgewählt. Natürlich können auch andere Tags verwendet werden.
- Das `:after` -Pseudoelement bewirkt eine zusätzliche 'Linie' in der `ul` und damit eine zusätzliche, leere Höhe dieses Blocks, die den anderen Inhalt nach unten drückt. Dies wird durch den negativen `margin-bottom` gelöst, der die gleiche Höhe wie die `line-height` (aber negativ).
- Wenn die Seite für alle Elemente zu eng wird, werden die Elemente in eine neue Zeile (von rechts beginnend) aufgeteilt und in dieser Zeile begründet. Die Gesamthöhe des Menüs wird nach Bedarf größer.

Inline-Block-Layout online lesen: https://riptutorial.com/de/css/topic/3308/inline-block-layout

Kapitel 28: Internet Explorer-Hacks

Bemerkungen

Diese "Hacks" können verwendet werden, um einen bestimmten Browser / Client anzusprechen. Dies kann verwendet werden, um Browser-Rendering-Unterschiede zu umgehen, indem Sie Stile in einer der oben aufgeführten Wrapper anwenden.

Examples

High Contrast-Modus in Internet Explorer 10 und höher

In Internet Explorer 10+ und Edge bietet Microsoft die `-ms-high-contrast` Medien an, um die Einstellung "High Contrast" im Browser anzuzeigen, wodurch der Programmierer die Stile der Site entsprechend anpassen kann.

Der `-ms-high-contrast` hat 3 Zustände: `active` , `black-on-white` und `white-on-black` . In IE10 + hatte es auch `none` Status, der in Edge jedoch nicht mehr unterstützt wird.

Beispiele

```
@media screen and (-ms-high-contrast: active), (-ms-high-contrast: black-on-white) {
   .header{
      background: #fff;
      color: #000;
   }
}
```

Dadurch wird der Hintergrund der Kopfzeile in Weiß und die Textfarbe in Schwarz geändert, wenn der Hochkontrastmodus aktiviert ist *und* sich der `black-on-white` Modus befindet.

```
@media screen and (-ms-high-contrast: white-on-black) {
   .header{
      background: #000;
      color: #fff;
   }
}
```

Ähnlich wie im ersten Beispiel, dies wählt jedoch nur den `white-on-black` Status aus und invertiert die Header-Farben in einen schwarzen Hintergrund mit weißem Text.

Mehr Informationen:

Microsoft-Dokumentation zu `-ms-high-contrast`

Nur Internet Explorer 6 und Internet Explorer 7

Um Internet Explorer 6 und Internet Explorer 7 als Ziel festzulegen, starten Sie Ihre Eigenschaften mit `*` :

```
.hide-on-ie6-and-ie7 {
    *display : none; // This line is processed only on IE6 and IE7
}
```

Nicht-alphanumerische Präfixe (mit Ausnahme von Bindestrichen und Unterstrichen) werden in IE6 und IE7 ignoriert. Daher funktioniert dieser Hack für jedes nicht Präfix- `property: value` .

Nur Internet Explorer 8

Um Internet Explorer 8 als Ziel `@media \0 screen { }` , wickeln Sie Ihre Selektoren in `@media \0 screen { }` :

```
@media \0 screen {
    .hide-on-ie8 {
        display : none;
    }
}
```

Alles zwischen `@media \0 screen { }` wird nur von I verarbeitet

Hinzufügen von Inline-Block-Unterstützung zu IE6 und IE7

```
display: inline-block;
```

Die `display` mit dem Wert von `inline-block` wird von Internet Explorer 6 und 7 nicht unterstützt. Eine Problemumgehung hierfür ist:

```
zoom: 1;
*display: inline;
```

Die `zoom` Eigenschaft löst die `hasLayout` Funktion von Elementen aus und ist nur in Internet Explorer verfügbar. Die `*display` stellt sicher, dass die ungültige Eigenschaft nur auf den betroffenen Browsern ausgeführt wird. Andere Browser ignorieren die Regel einfach.

Internet Explorer-Hacks online lesen: https://riptutorial.com/de/css/topic/5056/internet-explorer-hacks

Kapitel 29: Kaskadierung und Spezifität

Bemerkungen

Durch die CSS-Spezifität soll der Code prägnanter gemacht werden, indem einem Autor ermöglicht wird, einige allgemeine Formatierungsregeln für eine große Anzahl von Elementen zu definieren und diese dann für eine bestimmte Teilmenge zu überschreiben.

Examples

Kaskadieren

Kaskadierung und Spezifität werden zusammen verwendet, um den endgültigen Wert einer CSS-Stileigenschaft zu bestimmen. Sie definieren auch die Mechanismen zum Lösen von Konflikten in CSS-Regelsätzen.

CSS-Ladereihenfolge

Stile werden aus folgenden Quellen in dieser Reihenfolge gelesen:

1. User Agent-Stylesheet (die vom Browser-Anbieter bereitgestellten Stile)
2. User Stylesheet (das zusätzliche Styling, das ein Benutzer in seinem Browser festgelegt hat)
3. Autoren-Stylesheet (Autor bezeichnet hier den Ersteller der Webseite / Website)
    - Möglicherweise eine oder mehrere `.css` Dateien
    - Im `<style>` -Element des HTML-Dokuments
4. Inline-Stile (im `style` Attribut eines HTML-Elements)

Der Browser sucht beim Rendern eines Elements nach den entsprechenden Stilen.

Wie werden Konflikte gelöst?

Wenn nur ein CSS-Regelsatz versucht, einen Stil für ein Element festzulegen, liegt kein Konflikt vor, und dieser Regelsatz wird verwendet.

Wenn mehrere Regelsätze mit widersprüchlichen Einstellungen gefunden werden, werden zuerst die Specificty-Regeln und dann die Cascading-Regeln verwendet, um den zu verwendenden Stil zu bestimmen.

Beispiel 1 - Spezifitätsregeln

```
.mystyle { color: blue; }  /* specificity: 0, 0, 1, 0 */
div { color: red; }        /* specificity: 0, 0, 0, 1 */
```

```
<div class="mystyle">Hello World</div>
```

Welche Farbe wird der Text haben? (schweben, um die Antwort zu sehen)

> Blau

Zuerst werden die Spezifitätsregeln angewendet und diejenige mit der höchsten Spezifität "gewinnt".

Beispiel 2 - Kaskadenregeln mit identischen Selektoren

Externe CSS-Datei

```
.class {
  background: #FFF;
}
```

Interne CSS (in HTML-Datei)

```
<style>
.class {
  background: #000;
}
<style>
```

In diesem Fall setzt die Kaskade bei identischen Selektoren ein und stellt fest, dass der zuletzt geladene "gewinnt".

Beispiel 3 - Kaskadenregeln nach Spezifitätsregeln

```
body > .mystyle { background-color: blue; }   /* specificity: 0, 0, 1, 1 */
.otherstyle > div { background-color: red; }  /* specificity: 0, 0, 1, 1 */
```

```
<body class="otherstyle">
  <div class="mystyle">Hello World</div>
</body>
```

Welche Farbe wird der Hintergrund haben?

> rot

Nach dem Anwenden der Spezifitätsregeln gibt es immer noch einen Konflikt zwischen Blau und Rot, sodass die Kaskadenregeln zusätzlich zu den Spezifitätsregeln angewendet werden. Beim Kaskadieren wird die `.css` der Regeln untersucht, unabhängig davon, ob sie sich in derselben `.css` Datei oder in der Sammlung von Stilquellen befinden. Der zuletzt geladene überschreibt alle früheren. In diesem Fall "gewinnt" die Regel `.otherstyle > div` .

Eine letzte Notiz

- Selektorspezifität hat immer Vorrang.
- Stylesheet-Bestellung unterbricht Krawatten.
- Inline-Stile übertreffen alles.

Die! Wichtige Erklärung

Die `!important` Deklaration wird verwendet, um die übliche Spezifität in einem Stylesheet zu überschreiben, indem einer Regel eine höhere Priorität zugewiesen wird. Ihre Verwendung ist: `property : value !important;`

```
#mydiv {
    font-weight: bold !important;    /* This property won't be overridden
                                        by the rule below */
}

#outerdiv #mydiv {
    font-weight: normal;            /* #mydiv font-weight won't be set to normal
                                    even if it has a higher specificity because
                                    of the !important declaration above */
}
```

Das Vermeiden der Verwendung von `!important` wird dringend empfohlen (es sei denn, dies ist absolut erforderlich), da dies den natürlichen Fluss der CSS-Regeln stören würde, was zu Unsicherheit in Ihrem Stylesheet führen kann. Es ist auch wichtig zu beachten, dass, wenn mehrere `!important` Deklarationen auf die gleiche Regel auf ein bestimmtes Element angewendet werden, diejenige mit der höheren Spezifität das ona ist.

Hier einige Beispiele, bei denen die Verwendung `!important` Deklarationen gerechtfertigt ist:

- Wenn Ihre Regeln nicht durch einen Inline-Stil des Elements überschrieben werden sollen, der in das `style` Attribut des html-Elements geschrieben wird.
- Um dem Benutzer mehr Kontrolle über die Webzugriffsmöglichkeiten zu geben, z. B. das Vergrößern oder Verkleinern der Schriftgröße, indem der Autorstil mit `!important` überschrieben wird.
- Zum Testen und Debuggen mit inspect element.

Siehe auch:

- **W3C - 6 Eigenschaftenwerte, Kaskadierung und Vererbung zuweisen - 6.4.2! Wichtige Regeln**

Berechnung der Selektorspezifität

Jeder einzelne CSS-Selector hat seinen eigenen Spezifitätswert. Jeder Selektor in einer Sequenz erhöht die Gesamtspezifität der Sequenz. Selektoren fallen in eine von drei verschiedenen Spezifitätsgruppen: *A* , *B* und *c* . Wenn mehrere Auswahlsequenzen ein bestimmtes Element auswählen, verwendet der Browser die von der Sequenz verwendeten Stile mit der höchsten Gesamtspezifität.

Gruppe	Bestehend aus	Beispiele
EIN	ID-Selektoren	`#foo`

B	Klassen-Selektoren
	Attributselektoren
	Pseudoklassen
		`.bar`
		`[title]` , `[colspan="2"]`
		`:hover` `:nth-child(2)`

c	Typenauswahl Pseudoelemente	`div` , `li` `::before` , `::first-letter`

Gruppe *A* ist am spezifischsten, gefolgt von Gruppe *B* und schließlich Gruppe *c* .

Der Universalselektor (`*`) und die Kombinatoren (wie `>` und `~`) sind nicht spezifisch.

Beispiel 1: Spezifität verschiedener Selektorsequenzen

```
#foo #baz {}      /* a=2, b=0, c=0 */

#foo.bar {}       /* a=1, b=1, c=0 */

#foo {}           /* a=1, b=0, c=0 */

.bar:hover {}     /* a=0, b=2, c=0 */

div.bar {}        /* a=0, b=1, c=1 */

:hover {}         /* a=0, b=1, c=0 */

[title] {}        /* a=0, b=1, c=0 */

.bar {}           /* a=0, b=1, c=0 */

div ul + li {}    /* a=0, b=0, c=3 */

p::after {}       /* a=0, b=0, c=2 */

*::before {}      /* a=0, b=0, c=1 */

::before {}       /* a=0, b=0, c=1 */

div {}            /* a=0, b=0, c=1 */

* {}              /* a=0, b=0, c=0 */
```

Beispiel 2: Wie wird die Spezifität vom Browser verwendet?

Stellen Sie sich die folgende CSS-Implementierung vor:

```
#foo {
  color: blue;
}

.bar {
  color: red;
```

```
  background: black;
}
```

Hier haben wir einen ID-Selektor, der die `color` als *blau* deklariert, und einen Klassenselektor, der die `color` als *Rot* und den `background` als *Schwarz* deklariert.

Ein Element mit der ID `#foo` und der Klasse `.bar` wird von beiden Deklarationen ausgewählt. ID-Selektoren haben eine Spezifität der Gruppe *A* und Klassenselektoren haben eine Spezifität der Gruppe *B.* Ein ID-Selektor überwiegt eine beliebige Anzahl von Klassenselektoren. Aus diesem Grund `color:blue;` aus dem `#foo` Selektor und dem `background:black;` von der `.bar` wird auf das Element angewendet. Die höhere Spezifität des ID - Selektor wird der Browser das ignorieren `.bar` Wähler der `color` Erklärung.

Stellen Sie sich jetzt eine andere CSS-Implementierung vor:

```
.bar {
  color: red;
  background: black;
}

.baz {
  background: white;
}
```

Hier haben wir zwei Klassenselektoren; eine davon deklariert die `color` als *rot* und den `background` als *schwarz* und die andere den `background` als *weiß* .

Ein Element mit den Klassen `.bar` und `.baz` wird von beiden Deklarationen betroffen sein. Das Problem, das wir jetzt haben, ist jedoch, dass sowohl `.bar` als auch `.baz` eine identische Spezifität für Gruppe *B aufweisen* . Die Kaskadierung von CSS löst dies für uns auf: `.baz` *nach* `.bar` definiert `.bar` , endet unser Element mit der *roten* `color` von `.bar` , dem *weißen* `background` von `.baz` .

Beispiel 3: Manipulation der Spezifität

Der letzte Ausschnitt aus dem obigen Beispiel 2 kann manipuliert werden , um sicherzustellen , unsere `.bar` Klassenauswahl der `color` Erklärung statt , dass die verwendeten `.baz` Klassenauswahl.

```
.bar {}        /* a=0, b=1, c=0 */
.baz {}        /* a=0, b=1, c=0 */
```

Die gebräuchlichste Methode, dies zu erreichen, besteht darin, herauszufinden, welche anderen Selektoren auf die `.bar` Selektorsequenz angewendet werden können. Wenn beispielsweise die `.bar` Klasse nur auf `span` Elemente angewendet wurde, können Sie den `.bar` Selektor in `span.bar` . Dies würde zu einer neuen Gruppe *C*- Spezifität führen, die den Mangel der `.baz` außer Kraft setzen würde:

```
span.bar {}    /* a=0, b=1, c=1 */
.baz {}        /* a=0, b=1, c=0 */
```

Es kann jedoch nicht immer möglich sein, einen anderen allgemeinen Selektor zu finden, der von jedem Element gemeinsam genutzt wird, das die `.bar` Klasse verwendet. Aus diesem Grund ermöglicht CSS die Selektion von Selektoren, um die Spezifität zu erhöhen. Anstelle von `.bar` können `.bar.bar` stattdessen `.bar.bar` verwenden (siehe Die Grammatik von Selectors, W3C-Empfehlung). Dies wählt immer noch jedes Element mit der Klasse `.bar` , hat aber jetzt die doppelte Spezifität der Gruppe *B* :

```
.bar.bar {}     /* a=0, b=2, c=0 */
.baz {}         /* a=0, b=1, c=0 */
```

`!important` und Inline-Stildeklarationen

Die `!important` - Flagge auf einer Stildeklaration und Stile von dem HTML erklärten `style` - Attribut gilt als eine höhere Spezifität als jeder Wähler hat. Wenn diese vorhanden sind, überschreibt die Stildeklaration, auf die sie sich auswirkt, andere Deklarationen unabhängig von ihrer Spezifität. Das heißt, es sei denn, Sie haben mehr als eine Deklaration, die ein `!important` Flag für dieselbe Eigenschaft enthält, das für dasselbe Element gilt. Für diese Eigenschaften gelten dann normale Spezifitätsregeln, die sich aufeinander beziehen.

Da sie die Spezifität vollständig außer Kraft setzen, wird die Verwendung von `!important` in den meisten Anwendungsfällen missbilligt. Man sollte es so wenig wie möglich verwenden. CSS - Code effizient und wartbar auf lange Sicht zu halten, ist es fast immer besser , die Spezifität des umgebenden Wähler zu erhöhen , als zu verwenden , um `!important` .

Eine der seltenen Ausnahmen, bei denen `!important` ist, ist die Implementierung allgemeiner `.hidden` wie einer `.hidden` oder `.background-yellow` Klasse, die eine oder mehrere Eigenschaften immer dort überschreiben soll, wo sie vorkommen. Und selbst dann müssen Sie wissen, was Sie tun. Das letzte, was Sie beim Schreiben von wartungsfähigem CSS wollen, ist, `!important` Flaggen in Ihrem CSS zu haben.

Eine letzte Notiz

Ein häufiges Missverständnis bezüglich der CSS-Spezifität besteht darin, dass die Werte für Gruppe *A* , *B* und *c* miteinander kombiniert werden sollten (`a=1, b=5, c=1` => 151). Dies ist **nicht** der Fall. Wenn dies der Fall wäre, reicht es aus, 20 Gruppen der Gruppe *B* oder *c* zu wählen, um jeweils eine Gruppe *A* oder *B* zu überschreiben. Die drei Gruppen sollten als individuelle Spezifitätsebenen betrachtet werden. Die Spezifität kann nicht durch einen einzelnen Wert dargestellt werden.

Beim Erstellen Ihres CSS-Stylesheets sollten Sie die geringstmögliche Spezifität beibehalten. Wenn Sie die Spezifität etwas höher machen müssen, um eine andere Methode zu überschreiben, erhöhen Sie sie, aber so niedrig wie möglich, um sie zu erhöhen. Sie sollten keinen Selektor wie diesen brauchen:

```
body.page header.container nav div#main-nav li a {}
```

Dies macht zukünftige Änderungen schwieriger und verschmutzt diese CSS-Seite.

Sie können die Spezifität Ihrer Wähler berechnen hier

Beispiel für komplexere Spezifität

```
div {
    font-size: 7px;
    border: 3px dotted pink;
    background-color: yellow;
    color: purple;
}

body.mystyle > div.myotherstyle {
    font-size: 11px;
    background-color: green;
}

#elmnt1 {
    font-size: 24px;
    border-color: red;
}

.mystyle .myotherstyle {
    font-size: 16px;
    background-color: black;
    color: red;
}
```

```
<body class="mystyle">
    <div id="elmnt1" class="myotherstyle">
        Hello, world!
    </div>
</body>
```

Welche Rahmen, Farben und Schriftgrößen wird der Text haben?

Schriftgröße:

> `font-size: 24;` Da der `#elmnt1` die höchste Spezifität für das betreffende `<div>` , wird hier jede Eigenschaft festgelegt.

Rand:

> `border: 3px dotted red;` . Die `#elmnt1 red` wird aus dem `#elmnt1` Regelsatz übernommen, da sie die höchste Spezifität aufweist. Die anderen Eigenschaften des Rahmens, der Randstärke und des `div` stammen aus dem `div` Regelsatz.

Hintergrundfarbe:

> `background-color: green;` . Die `background-color` wird in den `body.mystyle > div.myotherstyle div` , `body.mystyle > div.myotherstyle` und `.mystyle .myotherstyle` . Die Besonderheiten sind (0, 0, 1) vs. (0, 2, 2) vs. (0, 2, 0), also "gewinnt" der mittlere.

Farbe:

> `color: red;`

. Die Farbe wird in den `.mystyle .myotherstyle div` und `.mystyle .myotherstyle` . Letzteres hat die höhere Spezifität von (0, 2, 0) und "gewinnt".

Kaskadierung und Spezifität online lesen: https://riptutorial.com/de/css/topic/450/kaskadierung-und-spezifitat

Kapitel 30: Längeneinheiten

Einführung

Eine CSS-Abstandsmessung ist eine Zahl, der unmittelbar eine Längeneinheit folgt (px, em, pc, in,…)

CSS unterstützt eine Reihe von Längenmaßeinheiten. Sie sind absolut oder relativ.

Syntax

- ***Werteinheit***
- 1em

Parameter

Einheit	Beschreibung
%	Definieren Sie Größen in Bezug auf übergeordnete Objekte oder aktuelle Objekte abhängig von der Eigenschaft
em	Relativ zur Schriftgröße des Elements (2em bedeutet 2-fache Größe der aktuellen Schrift)
rem	Relativ zur Schriftgröße des Wurzelelements
vw	Relativ zu 1% der Breite des Ansichtsfensters *
vh	Relativ zu 1% der Höhe des Darstellungsbereichs *
vmin	Relativ zu 1% der kleineren Dimension * des Ansichtsfensters
vmax	Relativ zu 1% der * größeren Abmessung des Ansichtsfensters
cm	Zentimeter
mm	Millimeter
im	Zoll (1 Zoll = 96px = 2,54 cm)
px	Pixel (1px = 1 / 96stel von 1 Zoll)
pt	Punkte (1 Punkt = 1/72 von 1 Zoll)
Stck	picas (1pc = 12 pt)
s	Sekunden (für Animationen und Übergänge)

Einheit	Beschreibung
Frau	Millisekunden (für Animationen und Übergänge)
Ex	Relativ zur x-Höhe der aktuellen Schrift
CH	Basierend auf der Breite des Nullzeichens (0)
fr	gebrochene Einheit (für CSS-Rasterlayout verwendet)

Bemerkungen

- Zwischen der Nummer und der Einheit darf kein Leerzeichen angezeigt werden. Wenn der Wert jedoch 0 ist, kann die Einheit weggelassen werden.
- Für einige CSS-Eigenschaften sind negative Längen zulässig.

Examples

Schriftgröße mit rem

CSS3 führt einige neue Units ein, einschließlich der `rem` Unit, die für "root em" steht. Schauen wir uns an, wie `rem` funktioniert.

Betrachten wir zunächst die Unterschiede zwischen `em` und `rem` .

- **em** : Relativ zur Schriftgröße des übergeordneten Elements. Dies verursacht das Compoundierungsproblem
- **rem** : Relativ zur Schriftgröße des Root- oder `<html>` -Elements. Das heißt, es ist möglich, eine einzige Schriftgröße für das HTML-Element zu definieren und alle `rem` Einheiten als Prozentsatz davon festzulegen.

Das Hauptproblem bei der Verwendung von `rem` für die Schriftgröße besteht darin, dass die Werte etwas schwer zu verwenden sind. Hier ein Beispiel für einige gängige Schriftgrößen, die in `rem` Einheiten ausgedrückt werden, wobei die Basisgröße 16px ist:

- 10px = 0,625rem
- 12px = 0,75rem
- 14px = 0,875rem
- 16px = 1rem (Basis)
- 18px = 1,125rem
- 20px = 1,25rem
- 24px = 1,5rem
- 30px = 1,875rem
- 32px = 2rem

CODE:

3

```
html {
  font-size: 16px;
}

h1 {
  font-size: 2rem;          /* 32px */
}

p {
  font-size: 1rem;          /* 16px */
}

li {
  font-size: 1.5em;         /* 24px */
}
```

Skalierbare Elemente mit rems und ems erstellen

3

Sie können `rem` durch die `font-size` Ihres `html` Tags definieren, um Elemente zu formatieren, indem Sie deren `font-size` auf `rem` und mit `em` innerhalb des Elements Elemente erstellen, die mit Ihrer globalen `font-size` skalieren.

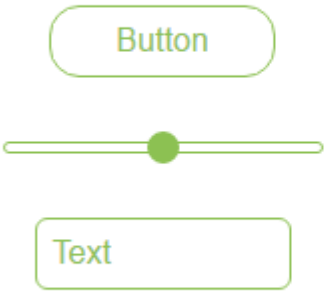
HTML:

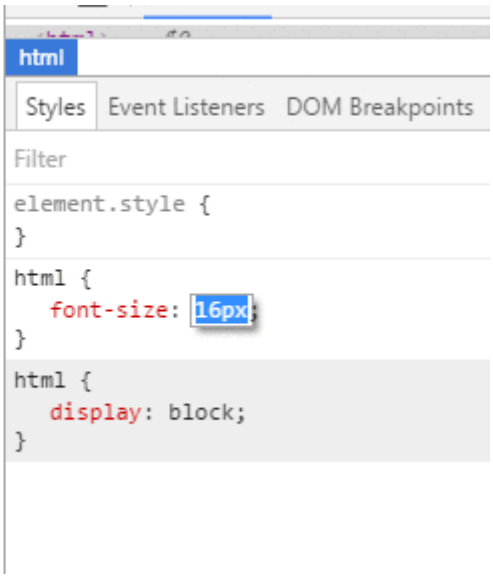
```
<input type="button" value="Button">
<input type="range">
<input type="text" value="Text">
```

Relevantes CSS:

```
html {
  font-size: 16px;
}

input[type="button"] {
  font-size: 1rem;
  padding: 0.5em 2em;
}

input[type="range"] {
  font-size: 1rem;
  width: 10em;
}

input[type=text] {
  font-size: 1rem;
  padding: 0.5em;
}
```

Mögliches Ergebnis:





vh und vw

CSS3 führte zwei Einheiten zur Darstellung der Größe ein.

- `vh` , was für `viewport height` ist relativ zu 1% der Viewport-Höhe
- `vw` , was für die `viewport width` `vw` steht, ist relativ zu 1% der Breite des Darstellungsbereichs

3

```
div {
  width: 20vw;
  height: 20vh;
}
```

Die Größe des div nimmt oben 20% der Breite und Höhe des Viewports ein

vmin und vmax

- **vmin** : Relativ zu 1 Prozent der kleineren Dimension des Ansichtsfensters
- **vmax** : Relativ zu 1 Prozent der größeren Dimension des Ansichtsfensters

Mit anderen Worten ist 1 `vmin` gleich dem kleineren von *1 vh* und *1 vw*

1 `vmax` ist gleich der größere von *1 vh* und *1 vw*

Hinweis : `vmax` wird nicht unterstützt in:

- beliebige Version von Internet Explorer
- Safari vor Version 6.1

mit Prozent%

Eine nützliche Einheit beim Erstellen einer responsiven Anwendung.

Seine Größe hängt vom übergeordneten Container ab.

Gleichung:

(Breite des übergeordneten Containers) * (Prozentsatz (%)) = Ausgabe

Zum Beispiel:

Elternteil hat **100px** Breite , während das *Kind* **50%** aufweist.

Bei der Ausgabe beträgt die Breite des *Kindes* die Hälfte (50%) der des *Elternteils* , was 50 **Pixel** beträgt.

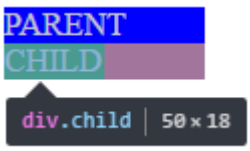
HTML

```
<div class="parent">
   PARENT
   <div class="child">
     CHILD
   </div>
</div>
```

CSS

```
<style>

*{
  color: #CCC;
}

.parent{
  background-color: blue;
  width: 100px;
}

.child{
  background-color: green;
  width: 50%;
}

</style>
```

AUSGABE



Längeneinheiten online lesen: https://riptutorial.com/de/css/topic/864/langeneinheiten

Kapitel 31: Layoutsteuerung

Syntax

- Anzeige: keine | inline | Block | Listenelement | Inline-Listenelement | Inline-Block | Inline-Tabelle | Tabelle | Tabellenzelle | Tabellenspalte | Tabellenspaltengruppe | Fußzeile-Gruppe | Tabellenkopfgruppe | Tischreihe | Tabellenzeilengruppe | flex | Inline-Flex | Gitter | Inline-Gitter | Einlauf | Rubin | Rubinbasis | Ruby-Text | Rubin-Behälter | Ruby-Text-Container | Inhalt;

Parameter

Wert	Bewirken
`none`	Blenden Sie das Element aus und verhindern Sie, dass es Platz einnimmt.
`block`	Element blockieren, 100% der verfügbaren Breite einnehmen, Element nach dem Element abbrechen.
`inline`	Inline-Element, keine Breite einnehmen, kein Element nach dem Element.
`inline-block`	Bei der Verwendung von speziellen Eigenschaften von Inline- und Blockelementen ist keine Unterbrechung möglich, sie kann jedoch Breite haben.
`inline-flex`	Zeigt ein Element als Flexcontainer auf Inline-Ebene an.
`inline-table`	Das Element wird als Inline-Tabelle angezeigt.
`grid`	Verhält sich wie ein Blockelement und ordnet seinen Inhalt dem Gittermodell entsprechend an.
`flex`	Verhält sich wie ein Blockelement und ordnet seinen Inhalt gemäß dem Flexbox-Modell.
`inherit`	Erben Sie den Wert vom übergeordneten Element.
`initial`	Setzen Sie den Wert auf den Standardwert zurück, der aus den in den HTML-Spezifikationen oder aus dem Browser- / Benutzerstandard-Stylesheet beschriebenen Verhalten stammt.
`table`	Verhält sich wie das HTML- `table` .
`table-cell`	Das Element sollte sich wie ein `<td>` -Element verhalten
`table-column`	Das Element sollte sich wie ein `<col>` -Element verhalten

Wert	Bewirken
`table-row`	Das Element sollte sich wie ein `<tr>` -Element verhalten
`list-item`	Lassen Sie das Element sich wie ein `<li>` -Element verhalten.

Examples

Die Anzeigeeigenschaft

Die Eigenschaft `display` CSS ist für die Steuerung des Layouts und des Flusses eines HTML-Dokuments von grundlegender Bedeutung. Die meisten Elemente haben eine `display` von entweder `block` oder `inline` (obwohl einige andere Elemente Standardwerte haben).

In der Reihe

Ein `inline` Element nimmt nur so viel Breite wie nötig ein. Es kann horizontal mit anderen Elementen desselben Typs gestapelt werden und darf keine anderen Nicht-Inline-Elemente enthalten.

```
<span>This is some <b>bolded</b> text!</span>
```

This is some **bolded** text!

Wie oben gezeigt, sind die beiden `inline` Elemente `<span>` und `<b>` `inline` (daher der Name) und unterbrechen den Textfluss nicht.

Block

Ein `block` nimmt die maximal verfügbare Breite seines übergeordneten Elements ein. Es beginnt mit einer neuen Zeile und beschränkt im Gegensatz zu `inline` Elementen nicht die Art der darin enthaltenen Elemente.

```
<div>Hello world!</div><div>This is an example!</div>
```

Hello world!
This is an example!

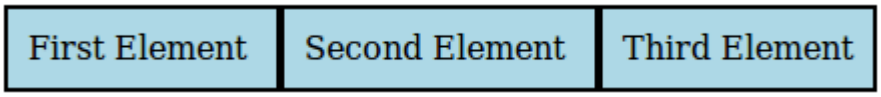
Das `div` Element ist standardmäßig auf Blockebene, und wie oben gezeigt, sind die beiden `block` vertikal gestapelt, und im Gegensatz zu den `inline` Elementen `inline` der Textfluss.

Inline-Block

Der `inline-block` Wert gibt uns das Beste aus beiden Welten: Er fügt das Element in den Textfluss ein und erlaubt die Verwendung von `padding` , `margin` , `height` und ähnlichen Eigenschaften, die auf `inline` Elemente keinen sichtbaren Effekt haben.

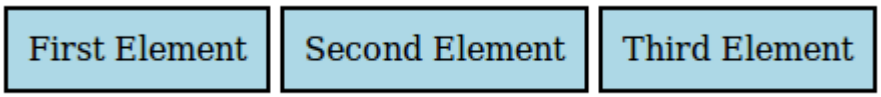
Elemente mit diesem Anzeigewert wirken so, als ob sie normaler Text wären, und werden daher von Regeln beeinflusst, die den Textfluss steuern, wie z. B. `text-align` . Standardmäßig werden sie auch auf die kleinstmögliche Größe geschrumpft, um ihren Inhalt unterzubringen.

```
<!--Inline: unordered list-->
<style>
li {
    display : inline;
    background : lightblue;
    padding:10px;

    border-width:2px;
    border-color:black;
    border-style:solid;
    }
</style>

<ul>
<li>First Element </li>
<li>Second Element </li>
<li>Third Element </li>
</ul>
```




```
<!--block: unordered list-->
<style>
li {
    display : block;
    background : lightblue;
    padding:10px;

    border-width:2px;
    border-color:black;
    border-style:solid;
    }
</style>

<ul>
<li>First Element </li>
<li>Second Element </li>
<li>Third Element </li>
</ul>
```


```
<!--Inline-block: unordered list-->
<style>
li {
    display : inline-block;
    background : lightblue;
    padding:10px;

    border-width:2px;
    border-color:black;
    border-style:solid;
    }

</style>

<ul>
<li>First Element </li>
<li>Second Element </li>
<li>Third Element </li>
</ul>
```



keiner

Ein Element, das seiner Anzeigeeigenschaft den Wert none zugewiesen hat, wird überhaupt nicht angezeigt.

Erstellen wir zum Beispiel ein div-Element mit der ID `myDiv` :

```
<div id="myDiv"></div>
```

Dies kann jetzt als nicht angezeigt markiert werden durch die folgende CSS-Regel:

```
#myDiv {
    display: none;
}
```

Wenn ein Element als `display:none;` Der Browser ignoriert alle anderen Layout-Eigenschaften für dieses bestimmte Element (sowohl `position` als auch `float`). Es wird keine Box für dieses Element gerendert und seine Existenz in html hat keinen Einfluss auf die Position der folgenden Elemente.

Beachten Sie, dass sich dies von der Einstellung der `visibility` auf `hidden` . `visibility: hidden;` einstellen `visibility: hidden;` Ein Element würde das Element nicht auf der Seite anzeigen, das Element würde jedoch beim Rendern den Platz beanspruchen, als wäre es sichtbar. Dies beeinflusst daher, wie die folgenden Elemente auf der Seite angezeigt werden.

Der Wert " `none` für die Anzeigeeigenschaft wird im Allgemeinen zusammen mit JavaScript verwendet, um Elemente nach Belieben anzuzeigen oder auszublenden. Dadurch müssen sie nicht wirklich gelöscht und neu erstellt werden.

Um eine alte Tabellenstruktur mit div zu erhalten

Dies ist die normale HTML-Tabellenstruktur

```
<style>
    table {
        width: 100%;
    }
</style>

<table>
  <tr>
    <td>
        I'm a table
    </td>
  </tr>
</table>
```

Sie können dieselbe Implementierung wie folgt durchführen

```
<style>
    .table-div {
        display: table;
    }
    .table-row-div {
        display: table-row;
    }
    .table-cell-div {
        display: table-cell;
    }
</style>

<div class="table-div>
  <div class="table-row-div>
    <div class="table-cell-div>
      I behave like a table now
    </div>
  </div>
</div>
```

Layoutsteuerung online lesen: https://riptutorial.com/de/css/topic/1473/layoutsteuerung

Kapitel 32: Listenstile

Syntax

- Listenstil: Listenstil-Typ | Listenstil-Position | list-style-image | initial | erben;

Parameter

Wert	Beschreibung
Listenart-Typ	die Art der Listenelementmarkierung.
Listenstil-Position	gibt an, wo die Markierung platziert werden soll
Listen-Style-Image	Gibt den Typ der Listenelementmarkierung an
Initiale	Setzt diese Eigenschaft auf ihren Standardwert
erben	erbt diese Eigenschaft von ihrem übergeordneten Element

Bemerkungen

Obwohl der `list-style-type` eigentlich eine Eigenschaft ist, die nur für Listenelemente gilt (normalerweise `<li>`), wird sie häufig für das Listentag (`<ol>` oder `<ul>`) angegeben. In diesem Fall erben die Listenelemente die Eigenschaft.

Examples

Aufzählungsart oder Nummerierung

Speziell für `<li>` -Tags in einer ungeordneten Liste (`<ul>`):

```
list-style: disc;               /* A filled circle (default) */
list-style: circle;             /* A hollow circle */
list-style: square;             /* A filled square */
list-style: '-';                /* any string */
```

Speziell für `<li>` -Tags innerhalb einer geordneten Liste (`<ol>`):

```
list-style: decimal;            /* Decimal numbers beginning with 1 (default) */
list-style: decimal-leading-zero;/* Decimal numbers padded by initial zeros (01, 02, 03, … 10)
*/
list-style: lower-roman;        /* Lowercase roman numerals (i., ii., iii., iv., ...) */
list-style: upper-roman;        /* Uppercase roman numerals (I., II., III., IV., ...) */
list-style-type: lower-greek;   /* Lowercase roman letters (α., β., γ., δ., ...) */
list-style-type: lower-alpha;   /* Lowercase letters (a., b., c., d., ...) */
```

```
list-style-type: lower-latin;    /* Lowercase letters (a., b., c., d., ...) */
list-style-type: upper-alpha;    /* Uppercase letters (A., B., C., D., ...) */
list-style-type: upper-latin;    /* Uppercase letters (A., B., C., D., ...) */
```

Nicht spezifisch:

```
list-style: none;                /* No visible list marker */
list-style: inherit;             /* Inherits from parent */
```

Bullet Position

Eine Liste besteht aus `<li>` Elementen innerhalb eines enthaltenden Elements (`<ul>` oder `<ol>`). Sowohl die Listenelemente als auch der Container können Ränder und Füllungen haben, die die genaue Position des Listenelementinhalts im Dokument beeinflussen. Die Standardwerte für den Rand und die Auffüllung können für jeden Browser unterschiedlich sein. Um dasselbe Layout browserübergreifend zu erhalten, müssen diese speziell festgelegt werden.

Jeder Listeneintrag erhält ein "Markierungsfeld", das den Aufzählungszeichen enthält. Dieses Feld kann entweder innerhalb oder außerhalb des Listenfelds platziert werden.

```
list-style-position: inside;
```

Platziert das Aufzählungszeichen innerhalb des `<li>` -Elements und drückt den Inhalt nach Bedarf nach rechts.

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

platziert das Aufzählungszeichen links vom `<li>` -Element. Wenn der Abstand des übergeordneten Elements nicht ausreicht, wird das Markierungsfeld nach links erweitert, auch wenn es von der Seite herunterfallen würde.

Anzeige des Ergebnisses der `inside` und `outside` : jsfiddle

Aufzählungszeichen / Zahlen entfernen

Manchmal sollte eine Liste keine Aufzählungspunkte oder Zahlen enthalten. Denken Sie in diesem Fall daran, den Rand und die Auffüllung anzugeben.

```
<ul>
    <li>first item</li>
    <li>second item</li>
</ul>
```

CSS

```
ul {
    list-style-type: none;
}
li {
```

```
    margin: 0;
    padding: 0;
}
```

Listenstile online lesen: https://riptutorial.com/de/css/topic/4215/listenstile

Kapitel 33: Medien-Anfragen

Syntax

- @media [nicht | nur] Mediatyp und (Medienfunktion) {/ * CSS-Regeln, die * /} gelten

Parameter

Parameter	Einzelheiten
`mediatype`	(Optional) Dies ist der Medientyp. Könnte alles im Bereich von `all` zu `screen` .
`not`	(Optional) Wendet CSS für diesen bestimmten Medientyp nicht an und gilt für alles andere.
`media feature`	Logik zur Identifizierung des Anwendungsfalls für CSS. Optionen unten beschrieben.
Medienfunktion	**Einzelheiten**
`aspect-ratio`	Beschreibt das Seitenverhältnis des Zielanzeigebereichs des Ausgabegeräts.
`color`	Gibt die Anzahl der Bits pro Farbkomponente des Ausgabegeräts an. Wenn das Gerät kein Farbgerät ist, ist dieser Wert Null.
`color-index`	Gibt die Anzahl der Einträge in der Farbnachschlagetabelle für das Ausgabegerät an.
`grid`	Bestimmt, ob das Ausgabegerät ein Grid-Gerät oder ein Bitmap-Gerät ist.
`height`	Die Höhenmedienfunktion beschreibt die Höhe der Rendering-Oberfläche des Ausgabegeräts.
`max-width`	CSS wird nicht auf eine Bildschirmbreite angewendet, die breiter als angegeben ist.
`min-width`	CSS wird nicht auf eine Bildschirmbreite angewendet, die schmaler als angegeben ist.
`max-height`	CSS wird nicht auf einer Bildschirmhöhe angewendet, die höher als angegeben ist.
`min-height`	CSS wird nicht auf einer Bildschirmhöhe angewendet, die kürzer ist als angegeben.

Parameter	Einzelheiten
`monochrome`	Gibt die Anzahl der Bits pro Pixel auf einem Monochromgerät (Graustufen) an.
`orientation`	CSS wird nur angezeigt, wenn das Gerät die angegebene Ausrichtung verwendet. Weitere Informationen finden Sie in den Anmerkungen.
`resolution`	Gibt die Auflösung (Pixeldichte) des Ausgabegeräts an.
`scan`	Beschreibt den Scanvorgang von Fernsehausgabegeräten.
`width`	Die Medienbreite-Funktion beschreibt die Breite der Rendering-Oberfläche des Ausgabegeräts (z. B. die Breite des Dokumentfensters oder die Breite des Seitenrahmens eines Druckers).
Veraltete Funktionen	**Einzelheiten**
`device-aspect-ratio`	**Deprecated** CSS wird nur auf Geräten angezeigt, deren Höhen- / Breitenverhältnis dem angegebenen Verhältnis entspricht. Dies ist eine `deprecated` Funktion und es kann nicht garantiert werden, dass sie funktioniert.
`max-device-width`	**Deprecated** Entspricht der `max-width` , misst jedoch die physische Bildschirmbreite und nicht die Anzeigebreite des Browsers.
`min-device-width`	**Deprecated** Entspricht `min-width` , misst jedoch die physische Bildschirmbreite und nicht die Anzeigebreite des Browsers.
`max-device-height`	**Deprecated** Entspricht der `max-height` , misst jedoch die physische Bildschirmbreite und nicht die Anzeigebreite des Browsers.
`min-device-height`	**Deprecated** Wie `min-height` , misst jedoch die physische Bildschirmbreite und nicht die Anzeigebreite des Browsers.

Bemerkungen

Medienabfragen werden in allen modernen Browsern unterstützt, einschließlich Chrome, Firefox, Opera und Internet Explorer 9 und höher.

Es ist wichtig zu beachten, dass die `orientation` nicht auf mobile Geräte beschränkt ist. Es basiert auf der Breite und Höhe des Ansichtsfensters (nicht Fenster oder Geräte).

Der Landschaftsmodus ist, wenn die Breite des Ansichtsfensters größer als die Höhe des Ansichtsfensters ist.

Der Porträtmodus ist, wenn die Höhe des Ansichtsfensters größer als die Breite des

Ansichtsfensters ist.

Dies bedeutet normalerweise, dass sich ein Desktop-Monitor im Querformat befindet. Manchmal kann es sich jedoch auch um ein Portrait handeln.

In den meisten Fällen melden mobile Geräte ihre Auflösung und nicht ihre tatsächliche Pixelgröße, die sich aufgrund der Pixeldichte unterscheiden kann. Um sie zu zwingen, ihre tatsächliche Pixelgröße anzugeben, fügen Sie Folgendes in Ihrem `head` Tag hinzu:

```
<meta name="viewport" content="width=device-width, initial-scale=1">
```

Examples

Basisbeispiel

```
@media screen and (min-width: 720px) {
    body {
        background-color: skyblue;
    }
}
```

Die obige Medienabfrage gibt zwei Bedingungen an:

1. Die Seite muss auf einem normalen Bildschirm angezeigt werden (keine gedruckte Seite, Projektor usw.).
2. Die Breite des Ansichtsports des Benutzers muss mindestens 720 Pixel betragen.

Wenn diese Bedingungen erfüllt sind, sind die Stile in der Medienabfrage aktiv und die Hintergrundfarbe der Seite ist himmelblau.

Medienabfragen werden dynamisch angewendet. Wenn beim Laden der Seite die in der Medienabfrage angegebenen Bedingungen erfüllt sind, wird das CSS angewendet, es wird jedoch sofort deaktiviert, wenn die Bedingungen nicht mehr erfüllt sind. Wenn die Bedingungen anfangs nicht erfüllt sind, wird das CSS erst dann angewendet, wenn die angegebenen Bedingungen erfüllt sind.

Wenn die Ansichtsanschlussbreite des Benutzers anfangs größer als 720 Pixel ist, der Benutzer jedoch die Breite des Browsers verkleinert, wird die Hintergrundfarbe nicht mehr himmelblau, sobald der Benutzer den Ansichtsanschluss auf weniger als 720 Pixel verkleinert Breite.

Verwenden Sie das Link-Tag

```
<link rel="stylesheet" media="min-width: 600px" href="example.css" />
```

Dieses Stylesheet wird noch heruntergeladen, jedoch nur auf Geräten mit einer Bildschirmbreite von mehr als 600px angewendet.

Medientyp

Medienabfragen haben einen optionalen `mediatype` Parameter. Dieser Parameter wird direkt nach der `@media` Deklaration (`@media mediatype`) `@media mediatype` Beispiel:

```
@media print {
    html {
        background-color: white;
    }
}
```

Der obige CSS-Code gibt dem DOM- `HTML` Element beim Drucken eine weiße Hintergrundfarbe.

Der `mediatype` Parameter hat ein optionales `not` oder `only` Präfix, das die Stile auf alles außer dem angegebenen Medientyp *oder* nur dem angegebenen Medientyp anwendet. Im folgenden Codebeispiel wird der Stil beispielsweise auf alle Medientypen mit Ausnahme des `print` .

```
@media not print {
    html {
        background-color: green;
    }
}
```

Und auf die gleiche Weise, um es nur auf dem Bildschirm anzuzeigen, kann dies verwendet werden:

```
@media only screen {
    .fadeInEffects {
        display: block;
    }
}
```

Die Liste der `mediatype` kann anhand der folgenden Tabelle besser verstanden werden:

Medientyp	Beschreibung
`all`	Auf alle Geräte anwenden
`screen`	Standardcomputer
`print`	Drucker im Allgemeinen. Wird verwendet, um Druckversionen von Websites zu gestalten
`handheld`	PDAs, Handys und Handgeräte mit kleinem Bildschirm
`projection`	Zur projizierten Präsentation, zum Beispiel Projektoren
`aural`	Sprachsysteme
`braille`	Braille-taktile Geräte

Medientyp	Beschreibung
`embossed`	Paged Braille-Drucker
`tv`	Fernsehgeräte
`tty`	Geräte mit einem Zeichenraster mit festem Abstand. Terminals, tragbare Geräte.

Verwenden von Medienabfragen zum Anpassen verschiedener Bildschirmgrößen

Beim responsive Webdesign handelt es sich häufig um Medienabfragen, bei denen es sich um CSS-Blöcke handelt, die nur ausgeführt werden, wenn eine Bedingung erfüllt ist. Dies ist hilfreich für responsives Webdesign, da Sie mithilfe von Medienabfragen andere CSS-Stile für die mobile Version Ihrer Website als für die Desktopversion angeben können.

```
@media only screen and (min-width: 300px) and (max-width: 767px) {
    .site-title {
        font-size: 80%;
    }

    /* Styles in this block are only applied if the screen size is atleast 300px wide, but no
more than 767px */
}

@media only screen and (min-width: 768px) and (max-width: 1023px) {
    .site-title {
        font-size: 90%;
    }

    /* Styles in this block are only applied if the screen size is atleast 768px wide, but no
more than 1023px */
}

@media only screen and (min-width: 1024px) {
    .site-title {
        font-size: 120%;
    }

    /* Styles in this block are only applied if the screen size is over 1024px wide. */
}
```

Breite vs. Sichtfenster

Bei der Verwendung von "width" bei Medienabfragen ist es wichtig, das Meta-Tag richtig einzustellen. Das grundlegende Meta-Tag sieht folgendermaßen aus und muss in das `<head>` -Tag eingefügt werden.

```
<meta name="viewport" content="width=device-width,initial-scale=1">
```

Warum ist das wichtig?

Basierend auf der Definition des MDN ist "width"

> Die Medienbreite-Funktion beschreibt die Breite der Rendering-Oberfläche des Ausgabegeräts (z. B. die Breite des Dokumentfensters oder die Breite des Seitenrahmens eines Druckers).

Was bedeutet das?

View-Port ist die Breite des Geräts selbst. Wenn Ihre Bildschirmauflösung eine Auflösung von 1280 x 720 angibt, ist die Breite des Ansichtsports "1280px".

Häufig weisen viele Geräte eine andere Pixelmenge zu, um ein Pixel anzuzeigen. Für ein Beispiel hat das iPhone 6 Plus eine Auflösung von 1242 x 2208. Die tatsächliche Breite des Ansichtsfensters und die Höhe des Ansichtsfensters beträgt jedoch 414 x 736. Das bedeutet, dass 3 Pixel zum Erstellen eines Pixels verwendet werden.

Wenn Sie das `meta` Tag jedoch nicht richtig eingestellt haben, wird versucht, Ihre Website mit ihrer nativen Auflösung anzuzeigen, die eine verkleinerte Ansicht (kleinere Texte und Bilder) ergibt.

Medienabfragen für Retina- und Nicht-Retina-Bildschirme

Obwohl dies nur für WebKit-basierte Browser funktioniert, ist dies hilfreich:

```
/* ----------- Non-Retina Screens ----------- */
@media screen
  and (min-width: 1200px)
  and (max-width: 1600px)
  and (-webkit-min-device-pixel-ratio: 1) {
}

/* ----------- Retina Screens ----------- */
@media screen
  and (min-width: 1200px)
  and (max-width: 1600px)
  and (-webkit-min-device-pixel-ratio: 2)
  and (min-resolution: 192dpi) {
}
```

Hintergrundinformation

Es gibt zwei Arten von Pixeln in der Anzeige. Eines sind die logischen Pixel und das andere sind die physischen Pixel. Meist bleiben die physischen Pixel immer gleich, da sie für alle Anzeigegeräte gleich sind. Die logischen Pixel ändern sich je nach Auflösung der Geräte, um Pixel mit höherer Qualität anzuzeigen. Das Gerätepixelverhältnis ist das Verhältnis zwischen physikalischen Pixeln und logischen Pixeln. Beispielsweise geben das MacBook Pro Retina, iPhone 4 und höher ein Gerätepixelverhältnis von 2 an, da die physikalische lineare Auflösung die doppelte logische Auflösung ist.

Der Grund, warum dies nur mit WebKit-basierten Browsern funktioniert, hat folgende Gründe:

- Das Lieferantenpräfix `-webkit-` vor der Regel.
- Dies wurde nicht in anderen Engines als WebKit und Blink implementiert.

Terminologie und Struktur

Medienabfragen ermöglichen die Anwendung von CSS-Regeln basierend auf dem Gerätetyp / Medium (z. B. Bildschirm, Druck oder Handheld), dem so genannten **Medientyp** . Zusätzliche Aspekte des Geräts werden mit **Medienfunktionen** beschrieben, wie z.

Allgemeine Struktur einer Medienanfrage

```
@media [...] {
    /* One or more CSS rules to apply when the query is satisfied */
}
```

Eine Medienanfrage, die einen Medientyp enthält

```
@media print {
    /* One or more CSS rules to apply when the query is satisfied */
}
```

Eine Medienabfrage, die einen Medientyp und eine Medienfunktion enthält

```
@media screen and (max-width: 600px) {
    /* One or more CSS rules to apply when the query is satisfied */
}
```

Eine Medienabfrage mit einer Medienfunktion (und einem impliziten Medientyp "Alle")

```
@media (orientation: portrait) {
    /* One or more CSS rules to apply when the query is satisfied */
}
```

Medienabfragen und IE8

Medienabfragen werden in IE8 und darunter überhaupt nicht unterstützt.

Eine Javascript-basierte Problemumgehung

Um die Unterstützung für IE8 hinzuzufügen, können Sie eine von mehreren JS-Lösungen verwenden. Zum Beispiel kann " **Antworten** " hinzugefügt werden, um die Unterstützung für Medienabfragen für IE8 nur mit dem folgenden Code hinzuzufügen:

```
<!--[if lt IE 9]>
<script
   src="respond.min.js">
</script>
<![endif]-->
```

CSS Mediaqueries ist eine andere Bibliothek, die dasselbe tut. Der Code zum Hinzufügen dieser Bibliothek zu Ihrem HTML-Code wäre identisch:

```
<!--[if lt IE 9]>
<script
   src="css3-mediaqueries.js">
</script>
<![endif]-->
```

Die Alternative

Wenn Sie keine JS-basierte Lösung mögen, sollten Sie auch ein IE <9-Stylesheet hinzufügen, in dem Sie Ihren Stil für IE <9 anpassen. Dazu sollten Sie Ihrem Code den folgenden HTML-Code hinzufügen:

```
<!--[if lt IE 9]>
<link rel="stylesheet" type="text/css" media="all" href="style-ielt9.css"/>
<![endif]-->
```

Hinweis :

Technisch gesehen ist es eine weitere Alternative: **CSS-Hacks verwenden,** um auf IE <9 zu zielen. Es hat die gleichen Auswirkungen wie ein IE <9-Stylesheet, aber Sie benötigen dafür kein separates Stylesheet. Ich empfehle diese Option jedoch nicht, da sie ungültigen CSS-Code erzeugen (was nur einer von mehreren Gründen ist, warum die Verwendung von CSS-Hacks heute allgemein missbilligt wird).

Medien-Anfragen online lesen: https://riptutorial.com/de/css/topic/317/medien-anfragen

Kapitel 34: Mehrere Spalten

Einführung

CSS ermöglicht die Definition des Elementinhalts in mehrere Spalten mit Lücken und Regeln dazwischen.

Bemerkungen

Das CSS-Modul für mehrspaltige Layouts der Stufe 1 ist seit dem 12. April 2011 eine W3C-Kandidatenempfehlung. Seitdem wurden einige kleinere Änderungen vorgenommen . Es wird angenommen, dass es sich in der stabilen Phase befindet .

Seit dem 3. Juli 2017 unterstützen die Browser Microsoft Internet Explorer 10 und 11 und Edge nur eine ältere Version der Spezifikation, die ein Herstellerpräfix verwendet.

Examples

Grundlegendes Beispiel

Betrachten Sie das folgende HTML-Markup:

```
<section>
  <p>Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et accusam et justo duo dolores et ea rebum.</p>
  <p> Stet clita kasd gubergren, no sea takimata sanctus est Lorem ipsum dolor sit amet. Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et accusam et justo duo dolores et ea rebum. Stet clita kasd gubergren, no sea takimata sanctus est Lorem ipsum dolor sit amet.</p>
  <p>Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et accusam et justo duo dolores et ea rebum. Stet clita kasd gubergren, no sea takimata sanctus est Lorem ipsum dolor sit amet.</p>
</section>
```

Mit dem folgenden CSS wird der Inhalt in drei Spalten aufgeteilt, die durch eine graue Spaltenregel von zwei Pixeln getrennt sind.

```
section {
  columns: 3;
  column-gap: 40px;
  column-rule: 2px solid gray;
}
```

Sehen Sie eine Live-Probe davon auf JSFiddle .

Erstellen Sie mehrere Spalten

```
<div class="content">
Lorem ipsum dolor sit amet, consectetuer adipiscing elit, sed diam nonummy nibh
euismod tincidunt ut laoreet dolore magna aliquam erat volutpat. Ut wisi enim
ad minim veniam, quis nostrud exerci tation ullamcorper suscipit lobortis nisl
ut aliquip ex ea commodo consequat. Duis autem vel eum iriure dolor in
hendrerit in vulputate velit esse molestie consequat, vel illum dolore eu
feugiat nulla facilisis at vero eros et accumsan et iusto odio dignissim qui
blandit praesent luptatum zzril delenit augue duis dolore te feugait nulla
facilisi. Nam liber tempor cum soluta nobis eleifend option congue nihil
imperdiet doming id quod mazim placerat facer possim assum.
</div>
```

Css

```
.content {
-webkit-column-count: 3; /* Chrome, Safari, Opera */
-moz-column-count: 3; /* Firefox */
column-count: 3;
}
```

Mehrere Spalten online lesen: https://riptutorial.com/de/css/topic/10688/mehrere-spalten

Kapitel 35: Objektanpassung und Platzierung

Bemerkungen

Die Eigenschaften `object-fit` und `object-position` werden von Internet Explorer nicht unterstützt.

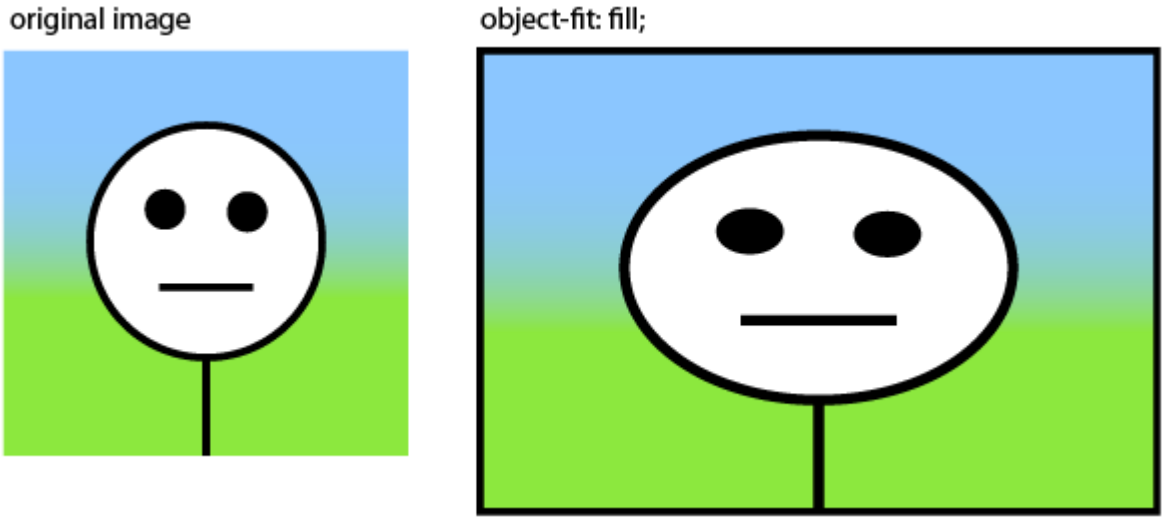
Examples

Objekt-fit

Die **Objektanpassungseigenschaft** definiert, wie ein Element in eine Box mit einer festgelegten Höhe und Breite passt. Object-fit wird normalerweise auf ein Bild oder Video angewendet und akzeptiert die folgenden fünf Werte:

FÜLLEN

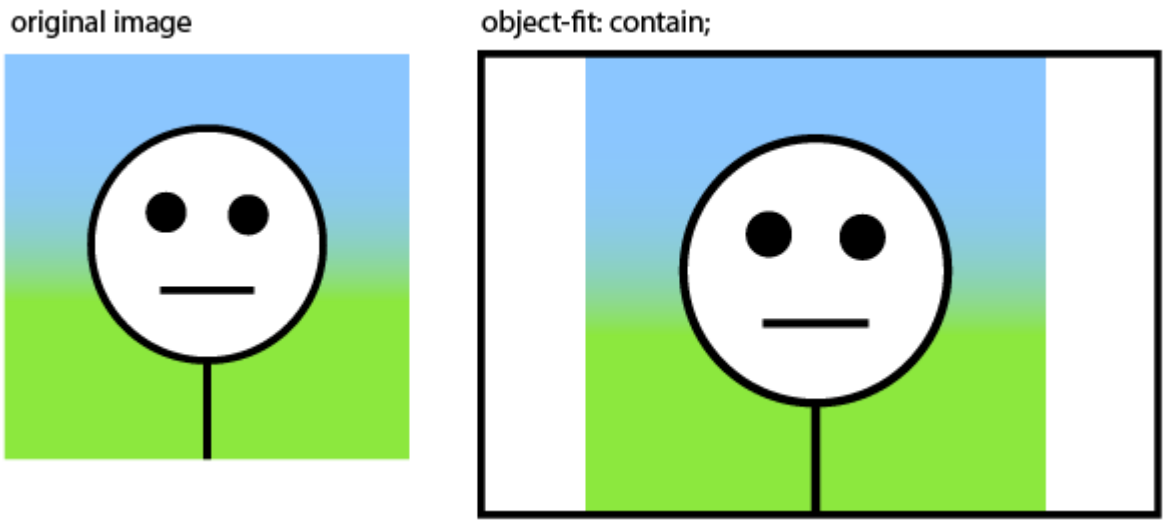
```
object-fit:fill;
```



Mit „Füllen“ wird das Bild ohne Berücksichtigung des ursprünglichen Seitenverhältnisses des Bildes an das Inhaltsfeld angepasst.

ENTHALTEN

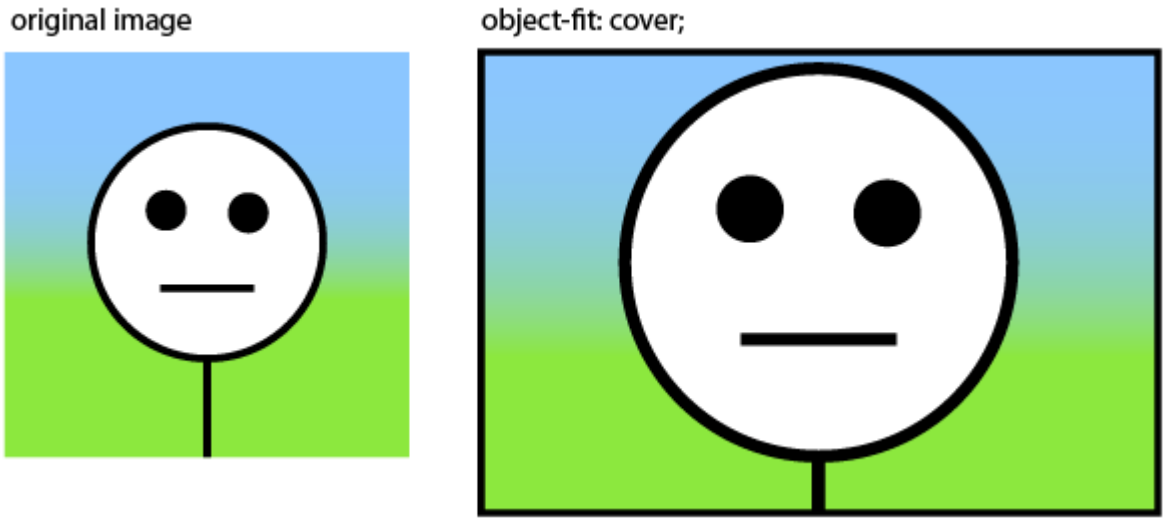
```
object-fit:contain;
```



"Inhalt" passt das Bild in die Höhe oder Breite des Felds ein, wobei das Seitenverhältnis des Bildes beibehalten wird.

ABDECKUNG

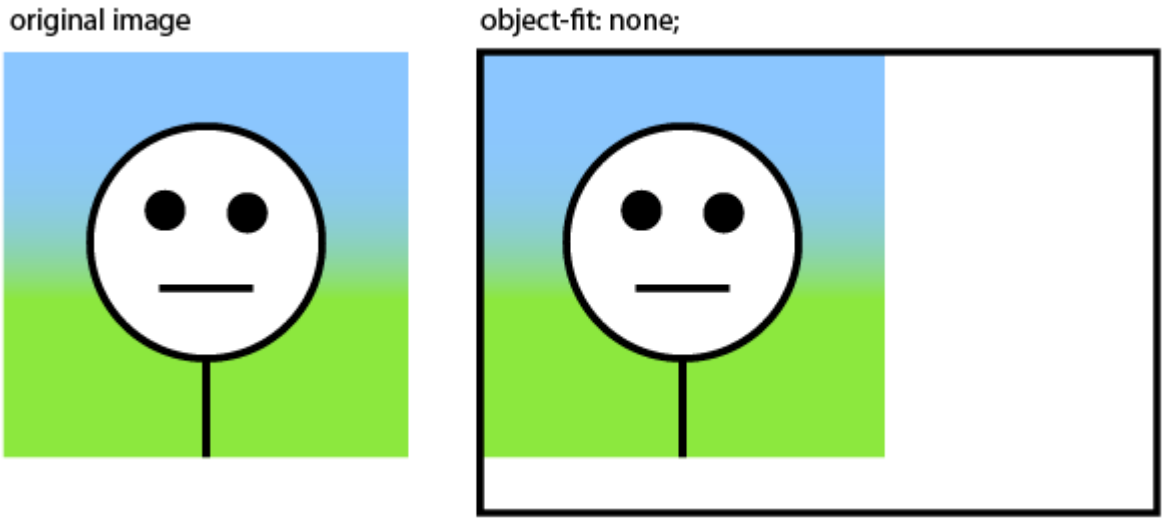
```
object-fit:cover;
```



Das Cover füllt die gesamte Box mit dem Bild. Das Bildseitenverhältnis wird beibehalten, aber das Bild wird auf die Abmessungen der Box zugeschnitten.

KEINER

```
object-fit:none;
```

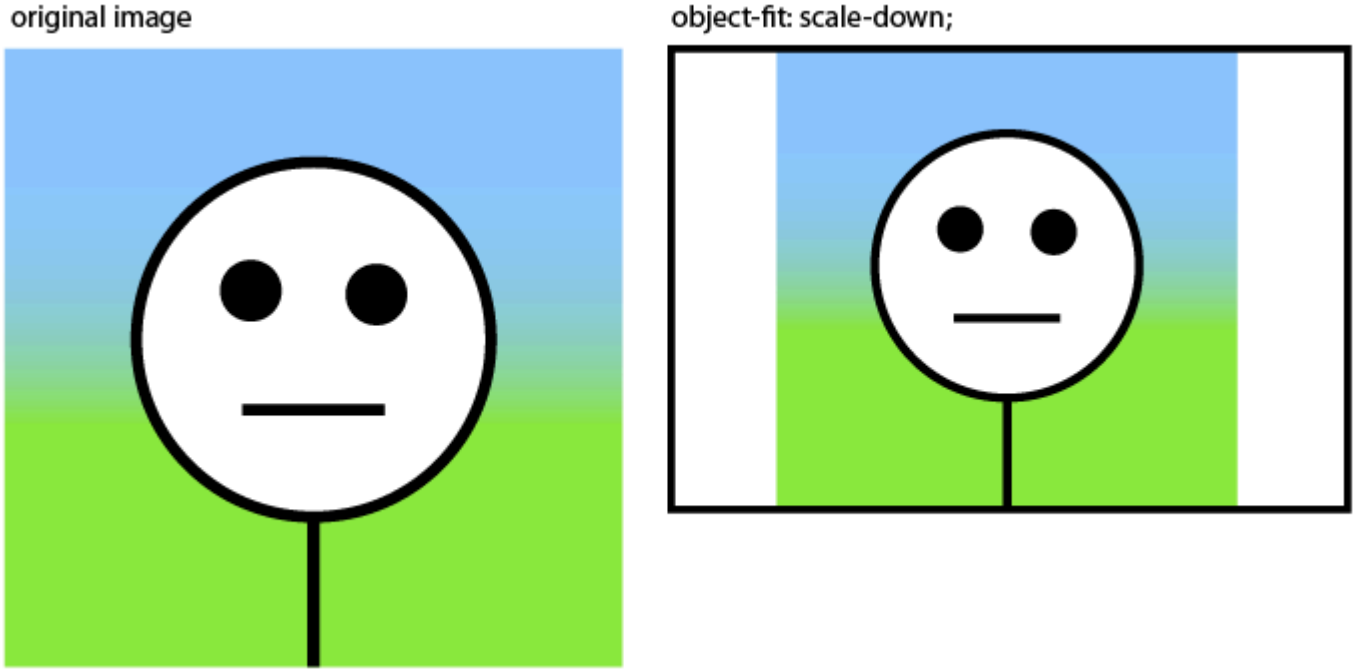


Keiner ignoriert die Größe des Feldes und wird nicht in der Größe geändert.

HERUNTERSKALIEREN

```
object-fit:scale-down;
```

Skalieren Sie das Objekt entweder als " `none` oder als " `contain` . Es zeigt an, welche Option zu einer kleineren Bildgröße führt.



Objektanpassung und Platzierung online lesen:
https://riptutorial.com/de/css/topic/5520/objektanpassung-und-platzierung

Kapitel 36: Opazität

Syntax

- Deckkraft: Zahl (* streng zwischen 0 und 1) | erben | initial | unset;

Bemerkungen

Wenn Sie keine Deckkraft anwenden möchten, können Sie stattdessen Folgendes verwenden:

Hintergrund: rgba (255, 255, 255, 0,6);

Ressourcen:

- MDN: https://developer.mozilla.org/de/docs/Web/CSS/opacity ;
- W3C-Transparenz: Die Eigenschaft "Deckkraft": https://www.w3.org/TR/css3-color/#transparency
- Browserunterstützung: http://caniuse.com/#feat=css-opacity

Examples

Deckkraft-Eigenschaft

Die Deckkraft eines Elements kann mit der `opacity` Eigenschaft festgelegt werden. Die Werte können zwischen `0.0` (transparent) und `1.0` (undurchsichtig) liegen.

Verwendungsbeispiel

```
<div style="opacity:0.8;">
    This is a partially transparent element
</div>
```

Eigentumswert	Transparenz
`opacity: 1.0;`	Undurchsichtig
`opacity: 0.75;`	25% transparent (75% undurchsichtig)
`opacity: 0.5;`	50% transparent (50% undurchsichtig)
`opacity: 0.25;`	75% transparent (25% undurchsichtig)
`opacity: 0.0;`	Transparent

IE-Kompatibilität für "Deckkraft"

Um die `opacity` in allen IE-Versionen zu verwenden, lautet die Reihenfolge:

```
.transparent-element {
  /* for IE 8 & 9 */
  -ms-filter:"progid:DXImageTransform.Microsoft.Alpha(Opacity=60)"; // IE8
  /* works in IE 8 & 9 too, but also 5, 6, 7 */
  filter: alpha(opacity=60); // IE 5-7
  /* Modern Browsers */
  opacity: 0.6;
}
```

Opazität online lesen: https://riptutorial.com/de/css/topic/2864/opazitat

Kapitel 37: Performance

Examples

Verwenden Sie Transformation und Deckkraft, um das Trigger-Layout zu vermeiden

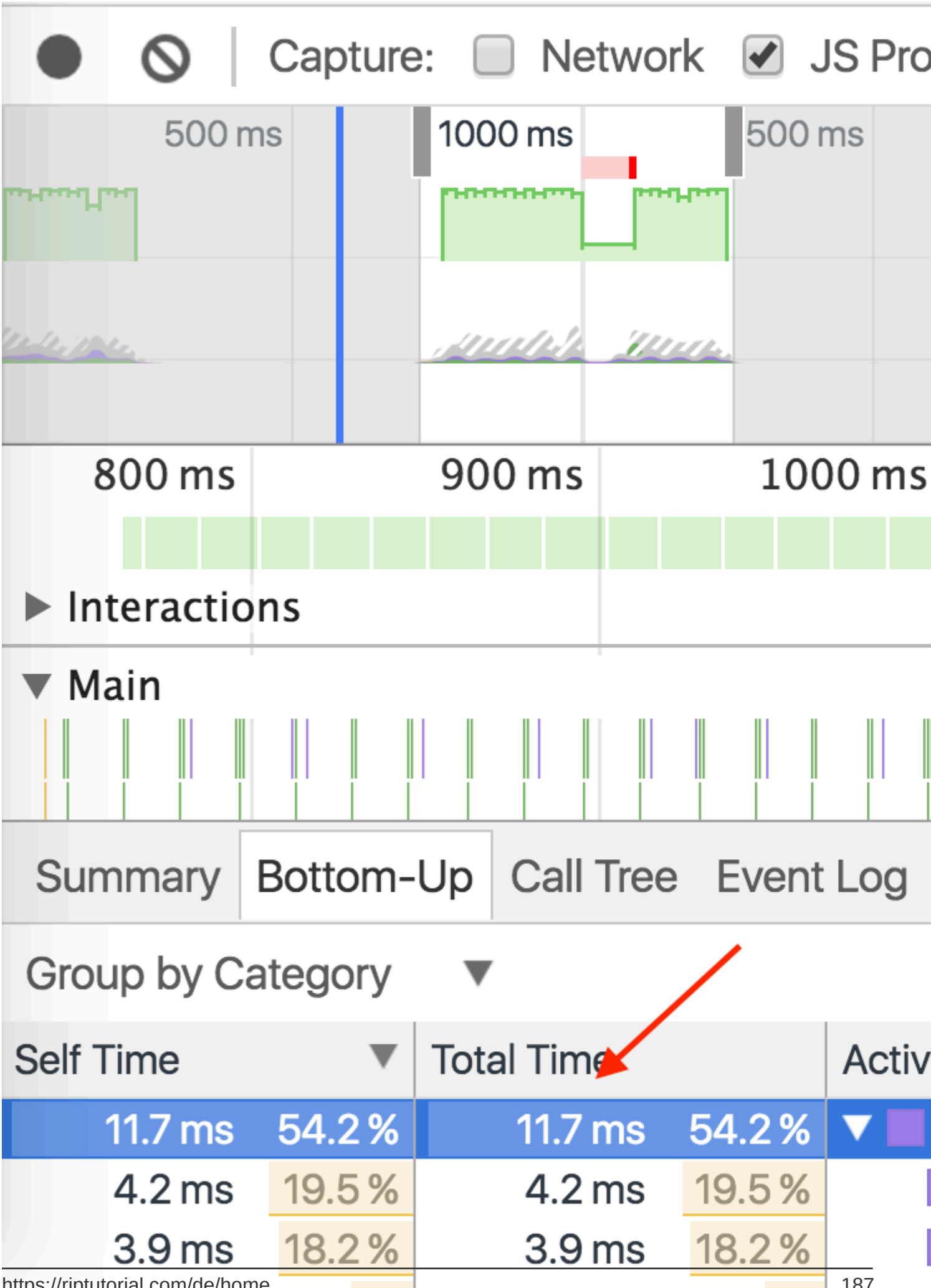
Wenn Sie einige CSS-Attribute ändern, wird der Browser dazu gezwungen, Stil und Layout synchron zu berechnen. Dies ist eine schlechte Sache, wenn Sie mit 60fps animieren müssen.

NICHT

Animieren Sie mit dem `left` und `top` Trigger-Layout.

```
#box {
  left: 0;
  top: 0;
  transition: left 0.5s, top 0.5s;
  position: absolute;
  width: 50px;
  height: 50px;
  background-color: gray;
}

#box.active {
  left: 100px;
  top: 100px;
}
```

Das Demo benötigte **11,7 ms** für das Rendern, **9,8 ms** für das Malen


Capture:
Network
JS Pro
500 ms
1000 ms
500 ms
800 ms
900 ms
1000 ms
Interactions
Main
Summary
Bottom-Up
Call Tree
Event Log
Group by Category
Self Time
Total Time
Activ
11.7 ms 54.2 %
11.7 ms 54.2 %
4.2 ms 19.5 %
4.2 ms 19.5 %
3.9 ms 18.2 %
3.9 ms 18.2 %
1.8 ms 8.3 %
1.8 ms 8.3 %

Kapitel 38: Polsterung

Syntax

- Polsterung: *Länge* | initial | erben | unset;
- padding-top: *länge* | initial | erben | unset;
- Polsterung rechts: *Länge* | initial | erben | unset;
- Polsterung unten: *Länge* | initial | erben | unset;
- Abfüllung links: *Länge* | initial | erben | unset;

Bemerkungen

> Die Eigenschaft padding legt den Abstand auf allen Seiten eines Elements fest. Der Auffüllbereich ist der Abstand zwischen dem Inhalt des Elements und seinem Rand. **Negative Werte sind nicht zulässig** .

1 : https://developer.mozilla.org/de/docs/Web/CSS/padding MDN

Siehe auch diese Frage : "Warum unterstützt CSS keine negative Auffüllung?" und seine Antworten.

Bitte berücksichtigen Sie auch das Box-Modell, wenn Sie die Polsterung verwenden. Abhängig von der Größe der Boxgröße kann die Auffüllung eines Elements die zuvor definierte Höhe / Breite eines Elements erhöhen oder nicht.

Verwandte Eigenschaften:

Spanne

Das Auffüllen von Inline-Elementen wird aufgrund der inhärenten Anzeigeeigenschaften von Inline-Elementen nur links und rechts des Elements und nicht oben und unten angewendet.

Examples

Auffüllen einer bestimmten Seite

Die Eigenschaft padding legt den Abstand auf allen Seiten eines Elements fest. Der Auffüllbereich ist der Abstand zwischen dem Inhalt des Elements und seinem Rand. Negative Werte sind nicht zulässig.

Sie können eine Seite einzeln angeben:

- `padding-top`
- `padding-right`
- `padding-bottom`
- `padding-left`

Der folgende Code würde eine Auffüllung von `5px` am oberen `5px` des div hinzufügen:

```
<style>
.myClass {
    padding-top: 5px;
}
</style>

<div class="myClass"></div>
```

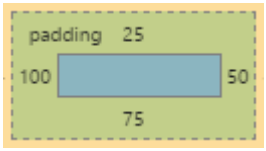
Padding-Kürzel

Die Eigenschaft padding legt den Abstand auf allen Seiten eines Elements fest. Der Auffüllbereich ist der Abstand zwischen dem Inhalt des Elements und seinem Rand. Negative Werte sind nicht zulässig.

Um das Hinzufügen von Padding zu jeder Seite einzeln zu speichern (`padding-top` , `padding-left` usw.), können Sie es wie folgt als Abkürzung schreiben:

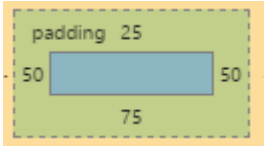
Vier Werte :

```
<style>
    .myDiv {
        padding: 25px 50px 75px 100px; /* top right bottom left; */
    }
</style>
<div class="myDiv"></div>
```



Drei Werte :

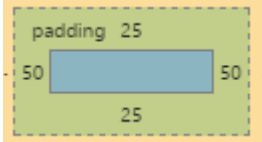
```
<style>
    .myDiv {
        padding: 25px 50px 75px; /* top left/right bottom */
    }
</style>
<div class="myDiv"></div>
```



Zwei Werte :

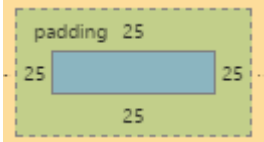
```
<style>
    .myDiv {
        padding: 25px 50px; /* top/bottom left/right */
    }
```

```
</style>
<div class="myDiv"></div>
```



Ein Wert :

```
<style>
    .myDiv {
        padding: 25px; /* top/right/bottom/left */
    }
</style>
<div class="myDiv"></div>
```



Polsterung online lesen: https://riptutorial.com/de/css/topic/1255/polsterung

Kapitel 39: Positionierung

Syntax

- Position: statisch | absolut | fest | relativ | sticky | initial | erben | unset;
- z-index: auto | *Nummer* | initial | erben;

Parameter

Parameter	Einzelheiten
statisch	Standardwert. Elemente werden in der Reihenfolge gerendert, wie sie im Dokumentenfluss angezeigt werden. Die Eigenschaften top, right, bottom, left und z-index gelten nicht.
relativ	Das Element wird relativ zu seiner normalen Position positioniert, also `left:20px` fügt der `left:20px` Position des Elements 20 Pixel hinzu
Fest	Das Element wird relativ zum Browserfenster positioniert
absolut	Das Element wird relativ zu seinem ersten positionierten (nicht statischen) Vorfahrenelement positioniert
Initiale	Setzt diese Eigenschaft auf ihren Standardwert.
erben	Übernimmt diese Eigenschaft von ihrem übergeordneten Element.
klebrig	Experimentelles Merkmal. Es verhält sich wie `position: static` innerhalb seines übergeordneten Elements, bis ein bestimmter Offset-Schwellenwert erreicht ist, und fungiert als `position: fixed` .
unset	Kombination von Anfang und Erbschaft. Mehr Infos hier .

Bemerkungen

Normaler Fluss ist der Fluss von Elementen, wenn die Position des Elements *statisch ist* .

1. Das Definieren der *Breite* ist vorteilhaft, da in manchen Fällen das Überlappen des Elementinhalts verhindert wird.

Examples

Fixierte Position

Wenn Sie die Position als fest definieren, können Sie ein Element aus dem Dokumentenfluss entfernen und seine Position relativ zum Browserfenster festlegen. Eine offensichtliche Verwendung ist, wenn wir möchten, dass etwas sichtbar ist, wenn wir zum Ende einer langen Seite scrollen.

```
#stickyDiv {
    position:fixed;
    top:10px;
    left:10px;
}
```

Überlappende Elemente mit Z-Index

Verwenden Sie die `z-index` -Eigenschaft, um die Standardpositionen der Stapelreihenfolge zu ändern (`position` auf `relative` , `absolute` oder `fixed`).

Je höher der Z-Index, desto höher wird er im Stapelkontext (auf der Z-Achse) platziert.

Beispiel

Im folgenden Beispiel wird ein Z-Indexwert von 3 grün dargestellt, ein Z-Index von 2 setzt Rot darunter, und ein Z-Index von 1 setzt Blau darunter.

HTML

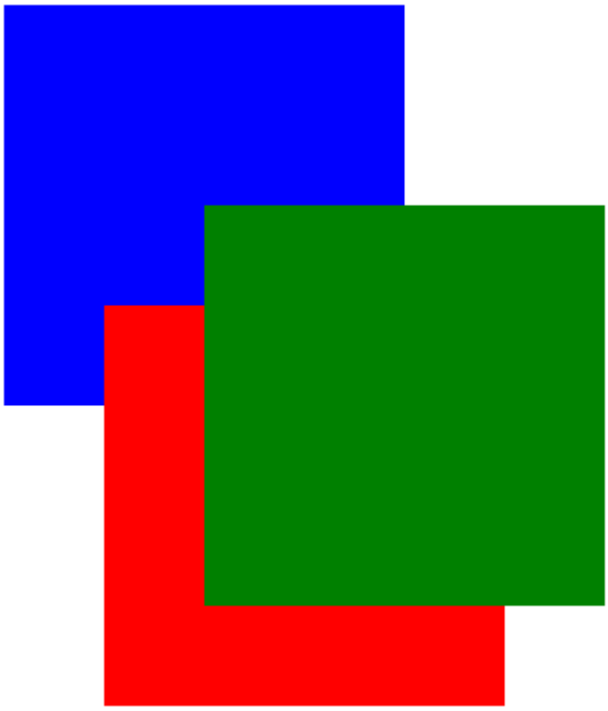
```
<div id="div1"></div>
<div id="div2"></div>
<div id="div3"></div>
```

CSS

```
div {
    position: absolute;
    height: 200px;
    width: 200px;
}
div#div1 {
    z-index: 1;
    left: 0px;
    top: 0px;
    background-color: blue;
}
div#div2 {
    z-index: 3;
    left: 100px;
    top: 100px;
    background-color: green;
}
div#div3 {
```

```
    z-index: 2;
    left: 50px;
    top: 150px;
    background-color: red;
}
```

Dies bewirkt den folgenden Effekt:



Siehe ein Arbeitsbeispiel bei JSFiddle .

Syntax

```
z-index: [ number ] | auto;
```

Parameter	Einzelheiten
`number`	Ein ganzzahliger Wert. Eine höhere Anzahl ist auf dem `z-index` Stack höher. `0` ist der Standardwert. Negative Werte sind zulässig.
`auto`	Gibt dem Element denselben Stapelungskontext wie sein übergeordnetes Element. (**Standardeinstellung**)

Bemerkungen

Alle Elemente werden in CSS in einer 3D-Achse angeordnet, einschließlich einer Tiefenachse, die

anhand der `z-index` Eigenschaft gemessen wird. `z-index` funktioniert nur bei positionierten Elementen: (siehe: Warum benötigt z-index eine definierte Position, um zu arbeiten?). Der einzige Wert, bei dem ignoriert wird, ist der Standardwert `static` .

Weitere Informationen zu den Eigenschaften von z-index und Stacking Contexts finden Sie in der CSS-Spezifikation in der geschichteten Präsentation und im Mozilla Developer Network .

Relative Position

Durch die relative Positionierung wird das Element relativ zu dem Bereich verschoben, in dem es sich im *normalen Fluss* befunden hätte. Offene Eigenschaften:

1. oben
2. links
3. Recht
4. Unterseite

werden verwendet, um anzugeben, wie weit das Element aus dem normalen Fluss bewegt werden soll.

```
.relpos{
    position:relative;
    top:20px;
    left:30px;
}
```

Dieser Code verschiebt das Feld, das das Element mit dem Attribut class = "relpos" 20px enthält, nach unten und 30px nach rechts von dem Punkt, an dem es im normalen Fluss wäre.

Absolute Position

Wenn die absolute Positionierung verwendet wird, wird die Box des gewünschten Elements aus dem *Normalfluss entfernt* und beeinflusst nicht mehr die Position der anderen Elemente auf der Seite. Offset-Eigenschaften:

1. oben
2. links
3. Recht
4. Unterseite

Geben Sie an, dass das Element in Bezug auf das nächste nicht statische Element angezeigt werden soll.

```
.abspos{
    position:absolute;
    top:0px;
    left:500px;
}
```

Dieser Code verschiebt das Feld, das das Element mit dem Attribut `class="abspos"` um 0px und

rechts um 500px relativ zu dem enthaltenen Element.

Statische Positionierung

Die Standardposition eines Elements ist `static` . Um MDN zu zitieren:

> Mit diesem Schlüsselwort kann das Element das normale Verhalten verwenden, d. H. Es wird an seiner aktuellen Position im Fluss angeordnet. Die Eigenschaften top, right, bottom, left und z-index gelten nicht.

```
.element{
  position:static;
}
```

Positionierung online lesen: https://riptutorial.com/de/css/topic/935/positionierung

Kapitel 40: Pseudo-Elemente

Einführung

Pseudo-Elemente werden wie Pseudo-Klassen einem CSS-Selektor hinzugefügt. Statt jedoch einen speziellen Zustand zu beschreiben, können Sie bestimmte Bereiche eines HTML-Elements bestimmen und gestalten.

Das :: Anfangsbuchstabe-Pseudoelement zielt beispielsweise nur auf den ersten Buchstaben eines Blockelements, das vom Selektor angegeben wird.

Syntax

- Selector :: Pseudoelement {Eigenschaft: Wert}

Parameter

Pseudoelement	Beschreibung
`::after`	Fügen Sie den Inhalt nach dem Inhalt eines Elements ein
`::before`	Fügen Sie den Inhalt vor dem Inhalt eines Elements ein
`::first-letter`	Wählt den ersten Buchstaben jedes Elements aus
`::first-line`	Wählt die erste Zeile jedes Elements aus
`::selection`	Stimmt mit dem Teil eines Elements überein, das von einem Benutzer ausgewählt wird
`::backdrop`	Wird verwendet, um einen Hintergrund zu erstellen, der das zugrunde liegende Dokument für ein Element im Stapel der obersten Ebene versteckt
`::placeholder`	Ermöglicht die Formatierung des Platzhaltertextes eines Formularelements (experimentell)
`::marker`	Zum Anwenden von Listenstilattributen auf ein bestimmtes Element (experimentell)
`::spelling-error`	Stellt ein Textsegment dar, das der Browser als falsch geschrieben markiert hat (experimentell).
`::grammar-error`	Stellt ein Textsegment dar, das vom Browser als grammatikalisch nicht korrekt gekennzeichnet wurde (experimentell).

Bemerkungen

- Manchmal werden Sie zwei Doppelpunkte (siehe `::`) anstelle von nur einem (`:`). Dies ist eine Möglichkeit , Pseudoklassen von Pseudo-Elemente zu trennen, aber einige ältere Browser wie Internet Explorer 8 unterstützt **nur** einzelne Doppelpunkt (`:`) für Pseudoelemente.
- In einem Selektor kann nur ein Pseudoelement verwendet werden. Es muss hinter den einfachen Selektoren in der Anweisung stehen.
- Pseudo-Elemente sind nicht Teil des DOMs und können daher nicht gezielt angesteuert werden durch `:hover` oder andere Benutzerereignisse.

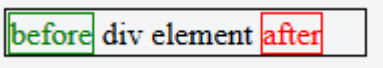
Examples

Pseudo-Elemente

Pseudo-Elemente werden Selektoren hinzugefügt, aber anstatt einen speziellen Status zu beschreiben, können Sie bestimmte Teile eines Dokuments formatieren.

Das `content` ist für das Rendern von Pseudoelementen erforderlich. Das Attribut kann jedoch einen leeren Wert haben (z. B. `content: ""`).

```
div::after {
  content: 'after';
  color: red;
  border: 1px solid red;
}

div {
  color: black;
  border: 1px solid black;
  padding: 1px;
}

div::before {
  content: 'before';
  color: green;
  border: 1px solid green;
}
```



Pseudoelemente in Listen

Pseudoelemente werden häufig verwendet, um das Aussehen von Listen zu ändern (meistens für ungeordnete Listen, `ul`).

Der erste Schritt besteht darin, die Standardaufzählungszeichen zu entfernen:

```
ul {
  list-style-type: none;
}
```

Dann fügen Sie das benutzerdefinierte Styling hinzu. In diesem Beispiel erstellen wir Verlaufsfelder für Aufzählungszeichen.

```
li:before {
  content: "";
  display: inline-block;
  margin-right: 10px;
  height: 10px;
  width: 10px;
  background: linear-gradient(red, blue);
}
```

HTML

```
<ul>
   <li>Test I</li>
   <li>Test II</li>
</ul>
```

Ergebnis

Pseudo-Elemente online lesen: https://riptutorial.com/de/css/topic/911/pseudo-elemente

Kapitel 41: Rand

Syntax

- **Rand**
- Rand: Randbreite Randart-Randfarbe | initial | erben;
- Rand oben: Randbreite Randart-Randfarbe | initial | erben;
- Rahmen unten: Rahmenbreite Rahmenfarbe | initial | erben;
- Rahmen links: Rahmenbreite Rahmenfarbe | initial | erben;
- Rahmen rechts: Rahmenbreite Rahmenfarbe | initial | erben;
- **Grenzstil**
- Bordürenstil: 1-4 keine | versteckt | gepunktet | gestrichelt | fest | doppelt | Nut | Grat | Einschub | Anfang | initial | erben;
- **Grenzradius**
- Grenzradius: 1-4 Länge | % / 1-4 Länge | % | initial | erben;
- Rand oben links Radius: Länge | % [Länge | %] | initial | erben;
- Rand oben rechts Radius: Länge | % [Länge | %] | initial | erben;
- Rand unten links Radius: Länge | % [Länge | %] | initial | erben;
- Rand unten rechts Radius: Länge | % [Länge | %] | initial | erben;
- **Rahmenbild**
- border-image: border-image-source border-image-slice [Rahmenbreite [Randbildausgang]]
- border-image-source: keine | Bild;
- Border-Image-Slice: 1-4 Anzahl | Prozentsatz [füllen]
- Randbildwiederholung: 1-2 Dehnung | wiederhole | rund | Platz
- **Grenzzusammenbruch**
- Grenzzusammenbruch: separate | zusammenbrechen | initial | erben

Bemerkungen

Verwandte Eigenschaften :

- Rand
- Rand unten
- Rahmenfarbe unten
- Rand unten links Radius
- Rand unten rechts Radius
- Border-Bottom-Style
- Randbreite
- Randfarbe
- Rahmenbild
- Border-Image-Beginn
- Rahmenbildwiederholung
- Border-Image-Slice
- Rahmenbildquelle
- Randbildbreite
- Grenze links
- Rahmenfarbe links
- Grenze-Links-Stil
- Randbreite links
- Grenzradius
- rechtsbündig
- Randfarbe rechts
- Grenze-Rechts-Stil
- Randbreite rechts
- Grenzstil
- Rand oben
- Bordürenfarbe

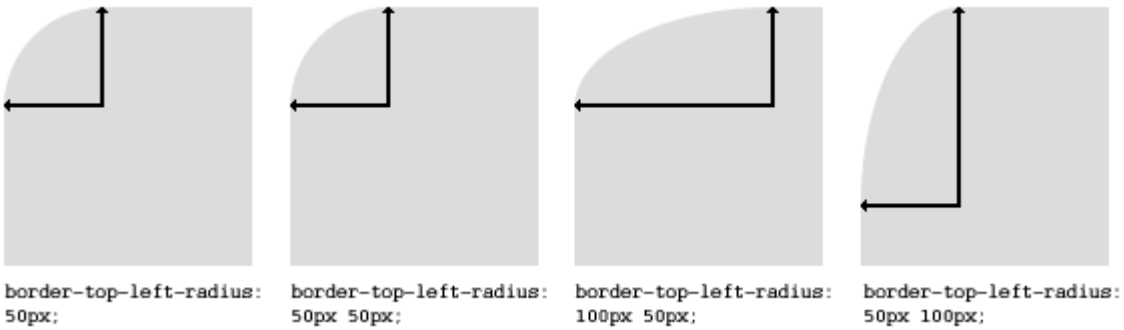
- Rand oben links Radius
- Rand oben rechts Radius
- Border-Top-Stil
- Bordürenbreite
- Rahmenbreite

Examples

Grenzradius

Mit der Eigenschaft "border-radius" können Sie die Form des Basisrahmenmodells ändern.

Jede Ecke eines Elements kann bis zu zwei Werte für den vertikalen und den horizontalen Radius dieser Ecke haben (für maximal 8 Werte).



Der erste Wertesatz definiert den horizontalen Radius. Der optionale zweite Satz von Werten, dem ein "/" vorangestellt ist, definiert den vertikalen Radius. Wenn nur ein Wertesatz angegeben wird, wird er sowohl für den vertikalen als auch für den horizontalen Radius verwendet.

```
border-radius: 10px 5% / 20px 25em 30px 35em;
```

Der `10px` ist der horizontale Radius von links oben und rechts unten. Und die `5%` sind der horizontale Radius von oben rechts und unten links. Die anderen vier Werte nach '/' sind die vertikalen Radien für oben links, oben rechts, unten rechts und unten links.

Wie bei vielen CSS-Eigenschaften können Abkürzungen für einen oder alle möglichen Werte verwendet werden. Sie können also einen bis acht Werte angeben. Mit der folgenden Abkürzung können Sie den horizontalen und den vertikalen Radius jeder Ecke auf denselben Wert einstellen:

HTML:

```
<div class='box'></div>
```

CSS:

```
.box {
    width: 250px;
    height: 250px;
    background-color: black;
    border-radius: 10px;
}
```

Der Randradius wird am häufigsten zum Konvertieren von Rahmenelementen in Kreise verwendet. Durch Festlegen des Grenzradius auf die Hälfte der Länge eines quadratischen Elements wird ein kreisförmiges Element erstellt:

```
.circle {
    width: 200px;
    height: 200px;
    border-radius: 100px;
}
```

Da der Grenzradius Prozentsätze akzeptiert, werden üblicherweise 50% verwendet, um zu vermeiden, dass der Grenzradiuswert manuell berechnet wird:

```
.circle {
    width: 150px;
    height: 150px;
    border-radius: 50%;
}
```

Wenn die Eigenschaften width und height nicht gleich sind, ist die resultierende Form eher oval als kreisförmig.

Browserspezifisches Beispiel für den Grenzradius:

```
  -webkit-border-top-right-radius: 4px;
  -webkit-border-bottom-right-radius: 4px;
  -webkit-border-bottom-left-radius: 0;
  -webkit-border-top-left-radius: 0;
  -moz-border-radius-topright: 4px;
  -moz-border-radius-bottomright: 4px;
  -moz-border-radius-bottomleft: 0;
  -moz-border-radius-topleft: 0;
  border-top-right-radius: 4px;
  border-bottom-right-radius: 4px;
  border-bottom-left-radius: 0;
  border-top-left-radius: 0;
```

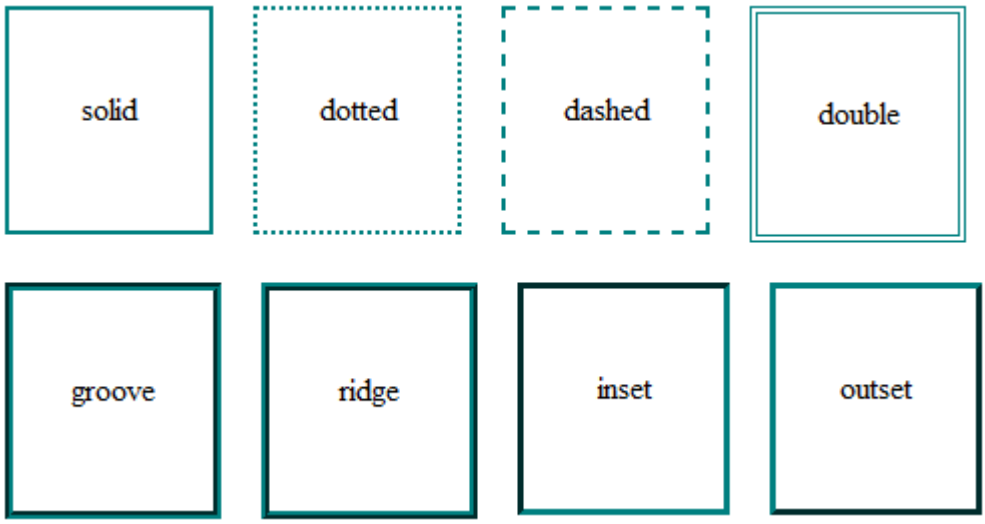
Grenzstil

Die `border-style` - Eigenschaft legt den Stil einer Grenze des Elements. Diese Eigenschaft kann einen bis vier Werte haben (für jede Seite des Elements einen Wert.)

Beispiele:

```
border-style: dotted;
```

```
border-style: dotted solid double dashed;
```



`border-style` können die Werte auch `none` und `hidden` . Sie haben den gleichen Effekt, mit Ausnahme der `hidden` Arbeiten für die Lösung von Grenzkonflikten für `<table>` -Elemente. In einer `<table>` mit mehreren Grenzen hat `none` die niedrigste Priorität (dh, in einem Konflikt würde die Grenze angezeigt), und `hidden` hat die höchste Priorität (dh in einem Konflikt würde die Grenze nicht angezeigt).

Grenze (Abkürzungen)

In den meisten Fällen möchten Sie mehrere Rahmeneigenschaften (`border-width` , `border-style` und `border-color`) für alle Seiten eines Elements definieren.

Anstatt zu schreiben:

```
border-width: 1px;
border-style: solid;
border-color: #000;
```

Sie können einfach schreiben:

```
border: 1px solid #000;
```

Diese Abkürzungen stehen auch für jede Seite eines Elements zur Verfügung: `border-top` , `border-left` , `border-right` und `border-bottom` . So können Sie tun:

```
border-top: 2px double #aaaaaa;
```

Rahmenbild

Mit der Eigenschaft `border-image` haben Sie die Möglichkeit, ein Bild anstelle von normalen Randstilen festzulegen.

Ein `border-image` Wesentlichen aus a

- `border-image-source` : Der Pfad zum zu verwendenden Bild
- `border-image-slice` : Bestimmt den Abstand, mit dem das Bild in **neun Bereiche unterteilt wird** (vier **Ecken** , vier **Kanten** und eine **Mitte**).
- Rahmenbild `border-image-repeat` : Gibt an, wie die Bilder für die Seiten und die Mitte des Rahmenbilds skaliert werden

Betrachten Sie das folgende Beispiel, wobei border.png ein Bild mit 90x90 Pixeln ist:

```
border-image: url("border.png") 30 stretch;
```

Das Bild wird in neun Bereiche mit 30x30 Pixeln aufgeteilt. Die Kanten werden als Ecken der Umrandung verwendet, während die Seite dazwischen verwendet wird. Wenn das Element höher / breiter als 30 Pixel ist, wird dieser Teil des Bildes **gestreckt** . Der mittlere Teil des Bildes ist standardmäßig transparent.

Grenze- [links | rechts | oben | unten]

Mit der `border-[left|right|top|bottom]` einer bestimmten Seite eines Elements einen Rand hinzufügen.

Wenn Sie beispielsweise einen Rand an der linken Seite eines Elements hinzufügen möchten, können Sie Folgendes tun:

```
#element {
    border-left: 1px solid black;
}
```

Grenzzusammenbruch

Die `border-collapse` Eigenschaft gilt nur für `table` (und Elemente, die als `display: table` oder `inline-table` angezeigt werden) und legt fest, ob die Tabellenränder wie im Standard-HTML-Code zu einem einzigen Rand reduziert oder getrennt werden.

```
table {
  border-collapse: separate; /* default */
  border-spacing: 2px; /* Only works if border-collapse is separate */
}
```

Siehe auch Tabellen - Dokumentationseintrag für Border-Collapse

Mehrere Grenzen

Umriss verwenden:

```
.div1{
  border: 3px solid black;
  outline: 6px solid blue;
  width: 100px;
  height: 100px;
  margin: 20px;
```

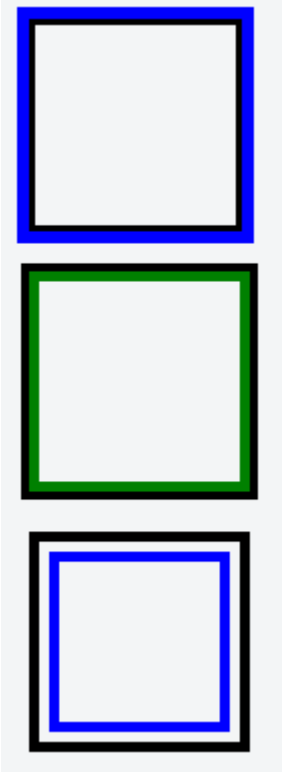
```
}
```

Box-Schatten verwenden:

```
.div2{
  border: 5px solid green;
  box-shadow: 0px 0px 0px 4px #000;
  width: 100px;
  height: 100px;
  margin: 20px;
}
```

Verwenden eines Pseudoelements:

```
.div3 {
  position: relative;
  border: 5px solid #000;
  width: 100px;
  height: 100px;
  margin: 20px;
}
.div3:before {
  content: " ";
  position: absolute;
  border: 5px solid blue;
  z-index: -1;
  top: 5px;
  left: 5px;
  right: 5px;
  bottom: 5px;
}
```



http://jsfiddle.net/MadalinaTn/bvqpcohm/2/

Erstellen eines mehrfarbigen Rahmens mit Rahmenbild

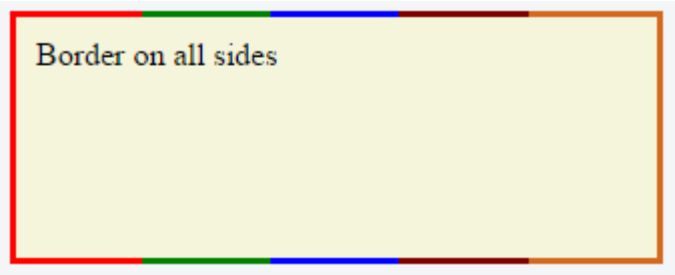
CSS

```
.bordered {
  border-image: linear-gradient(to right, red 20%, green 20%, green 40%, blue 40%, blue 60%,
maroon 60%, maroon 80%, chocolate 80%); /* gradient with required colors */
  border-image-slice: 1;
}
```

HTML

```
<div class='bordered'>Border on all sides</div>
```

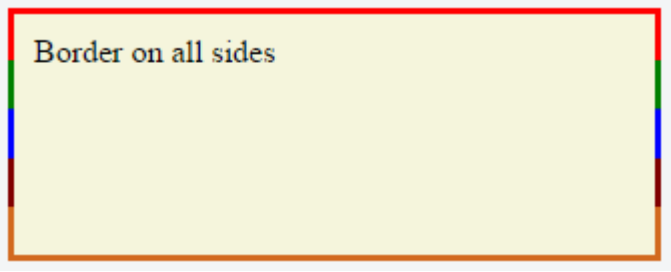
Das obige Beispiel würde einen Rand ergeben, der aus 5 verschiedenen Farben besteht. Die Farben werden durch einen `linear-gradient` (weitere Informationen zu Farbverläufen finden Sie in den Dokumenten). Weitere Informationen zur Eigenschaft `border-image-slice` im Beispiel für `border-image` auf derselben Seite.



(*Hinweis: Dem Element wurden zu Darstellungszwecken zusätzliche Eigenschaften hinzugefügt.*)

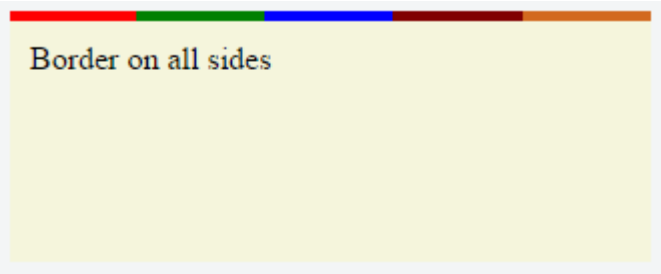
Sie hätten bemerkt, dass der linke Rand nur eine einzige Farbe hat (die Startfarbe des Farbverlaufs), während der rechte Rand nur eine einzige Farbe (die Endfarbe des Farbverlaufs) hat. Dies liegt an der Art und Weise, wie die Eigenschaften für die Rahmenabbildung funktionieren. Es ist, als würde der Farbverlauf auf das gesamte Feld angewendet, und dann werden die Farben in den Bereichen für das Auffüllen und den Inhalt maskiert, sodass der Eindruck entsteht, als habe nur der Rand den Farbverlauf.

Welche Umrandung (en) eine einzige Farbe haben, hängt von der Gradientendefinition ab. Wenn der Farbverlauf `to right` , ist der linke Rand die Startfarbe des Farbverlaufs und der rechte Rand die Endfarbe. Bei einem `to bottom` Farbverlauf wäre der obere Rand die Startfarbe des Farbverlaufs und der untere Rand die Endfarbe. Unten ist der Ausgang eines `to bottom` 5 farbigen Gradienten.



Wenn der Rand nur auf bestimmten Seiten des Elements erforderlich ist, kann die Eigenschaft `border-width` wie jeder andere normale Rand verwendet werden. Wenn Sie beispielsweise den folgenden Code hinzufügen, wird nur ein Rand oben auf dem Element angezeigt.

```
border-width: 5px 0px 0px 0px;
```



Beachten Sie, dass jedes Element mit `border-image` Eigenschaft `border-image` **Randradius nicht** `border-radius` (dh der Rand wird nicht gekrümmt). Dies basiert auf der folgenden Aussage in der Spezifikation:

> Die Hintergründe einer Box, aber nicht ihr Rahmenbild, werden auf die entsprechende Kurve (durch den 'Hintergrundclip' festgelegt) beschnitten.

Rand online lesen: https://riptutorial.com/de/css/topic/2160/rand

Kapitel 42: Ränder

Syntax

- Rand: <*oben & rechts & unten & links*> ;
- Rand: <*oben*> , <*links und rechts*> , <*unten*> ;
- Rand: <*oben & unten*> , <*links & rechts*> ;
- Rand: <*oben*> , <*rechts*> , <*unten*> , <*links*> ;
- Rand oben: <*oben*> ;
- Rand rechts: <*rechts*> ;
- Rand unten: <*unten*> ;
- Rand links: <*links*> ;

Parameter

Parameter	Einzelheiten
0	Marge auf keine setzen
Auto	Wird zum Zentrieren verwendet, indem auf jeder Seite gleichmäßig Werte eingestellt werden
Einheiten (zB px)	Eine Liste der gültigen Einheiten finden Sie im Parameterabschnitt in Units
erben	Margin-Wert vom übergeordneten Element übernehmen
Initiale	auf den ursprünglichen Wert zurücksetzen

Bemerkungen

Mehr zu "Collapsing Margins": hier .

Examples

Margin auf einer bestimmten Seite anwenden

Richtungsspezifische Eigenschaften

Mit CSS können Sie eine bestimmte Seite angeben, auf die ein Rand angewendet werden soll. Die vier Eigenschaften, die für diesen Zweck bereitgestellt werden, sind:

- `margin-left`

- `margin-right`
- `margin-top`
- `margin-bottom`

Der folgende Code würde einen Rand von 30 Pixeln auf die linke Seite des ausgewählten div anwenden. *Zeige Ergebnis*

HTML

```
<div id="myDiv"></div>
```

CSS

```
#myDiv {
    margin-left: 30px;
    height: 40px;
    width: 40px;
    background-color: red;
}
```

Parameter	Einzelheiten
Rand links	Die Richtung, in die der Rand angewendet werden soll.
30px	Die Breite des Randes.

Festlegen der Richtung mit der Shorthand-Eigenschaft

Die Standard- `margin` Eigenschaft kann erweitert werden, um für jede Seite der ausgewählten Elemente unterschiedliche Breiten festzulegen. Die Syntax dafür ist wie folgt:

```
margin: <top> <right> <bottom> <left>;
```

Im folgenden Beispiel wird ein Rand mit einer Breite von Null auf den oberen Rand des div, ein Rand von 10 Pixeln auf der rechten Seite, ein Rand von 50 Pixeln auf der linken Seite und ein Rand von 100 Pixeln auf der linken Seite angewendet. *Zeige Ergebnis*

HTML

```
<div id="myDiv"></div>
```

CSS

```
#myDiv {
    margin: 0 10px 50px 100px;
    height: 40px;
```

```
    width: 40px;
    background-color: red;
}
```

Marge kollabiert

Wenn sich zwei Ränder vertikal berühren, werden sie zusammengeklappt. Wenn sich zwei Ränder horizontal berühren, kollabieren sie nicht.

Beispiel für angrenzende vertikale Ränder:

Beachten Sie die folgenden Stile und Markierungen:

```
div{
    margin: 10px;
}
```

```
<div>
    some content
</div>
<div>
    some more content
</div>
```

Sie liegen 10px auseinander, da die vertikalen Ränder über den einen und den anderen fallen. (Der Abstand ist nicht die Summe zweier Ränder.)

Beispiel für angrenzende horizontale Ränder:

Beachten Sie die folgenden Stile und Markierungen:

```
span{
    margin: 10px;
}
```

```
<span>some</span><span>content</span>
```

Sie liegen 20px auseinander, da die horizontalen Ränder nicht über den einen und den anderen fallen. (Der Abstand wird die Summe zweier Ränder sein.)

Überlappung mit verschiedenen Größen

```
.top{
    margin: 10px;
}
.bottom{
    margin: 15px;
}
```

```
<div class="top">
    some content
```

```
</div>
<div class="bottom">
    some more content
</div>
```

Diese Elemente sind um 15 Pixel vertikal voneinander beabstandet. Die Ränder überlappen sich soweit wie möglich, der größere Abstand bestimmt jedoch den Abstand zwischen den Elementen.

Überlappender Rand Gotcha

```
.outer-top{
    margin: 10px;
}
.inner-top{
    margin: 15px;
}
.outer-bottom{
    margin: 20px;
}
.inner-bottom{
    margin: 25px;
}
```

```
<div class="outer-top">
    <div class="inner-top">
        some content
    </div>
</div>
<div class="outer-bottom">
    <div class="inner-bottom">
        some more content
    </div>
</div>
```

Wie groß ist der Abstand zwischen den beiden Texten? (schweben, um die Antwort zu sehen)

> Der Abstand beträgt 25px. Da sich alle vier Margen gegenseitig berühren, werden sie zusammenbrechen und nutzen dabei den größten Rand der vier Margen.

Wie sieht es aus, wenn wir der Markierung oben einige Rahmen hinzufügen.

```
div{
    border: 1px solid red;
}
```

Wie groß ist der Abstand zwischen den beiden Texten? (schweben, um die Antwort zu sehen)

> Der Abstand beträgt 59px! Jetzt berühren sich nur die Ränder von .outer-top und .outer-bottom und sind die einzigen eingestürzten Ränder. Die verbleibenden Ränder sind durch die Ränder getrennt. Wir haben also 1px + 10px + 1px + ~~15px~~ + 20px + 1px + 25px + 1px. (Die 1px sind die Grenzen ...)

Ränder zwischen übergeordneten und untergeordneten Elementen reduzieren:

HTML:

```
<h1>Title</h1>
<div>
  <p>Paragraph</p>
</div>
```

CSS

```
h1 {
  margin: 0;
  background: #cff;
}
div {
  margin: 50px 0 0 0;
  background: #cfc;
}
p {
  margin: 25px 0 0 0;
  background: #cf9;
}
```

Im obigen Beispiel gilt nur die größte Marge. Sie haben möglicherweise erwartet, dass der Absatz 60px von h1 entfernt sein würde (da das div-Element eine Randspitze von 40px hat und der p eine 20px-Randspitze hat). Dies geschieht nicht, weil die Ränder zu einem Rand zusammenfallen.

Elemente auf einer Seite mit Rand horizontal zentrieren

Solange das Element ein **Block** ist und es einen **explizit festgelegten Breitenwert hat** , können Ränder verwendet werden, um Blockelemente auf einer Seite horizontal zu zentrieren.

Wir fügen einen Breitenwert hinzu, der niedriger als die Breite des Fensters ist, und die automatische Eigenschaft des Rands verteilt dann den verbleibenden Platz nach links und nach rechts:

```
#myDiv {
 width:80%;
 margin:0 auto;
}
```

In dem obigen Beispiel verwenden wir die Kurz `margin` Erklärung ersten Satzes `0` in den oberen und unteren Randwerte (obwohl dies einen beliebigen Wert sein könnte) und dann verwenden wir `auto` der Browser lassen auf der linken und rechten Randwerte den Raum automatisch zuweisen.

Im obigen Beispiel ist das #myDiv-Element auf 80% Breite festgelegt, sodass nur noch 20% übrig bleiben. Der Browser verteilt diesen Wert an die verbleibenden Seiten.

(100% - 80%) / 2 = 10%

Margin Property Vereinfachung

```
p {
    margin:1px;                 /* 1px margin in all directions */

    /*equals to:*/

    margin:1px 1px;

    /*equals to:*/

    margin:1px 1px 1px;

    /*equals to:*/

    margin:1px 1px 1px 1px;
}
```

Ein anderes Beispiel:

```
p{
    margin:10px 15px;        /* 10px margin-top & bottom And 15px margin-right & left*/

    /*equals to:*/

    margin:10px 15px 10px 15px;

    /*equals to:*/

    margin:10px 15px 10px;
    /* margin left will be calculated from the margin right value (=15px) */
}
```

Negative Margen

Margin ist eine der wenigen CSS-Eigenschaften, die auf negative Werte gesetzt werden kann. Diese Eigenschaft kann verwendet werden, **um Elemente ohne absolute Positionierung** zu **überlappen** .

```
div{
  display: inline;
}

#over{
  margin-left: -20px;
}

<div>Base div</div>
<div id="over">Overlapping div</div>
```

Beispiel 1:

Es ist offensichtlich, anzunehmen, dass der prozentuale Wert des Randes zum `margin-left` und `margin-right` relativ zu seinem übergeordneten Element ist.

```
.parent {
```

```
    width : 500px;
    height: 300px;
}

.child {
    width : 100px;
    height: 100px;
    margin-left: 10%;  /* (parentWidth * 10/100) => 50px */
}
```

Dies ist jedoch nicht der Fall, wenn es zu `margin-top` und `margin-bottom` . Diese beiden Eigenschaften in Prozent beziehen sich nicht auf die Höhe des übergeordneten Containers, sondern auf die **Breite** des übergeordneten Containers.

So,

```
.parent {
    width : 500px;
    height: 300px;
}

.child {
    width : 100px;
    height: 100px;
    margin-left: 10%;  /* (parentWidth * 10/100) => 50px  */
    margin-top: 20%;   /* (parentWidth * 20/100) => 100px */
}
```

Ränder online lesen: https://riptutorial.com/de/css/topic/305/rander

Kapitel 43: Säulen

Syntax

- Spaltenanzahl: Auto | Nummer | Vererbung | Anfang | Unset;
- Spaltenbreite: Auto | Länge;
- Spalte: [Spaltenbreite] | [Spaltenanzahl];
- Spaltenbereich: keine | alle | erben | initial | unset;
- Spaltenlücke: normal | length | erben | initial | unset;
- Spaltenfüllung: Auto | Balance | Vererbung | Intial | Unset;
- Spaltenregelfarbe: Farbe | erben | initial | unset;
- Spaltenregel-Stil: keine | ausgeblendet | gepunktet | gestrichelt | fest | doppelt | Nut | Kante | Einfügung | Anfang | erben | initial |
- Spaltenregelbreite: dünn | mittel | dick | Länge | erben | initial | unset;
- Spaltenregel: [Spaltenregelbreite] | [Colum-Regelstil] | [Spaltenregelfarbe];
- break-after: auto | always | left | right | recto | verso | page | column | region | vermeiden |
- break-before: auto | immer | left | right | recto | verso | page | column | region | vermeiden |
- Break-Inside: Auto | Vermeiden | Vermeiden-Seite | Vermeiden-Spalte | Vermeiden-Region;

Examples

Einfaches Beispiel (Spaltenanzahl)

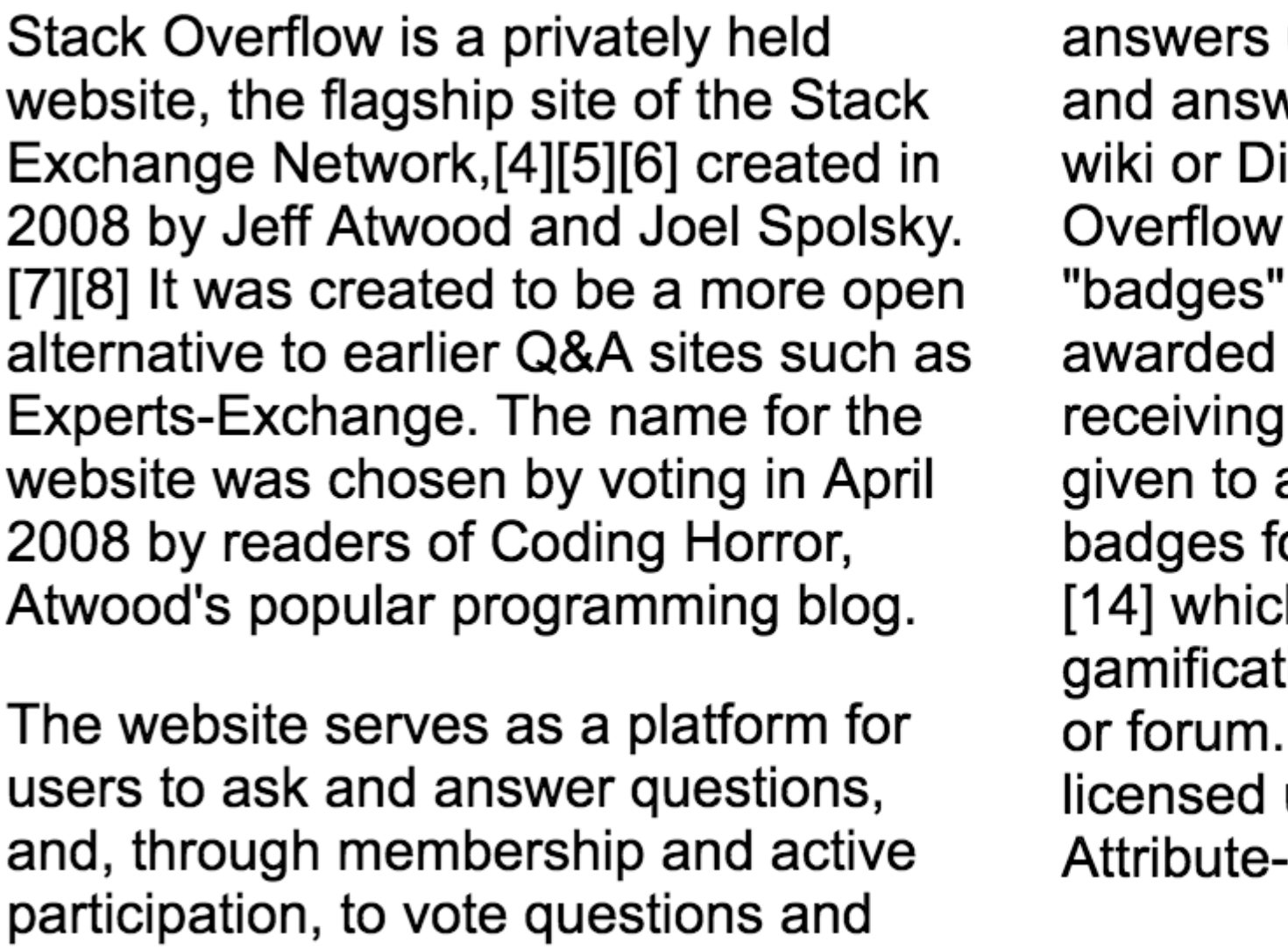
Das CSS-Layout mit mehreren Spalten erleichtert das Erstellen mehrerer Textspalten.

Code

```
<div id="multi-columns">Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do
eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis
nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure
dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur.
Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim
id est laborum</div>
```

```
.multi-columns {
  -moz-column-count: 2;
  -webkit-column-count: 2;
  column-count: 2;
}
```

Ergebnis



Spaltenbreite

Mit `column-width` Eigenschaft column width wird die minimale Spaltenbreite festgelegt. Wenn die `column-count` nicht definiert ist, erstellt der Browser so viele Spalten, wie in die verfügbare Breite passen.

Code:

```
<div id="multi-columns">
    Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum
</div>
```

```
.multi-columns {
  -moz-column-width: 100px;
  -webkit-column-width: 100px;
  column-width: 100px;
}
```

Ergebnis

Stack Overflow is a privately held website, the flagship site of the Stack Exchange Network,[4][5] [6] created in 2008 by Jeff Atwood and Joel Spolsky. [7][8] It was created to be a more open alternative to earlier Q&A sites such as Experts-	Exchange. The name for the website was chosen by voting in April 2008 by readers of Coding Horror, Atwood's popular programming blog. The website serves as a platform for users to ask and answer	questions, and, through membership and active participation, to vote questions and answers up or down and edit questions and answers in a fashion similar to a wiki or Digg.[13] Users of Stack Overflow can earn reputation	points and "badges"; for example, a person is awarded 10 reputation points for receiving an "up" vote on an answer given to a question, and can receive badges for their valued contributions, [14] which represents a kind of	gamification of the traditional Q&A site or forum. All user-generated content is licensed under a Creative Commons Attribute-ShareAlike license.

Säulen online lesen: https://riptutorial.com/de/css/topic/3042/saulen

Kapitel 44: Schwimmt

Syntax

- klar: keine | links | richtig | beide | Inline-Start | Inline-Ende;
- Schwimmer: links | richtig | Keine | Inline-Start | Inline-Ende;

Bemerkungen

> Da float die Verwendung des Blocklayouts impliziert, ändert es in einigen Fällen den berechneten Wert der Anzeigewerte. [1]

[1]: https://developer.mozilla.org/en-US/docs/Web/CSS/float MDN

Examples

Float ein Bild im Text

Die einfachste Verwendung eines Floats besteht darin, dass Text um ein Bild gelegt wird. Der folgende Code erzeugt zwei Absätze und ein Bild, wobei der zweite Absatz um das Bild fließt. Beachten Sie, dass es immer *nach* dem floated-Element, das um das floated-Element fließt, Inhalt ist.

HTML:

```
<p>Lorem ipsum dolor sit amet, consectetur adipiscing elit. Integer nec odio. Praesent libero. Sed cursus ante dapibus diam. Sed nisi. Nulla quis sem at nibh elementum imperdiet. Duis sagittis ipsum. Praesent mauris. Fusce nec tellus sed augue semper porta. Mauris massa. Vestibulum lacinia arcu eget nulla. </p>

<img src="http://lorempixel.com/200/100/" />

<p>Class aptent taciti sociosqu ad litora torquent per conubia nostra, per inceptos himenaeos. Curabitur sodales ligula in libero. Sed dignissim lacinia nunc. Curabitur tortor. Pellentesque nibh. Aenean quam. In scelerisque sem at dolor. Maecenas mattis. Sed convallis tristique sem. Proin ut ligula vel nunc egestas porttitor. Morbi lectus risus, iaculis vel, suscipit quis, luctus non, massa. Fusce ac turpis quis ligula lacinia aliquet. </p>
```

CSS:

```
img {
  float:left;
  margin-right:1rem;
}
```

Dies wird die Ausgabe sein

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Integer nec odio. Praesent libero. Sed cursus ante dapibus diam. Sed nisi. Nulla quis sem at nibh elementum imperdiet. Duis sagittis ipsum. Praesent mauris. Fusce nec tellus sed augue semper porta. Mauris massa. Vestibulum lacinia arcu eget nulla.

Class aptent taciti sociosqu ad litora torquent per conubia nostra, per inceptos himenaeos. Curabitur sodales ligula in libero. Sed dignissim lacinia nunc. Curabitur tortor. Pellentesque nibh. Aenean quam. In scelerisque sem at dolor. Maecenas mattis. Sed convallis tristique sem. Proin ut ligula vel nunc egestas porttitor. Morbi lectus risus, iaculis vel, suscipit quis, luctus non, massa. Fusce ac turpis quis ligula lacinia aliquet.

Codepen Link

Einfaches Layout mit zwei Spalten mit fester Breite

Ein einfaches zweispaltiges Layout besteht aus zwei Floating-Elementen mit fester Breite. Beachten Sie, dass die Seitenleiste und der Inhaltsbereich in diesem Beispiel nicht dieselbe Höhe haben. Dies ist einer der schwierigsten Teile mit mehrspaltigen Layouts, die Floats verwenden, und erfordert Umgehungen, damit mehrere Spalten dieselbe Höhe haben.

HTML:

```
<div class="wrapper">

<div class="sidebar">
  <h2>Sidebar</h2>

  <p>Lorem ipsum dolor sit amet, consectetur adipiscing elit. Integer nec odio.</p>
</div>

<div class="content">
  <h1>Content</h1>

  <p>Class aptent taciti sociosqu ad litora torquent per conubia nostra, per inceptos
himenaeos. Curabitur sodales ligula in libero. Sed dignissim lacinia nunc. Curabitur tortor.
Pellentesque nibh. Aenean quam. In scelerisque sem at dolor. Maecenas mattis. Sed convallis
tristique sem. Proin ut ligula vel nunc egestas porttitor. Morbi lectus risus, iaculis vel,
suscipit quis, luctus non, massa. Fusce ac turpis quis ligula lacinia aliquet. </p>
</div>

</div>
```

CSS:

```
.wrapper {
  width:600px;
  padding:20px;
  background-color:pink;

  /* Floated elements don't use any height. Adding "overflow:hidden;" forces the
     parent element to expand to contain its floated children. */
  overflow:hidden;
}

.sidebar {
  width:150px;
  float:left;
  background-color:blue;
}

.content {
  width:450px;
  float:right;
  background-color:yellow;
}
```

Einfaches Layout mit drei Spalten mit fester Breite

HTML:

```
<div class="wrapper">
  <div class="left-sidebar">
    <h1>Left Sidebar</h1>
    <p>Lorem ipsum dolor sit amet, consectetur adipiscing elit. </p>
  </div>
  <div class="content">
    <h1>Content</h1>
    <p>Class aptent taciti sociosqu ad litora torquent per conubia nostra, per inceptos
himenaeos. Curabitur sodales ligula in libero. Sed dignissim lacinia nunc. Curabitur tortor.
Pellentesque nibh. Aenean quam. In scelerisque sem at dolor. Maecenas mattis. Sed convallis
```

```
tristique sem. Proin ut ligula vel nunc egestas porttitor. Morbi lectus risus, iaculis vel,
suscipit quis, luctus non, massa. </p>
  </div>
  <div class="right-sidebar">
    <h1>Right Sidebar</h1>
    <p>Fusce ac turpis quis ligula lacinia aliquet.</p>
  </div>
</div>
```

CSS:

```
.wrapper {
  width:600px;
  background-color:pink;
  padding:20px;

  /* Floated elements don't use any height. Adding "overflow:hidden;" forces the
     parent element to expand to contain its floated children. */
  overflow:hidden;
}

.left-sidebar {
  width:150px;
  background-color:blue;
  float:left;
}

.content {
  width:300px;
  background-color:yellow;
  float:left;
}

.right-sidebar {
  width:150px;
  background-color:green;
  float:right;
}
```

Zwei Spalten faul / gierig Layout

Dieses Layout verwendet eine Floated-Spalte, um ein zweispaltiges Layout ohne definierte Breiten zu erstellen. In diesem Beispiel ist die linke Seitenleiste "faul", da sie nur so viel Speicherplatz beansprucht, wie sie benötigt. Eine andere Möglichkeit, dies zu sagen, ist, dass die linke Seitenleiste "eingeschweißt" ist. Die rechte Inhaltsspalte ist "gierig", da sie den gesamten verbleibenden Speicherplatz belegt.

HTML:

```
<div class="sidebar">
<h1>Sidebar</h1>
<img src="http://lorempixel.com/150/200/" />
</div>

<div class="content">
<h1>Content</h1>
```

```
<p>Lorem ipsum dolor sit amet, consectetur adipiscing elit. Integer nec odio. Praesent libero. Sed cursus ante dapibus diam. Sed nisi. Nulla quis sem at nibh elementum imperdiet. Duis sagittis ipsum. Praesent mauris. Fusce nec tellus sed augue semper porta. Mauris massa. Vestibulum lacinia arcu eget nulla. </p>
<p>Class aptent taciti sociosqu ad litora torquent per conubia nostra, per inceptos himenaeos. Curabitur sodales ligula in libero. Sed dignissim lacinia nunc. Curabitur tortor. Pellentesque nibh. Aenean quam. In scelerisque sem at dolor. Maecenas mattis. Sed convallis tristique sem. Proin ut ligula vel nunc egestas porttitor. Morbi lectus risus, iaculis vel, suscipit quis, luctus non, massa. Fusce ac turpis quis ligula lacinia aliquet. Mauris ipsum. Nulla metus metus, ullamcorper vel, tincidunt sed, euismod in, nibh. </p>
</div>
```

CSS:

```
.sidebar {
  /* `display:table;` shrink-wraps the column */
  display:table;
  float:left;
  background-color:blue;
}

.content {
  /* `overflow:hidden;` prevents `.content` from flowing under `.sidebar` */
  overflow:hidden;
  background-color:yellow;
}
```

Geige

klares Eigentum

Die klare Eigenschaft steht in direktem Zusammenhang mit Floats. Eigenschaftswerte:

- none - Standard Erlaubt schwebende Elemente auf beiden Seiten
- left - Auf der linken Seite sind keine schwebenden Elemente erlaubt
- right - Auf der rechten Seite sind keine schwebenden Elemente erlaubt
- beide - Keine schwebenden Elemente auf der linken oder rechten Seite zulässig
- initial - Setzt diese Eigenschaft auf ihren Standardwert. Lesen Sie über den Anfang
- erben - Übernimmt diese Eigenschaft von ihrem übergeordneten Element. Lesen Sie mehr über das Erben

```
<html>
<head>
<style>
img {
    float: left;
}

p.clear {
    clear: both;
}
</style>
</head>
<body>
```

```
<img src="https://static.pexels.com/photos/69372/pexels-photo-69372-medium.jpeg" width="100">
<p>Lorem ipsoum Lorem ipsoum Lorem ipsoum Lorem ipsoum Lorem ipsoum Lorem ipsoum Lorem ipsoum Lorem ipsoum
Lorem ipsoum Lorem ipsoum Lorem ipsoum Lorem ipsoum Lorem ipsoum Lorem ipsoum Lorem ipsoum </p>
<p class="clear">Lorem ipsoum Lorem ipsoum Lorem ipsoum Lorem ipsoum Lorem ipsoum Lorem ipsoum Lorem ipsoum
Lorem ipsoum Lorem ipsoum Lorem ipsoum Lorem ipsoum Lorem ipsoum Lorem ipsoum Lorem ipsoum </p>

</body>
</html>
```

Clearfix

> Der Clearfix-Hack ist eine beliebte Methode, um Floats einzudämmen (N. Gallagher aka @necolas)

Nicht mit der `clear` Eigenschaft zu verwechseln, ist clearfix ein *Konzept* (das sich auch auf Floats bezieht, also die mögliche Verwirrung). Um *Floats* zu *enthalten* , müssen Sie dem Container (**dem übergeordneten** Container) **die** Klasse `.cf` oder `.clearfix` hinzufügen und diese Klasse mit einigen unten beschriebenen Regeln `.clearfix` .

3 Versionen mit leicht unterschiedlichen Effekten (Quellen: Ein neuer Micro-Clearfix-Hack von N. Gallagher und Clearfix, die von TJ Koblentz neu geladen wurden):

Clearfix (wobei der obere Rand der eingeschlossenen Schwimmer weiterhin zusammenbricht)

```
.cf:after {
    content: "";
    display: table;
}

.cf:after {
    clear: both;
}
```

Clearfix verhindert auch das Zusammenfallen der oberen Randbereiche

```
/**
 * For modern browsers
 * 1. The space content is one way to avoid an Opera bug when the
 *    contenteditable attribute is included anywhere else in the document.
 *    Otherwise it causes space to appear at the top and bottom of elements
 *    that are clearfixed.
 * 2. The use of `table` rather than `block` is only necessary if using
 *    `:before` to contain the top-margins of child elements.
```

```
 */
.cf:before,
.cf:after {
    content: " "; /* 1 */
    display: table; /* 2 */
}

.cf:after {
    clear: both;
}
```

Clearfix mit Unterstützung der veralteten Browser IE6 und IE7

```
.cf:before,
.cf:after {
    content: " ";
    display: table;
}

.cf:after {
    clear: both;
}

/**
 * For IE 6/7 only
 * Include this rule to trigger hasLayout and contain floats.
 */
.cf {
    *zoom: 1;
}
```

Codepen mit Clearfix-Effekt

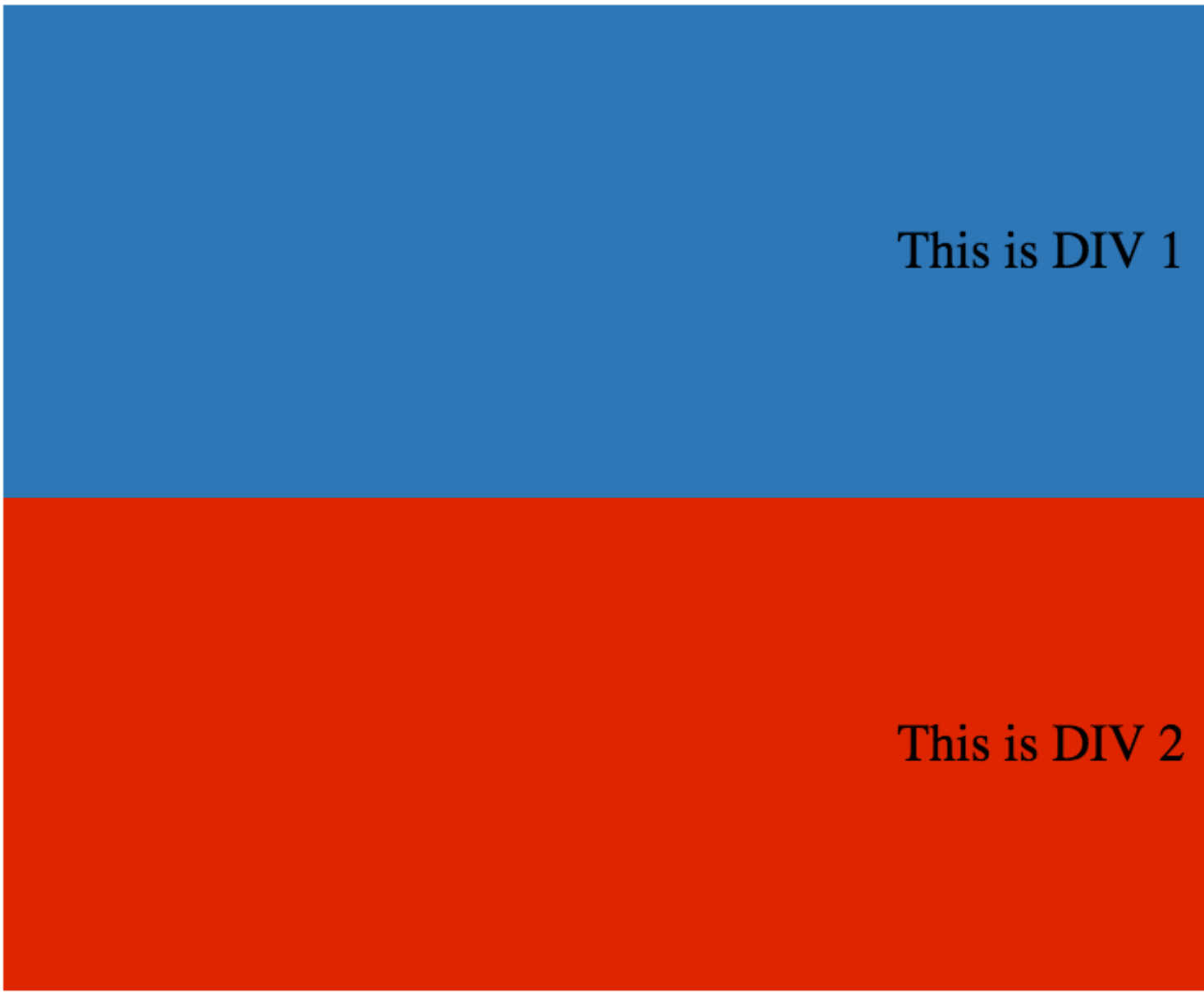
Andere Ressource: Alles, was Sie über clearfix wissen, ist falsch (clearfix und BFC - Block Formatting Context, während sich hasLayout auf veraltete Browser bezieht, IE6 möglicherweise 7).

Inline-DIV mit Float

Das `div` ist ein Block-Level-Element, dh es nimmt die gesamte Seitenbreite ein und die Geschwister liegen unabhängig von ihrer Breite untereinander.

```
<div>
    <p>This is DIV 1</p>
</div>
<div>
    <p>This is DIV 2</p>
</div>
```

Die Ausgabe des folgenden Codes wird sein



Wir können sie inline machen, indem wir dem `div` eine `float` css -Eigenschaft hinzufügen.

HTML:

```
<div class="outer-div">
    <div class="inner-div1">
        <p>This is DIV 1</p>
    </div>
    <div class="inner-div2">
        <p>This is DIV 2</p>
    </div>
</div>
```

CSS

```
.inner-div1 {
    width: 50%;
```

```
    margin-right:0px;
    float:left;
    background : #337ab7;
    padding:50px 0px;
}

.inner-div2 {
    width: 50%;
    margin-right:0px;
    float:left;
    background : #dd2c00;
    padding:50px 0px;
}

p {
    text-align:center;
}
```

Codepen Link

Verwendung der Überlaufeigenschaft zum Löschen von Schwimmern

Wenn Sie den `overflow` auf " `hidden` , " `auto` oder " `scroll` zu einem Element setzen, werden alle Floats innerhalb dieses Elements gelöscht.

Hinweis: `overflow:scroll` Scrollen zeigt immer die Scrollbox an

Schwimmt online lesen: https://riptutorial.com/de/css/topic/405/schwimmt

Kapitel 45: Selektoren

Einführung

CSS-Selektoren identifizieren bestimmte HTML-Elemente als Ziele für CSS-Stile. In diesem Thema wird beschrieben, wie CSS-Selektoren auf HTML-Elemente abzielen. Selektoren verwenden eine breite Palette von über 50 Auswahlmethoden, die von der CSS-Sprache angeboten werden, einschließlich Elemente, Klassen, IDs, Pseudo-Elemente und Pseudo-Klassen sowie Muster.

Syntax

- # *id*
- . *Klassenname*
- : *Pseudoklassenname*
- :: *Pseudoelementname*
- [*attr*] / * hat das Attribut *attr* . * /
- [*attr* = " *value* "] / * hat das Attribut *attr* , und sein Wert ist genau " *value* ". * /
- [*attr* ~ = " *value* "] / * hat das Attribut *attr* , und sein Wert enthält, wenn er auf **Leerzeichen aufgeteilt ist** , " *value* ". * /
- [*attr* | = " *value* "] / * hat das *attr*- Attribut und sein Wert ist genau " *value* " **oder** sein Wert beginnt mit " *value* **-** ". * /
- [*attr* ^ = " *value* "] / * hat das Attribut *attr* , und sein Wert beginnt mit " *value* ". * /
- [*attr* $ = " *value* "] / * hat das Attribut *attr* , und sein Wert endet mit " *value* ". * /
- [*attr* * = " *value* "] / * hat das Attribut *attr* , und sein Wert enthält " *value* ". * /
- *Elementname*
- *

Bemerkungen

- Manchmal werden Sie zwei Doppelpunkte (siehe ::) anstelle von nur einem (:). Dies ist eine Möglichkeit, Pseudoklassen von Pseudoelementen zu trennen.
- Alter Browser wie Internet Explorer 8, unterstützt **nur** einen einzigen Doppelpunkt (:) zum Definieren Pseudo-Elemente.
- Im Gegensatz zu Pseudoklassen kann nur ein Pseudoelement pro Selektor verwendet werden. Wenn es vorhanden ist, muss es nach der Sequenz einfacher Selektoren erscheinen, die die Subjekte des Selektors darstellen (eine zukünftige Version der W3C-Spezifikation kann mehrere Pseudoelemente pro Selektor zulassen).

Examples

Attribut-Selektoren

Überblick

Attributselektoren können mit verschiedenen Arten von Operatoren verwendet werden, die die Auswahlkriterien entsprechend ändern. Sie wählen ein Element anhand des Vorhandenseins eines bestimmten Attributs oder Attributwerts aus.

Auswahl [1]	Übereinstimmendes Element	Wählt Elemente aus ...	CSS-Version
`[attr]`	`<div attr>`	Mit Attribut `attr`	2
`[attr='val']`	`<div attr="val">`	Dabei hat das Attribut `attr` den Wert `val`	2
`[attr~='val']`	`<div attr="val val2 val3">`	Wo `val` erscheint durch Leerzeichen getrennte Liste von `attr`	2
`[attr^='val']`	`<div attr="val1 val2">`	Wo der Wert von `attr` mit `val` *beginnt*	3
`[attr$='val']`	`<div attr="sth aval">`	Wo der Wert des `attr` mit `val` *endet*	3
`[attr*='val']`	`<div attr="somevalhere">`	Wo `attr` `val` überall enthält	3
`[attr|='val']`	`<div attr="val-sth etc">`	Wo der Wert von `attr` genau `val` , oder beginnt mit `val` und sofort gefolgt von `-` (U + 002D)	2
`[attr='val' i]`	`<div attr="val">`	Wo `attr` den Wert `val` , ignorieren die `val` .	4 [2]

Anmerkungen:

1. Der Attributwert kann entweder in einfache oder doppelte Anführungszeichen gesetzt werden. Es können auch keine Anführungszeichen funktionieren, dies gilt jedoch gemäß dem CSS-Standard und wird nicht empfohlen.

2. Es gibt keine integrierte CSS4-Spezifikation, da diese in separate Module aufgeteilt ist. Es gibt jedoch "Level 4" -Module. Siehe Browserunterstützung .

Einzelheiten

`[attribute]`

Wählt Elemente mit dem angegebenen Attribut aus.

```
div[data-color] {
  color: red;
}
```

```
<div data-color="red">This will be red</div>
<div data-color="green">This will be red</div>
<div data-background="red">This will NOT be red</div>
```

Live Demo auf JSBin

[attribute="value"]

Wählt Elemente mit dem angegebenen Attribut und Wert aus.

```
div[data-color="red"] {
  color: red;
}
```

```
<div data-color="red">This will be red</div>
<div data-color="green">This will NOT be red</div>
<div data-color="blue">This will NOT be red</div>
```

Live Demo auf JSBin

[attribute*="value"]

Wählt Elemente mit dem angegebenen Attribut und Wert aus, wobei das angegebene Attribut den angegebenen Wert an einer beliebigen Stelle (als Unterzeichenfolge) enthält.

```
[class*="foo"] {
  color: red;
}
```

```
<div class="foo-123">This will be red</div>
<div class="foo123">This will be red</div>
<div class="bar123foo">This will be red</div>
<div class="barfooo123">This will be red</div>
<div class="barfo0">This will NOT be red</div>
```

Live Demo auf JSBin

[attribute~="value"]

Wählt Elemente mit dem angegebenen Attribut und Wert aus, wobei der angegebene Wert in einer durch Leerzeichen getrennten Liste angezeigt wird.

```
[class~="color-red"] {
  color: red;
}
```

```
<div class="color-red foo-bar the-div">This will be red</div>
<div class="color-blue foo-bar the-div">This will NOT be red</div>
```

Live Demo auf JSBin

[attribute^="value"]

Wählt Elemente mit dem angegebenen Attribut und Wert aus, bei denen das angegebene Attribut mit dem Wert beginnt.

```
[class^="foo-"] {
  color: red;
}
```

```
<div class="foo-123">This will be red</div>
<div class="foo-234">This will be red</div>
<div class="bar-123">This will NOT be red</div>
```

Live Demo auf JSBin

[attribute$="value"]

Wählt Elemente mit dem angegebenen Attribut und Wert aus, wobei das angegebene Attribut mit dem angegebenen Wert endet.

```
[class$="file"] {
  color: red;
}
```

```
<div class="foobar-file">This will be red</div>
<div class="foobar-file">This will be red</div>
<div class="foobar-input">This will NOT be red</div>
```

Live Demo auf JSBin

[attribute|="value"]

Wählt Elemente mit einem bestimmten Attribut und einem Wert aus, wobei der Wert des Attributs genau dem angegebenen Wert oder genau dem angegebenen Wert gefolgt von - (U + 002D) entspricht.

```
[lang|="EN"] {
  color: red;
}
```

```
<div lang="EN-us">This will be red</div>
<div lang="EN-gb">This will be red</div>
<div lang="PT-pt">This will NOT be red</div>
```

Live Demo auf JSBin

`[attribute="value" i]`

Wählt Elemente mit einem bestimmten Attribut und Wert , bei dem der Wert des Attributs kann dargestellt werden als `Value` , `VALUE` , `vAlUe` oder andere Groß- und Kleinschreibung Möglichkeit.

```
[lang="EN" i] {
  color: red;
}
```

```
<div lang="EN">This will be red</div>
<div lang="en">This will be red</div>
<div lang="PT">This will NOT be red</div>
```

Live Demo auf JSBin

Spezifität der Attributselektoren

`0-1-0`

Entspricht dem Klassenselektor und der Pseudoklasse.

```
*[type=checkbox] // 0-1-0
```

Beachten Sie, dass dies bedeutet, dass ein Attributselektor verwendet werden kann, um ein Element anhand seiner ID auf einer niedrigeren `#my-ID` auszuwählen, als wenn es mit einem ID-Selektor ausgewählt wurde : `[id="my-ID"]` zielt auf dasselbe Element wie `#my-ID` aber mit geringerer Spezifität.

Weitere Informationen finden Sie im Abschnitt "Syntax" .

Kombinatoren

Überblick

Wähler	Beschreibung
`div span`	Nachkommen-Selektor (alle `<span>` s, die Nachkommen eines `<div>`)
`div > span`	Child Selector (alle `<span>` s, die ein direktes Kind eines `<div>`)
`a ~ span`	Allgemeine Geschwisterauswahl (alle `<span>` s, die nach einem `<a>` Geschwister sind)
`a + span`	Benachbarte Geschwisterauswahl (alle `<span>` s, die unmittelbar nach einem `<a>`)

Hinweis: Geschwister-Selektoren zielen auf Elemente, die im Quelldokument nach ihnen kommen. CSS kann (aufgrund seiner Kaskaden) nicht auf *frühere* oder *übergeordnete* Elemente zielen. Unter Verwendung der Flex- `order` kann jedoch ein vorheriger Geschwister-Selektor auf visuellen Medien simuliert werden .

Nachkomme-Combinator: `selector selector`

Ein Nachkomme-Kombinator, dargestellt durch mindestens ein Leerzeichen ( ) wählt Elemente aus, die von dem definierten Element abhängen. Dieser Kombinator wählt **alle** Nachkommen des Elements aus (von untergeordneten Elementen abwärts).

```
div p {
  color:red;
}
```

```
<div>
  <p>My text is red</p>
  <section>
    <p>My text is red</p>
  </section>
</div>

<p>My text is not red</p>
```

Live Demo auf JSBin

Im obigen Beispiel werden die ersten beiden `<p>` -Elemente ausgewählt, da sie beide Nachkommen von `<div>` .

Child Combinator: `selector > selector`

Das Kind (`>`) combinator wird verwendet , um Elemente auszuwählen , die **Kinder** oder **direkte Nachkommen** sind, des angegebenen Elements.

```
div > p {
  color:red;
}
```

```
<div>
  <p>My text is red</p>
  <section>
    <p>My text is not red</p>
  </section>
</div>
```

Live Demo auf JSBin

Das obige CSS wählt nur das erste `<p>` -Element aus, da es der einzige Absatz ist, der direkt von

einem `<div>` .

Das zweite `<p>` -Element ist nicht ausgewählt, da es kein direktes `<p>` Element von `<div>` .

Angrenzender Geschwister-Kombinator: `selector + selector`

Der benachbarte Geschwister (`+`) - Kombinator wählt ein Geschwisterelement aus, das unmittelbar auf ein bestimmtes Element folgt.

```
p + p {
  color:red;
}
```

```
<p>My text is not red</p>
<p>My text is red</p>
<p>My text is red</p>
<hr>
<p>My text is not red</p>
```

Live Demo auf JSBin

Im obigen Beispiel werden nur die `<p>` -Elemente ausgewählt, denen ein anderes `<p>` -Element *direkt vorangeht* .

General Sibling Combinator: `selector ~ selector`

Der allgemeine Geschwister (`~`) Kombinator wählt *alle* Geschwister aus, die dem angegebenen Element folgen.

```
p ~ p {
  color:red;
}
```

```
<p>My text is not red</p>
<p>My text is red</p>
<hr>
<h1>And now a title</h1>
<p>My text is red</p>
```

Live Demo auf JSBin

Im obigen Beispiel werden alle `<p>` -Elemente ausgewählt, *denen ein* anderes `<p>` -Element *vorangeht* , unabhängig davon, ob sie unmittelbar nebeneinander liegen oder nicht.

Klassennamen-Selektoren

Der Klassennamenselektor wählt alle Elemente mit dem Zielklassennamen aus. Zum Beispiel würde der Klassenname `.warning` das folgende `<div>` -Element auswählen:

```
<div class="warning">
    <p>This would be some warning copy.</p>
</div>
```

Sie können Klassennamen auch spezifischer mit Zielelementen kombinieren. Bauen wir auf das obige Beispiel auf, um eine kompliziertere Klassenauswahl zu zeigen.

CSS

```
.important {
    color: orange;
}
.warning {
    color: blue;
}
.warning.important {
    color: red;
}
```

HTML

```
<div class="warning">
    <p>This would be some warning copy.</p>
</div>

<div class="important warning">
    <p class="important">This is some really important warning copy.</p>
</div>
```

In diesem Beispiel haben alle Elemente mit der Klasse `.warning` eine blaue `.important` , Elemente mit der `.important` Klasse mit einer orangefarbenen `.important` , und alle Elemente, die *sowohl* den `.important` Namen als auch die Klasse `.important .warning` , haben einen roten Text Farbe.

Beachten Sie, dass die Deklaration `.warning.important` innerhalb des CSS keine Leerzeichen zwischen den beiden Klassennamen hatte. Das bedeutet , es wird nur Elemente finden , die beiden Klassennamen enthalten `warning` und `important` in ihrer `class` Attribute. Diese Klassennamen können sich in beliebiger Reihenfolge auf dem Element befinden.

Wenn zwischen den beiden Klassen in der CSS-Deklaration ein Leerzeichen enthalten ist, werden nur Elemente ausgewählt, die übergeordnete Elemente mit einem `.warning` Klassennamen und `.important` Elemente mit `.important` Klassennamen aufweisen.

ID-Selektoren

ID-Selektoren wählen DOM-Elemente mit der Ziel-ID aus. Um ein Element anhand einer bestimmten ID in CSS auszuwählen, wird das Präfix `#` verwendet.

Zum Beispiel das folgende HTML-Element `div` …

```
<div id="exampleID">
    <p>Example</p>
</div>
```

... `#exampleID` in CSS wie `#exampleID` mit `#exampleID` ausgewählt werden:

```
#exampleID {
    width: 20px;
}
```

> **Hinweis** : Die HTML-Spezifikationen lassen nicht mehrere Elemente mit derselben ID zu

Pseudoklassen

Pseudoklassen sind **Schlüsselwörter,** die eine Auswahl anhand von Informationen ermöglichen, die außerhalb des Dokumentbaums liegen oder von anderen Selektoren oder Kombinatoren nicht ausgedrückt werden können. Diese Informationen können in einen bestimmten Zustand (in Verbindung gebracht werden Zustand und dynamische Pseudoklassen), an Orten (strukturellen und Ziel Pseudo-Klassen), zu Negationen der ehemaligen (Negation Pseudo-Klasse) oder Sprachen (lang pseudo-Klasse). Beispiele sind, ob ein Link verfolgt wurde (`:visited`) oder nicht, der Mauszeiger über einem Element (`:hover`) ist, ein Kontrollkästchen ist markiert (`:checked`) usw.

Syntax

```
selector:pseudo-class {
    property: value;
}
```

Liste der Pseudoklassen:

Name	Beschreibung
`:active`	Gilt für jedes Element, das vom Benutzer aktiviert (dh angeklickt) wird.
`:any`	Ermöglicht das Erstellen von Gruppen zusammengehöriger Selektoren, indem Sie Gruppen erstellen, die die enthaltene Elemente werden übereinstimmen. Dies ist eine Alternative zum Wiederholen eines gesamten Selektors.
`:target`	Wählt das aktuell aktive #news-Element aus (auf eine URL geklickt) diesen Ankernamen enthalten)
`:checked`	Gilt für Funk-, Kontrollkästchen- oder Optionselemente, die markiert sind oder umgeschaltet in einen "Ein" -Zustand.

Name	Beschreibung
`:default`	Stellt ein beliebiges Benutzeroberflächenelement dar, das in einer Gruppe von standardmäßig verwendet wird ähnliche Elemente.
`:disabled`	Gilt für jedes UI-Element, das sich in einem deaktivierten Zustand befindet.
`:empty`	Gilt für Elemente, die keine Kinder haben.
`:enabled`	Gilt für jedes UI-Element, das sich im aktivierten Zustand befindet.
`:first`	Wird in Verbindung mit der `@page` Regel verwendet, wird die erste Seite in a ausgewählt gedrucktes Dokument.
`:first-child`	Stellt ein beliebiges Element dar, das das erste untergeordnete Element des übergeordneten Elements ist.
`:first-of-type`	Trifft zu, wenn ein Element das erste des ausgewählten Elementtyps ist in seinem Elternteil. Dies kann das erste Kind sein oder nicht.
`:focus`	Gilt für jedes Element, das den Fokus des Benutzers hat. Dies kann durch die gegeben werden Tastatur, Mausereignisse oder andere Eingabearten des Benutzers.
`:focus-within`	Kann verwendet werden, um einen ganzen Abschnitt hervorzuheben, wenn ein Element darin fokussiert ist. Es stimmt mit jedem Element überein, mit dem die: focus-Pseudoklasse übereinstimmt oder auf das ein Nachkomme ausgerichtet ist.
`:full-screen`	Gilt für alle Elemente, die im Vollbildmodus angezeigt werden. Es wählt den gesamten Stapel aus von Elementen und nicht nur das Element der obersten Ebene.
`:hover`	Gilt für jedes Element, das vom Zeigegerät des Benutzers angezeigt wird, aber nicht aktiviert.
`:indeterminate`	Wendet Radio- oder Checkbox-UI-Elemente an, die weder markiert noch aktiviert sind ungeprüft, aber in einem unbestimmten Zustand. Das kann an einem liegen Attribut des Elements oder DOM-Manipulation.
`:in-range`	Die `:in-range` CSS-Pseudoklasse stimmt überein, wenn ein Element dies hat Sein Wertattribut innerhalb der angegebenen Bereichsbeschränkungen für dieses Element. Dadurch kann die Seite eine Rückmeldung über den aktuell definierten Wert geben

Name	Beschreibung
	Die Verwendung des Elements liegt innerhalb der Bereichsgrenzen.
`:invalid`	Gilt für `<input>` -Elemente, deren Werte gemäß ungültig sind der im Attribut `type=` angegebene `type=` .
`:lang`	Gilt für jedes Element, dessen `<body>` -Element korrekt umschlossen ist bezeichnet `lang=` Attribut. Damit die Pseudoklasse gültig ist, muss sie gültig sein enthalten einen gültigen zwei- oder dreibuchstabigen Sprachcode.
`:last-child`	Stellt ein beliebiges Element dar, das das letzte untergeordnete Element des übergeordneten Elements ist.
`:last-of-type`	Wird angewendet, wenn ein Element das letzte Element des ausgewählten Elementtyps ist sein Elternteil. Dies kann das letzte Kind sein oder nicht.
`:left`	In Verbindung mit der `@page` Regel werden alle Links ausgewählt Seiten in einem gedruckten Dokument.
`:link`	Gilt für Links, die der Benutzer nicht besucht hat.
`:not()`	Gilt für alle Elemente, **die nicht** mit dem übergebenen Wert übereinstimmen (`:not(p)` oder `:not(.class-name)` zum Beispiel muss es einen Wert haben gültig und kann nur einen Selektor enthalten. Sie können jedoch mehrere verketten `:not` Selektoren zusammen.
`:nth-child`	Trifft zu, wenn ein Element das `n` te Element des übergeordneten Elements ist, wobei `n` kann eine ganze Zahl, ein mathematischer Ausdruck (z. B. `n+3`) oder die Schlüsselwörter sein `odd` oder `even` .
`:nth-of-type`	Trifft zu, wenn ein Element das `n` te Element seines übergeordneten Elements ist derselbe Elementtyp, wobei `n` eine ganze Zahl sein kann, eine mathematische Ausdruck (zB `n+3`) oder die Schlüsselwörter `odd` oder `even` .
`:only-child`	Die `:only-child` CSS-Pseudoklasse repräsentiert ein beliebiges Element das ist das einzige Kind seiner Eltern. Das ist das Gleiche wie `:first-child:last-child` oder `:nth-child(1):nth-last-child(1)` , aber mit einer geringeren Spezifität.
`:optional`	Die `:optional` CSS-Pseudoklasse repräsentiert ein beliebiges Element das hat nicht das erforderliche Attribut gesetzt. Dies erlaubt

Name	Beschreibung
	Formulare, um optionale Felder einfach anzuzeigen und entsprechend zu gestalten.
`:out-of-range`	Die `:out-of-range` CSS-Pseudoklasse stimmt überein, wenn ein Element über eine solche verfügt value-Attribut außerhalb der angegebenen Bereichsbeschränkungen für dieses Element. Es gibt der Seite die Möglichkeit, eine Rückmeldung darüber zu geben, dass der aktuell definierte Wert mit der Element liegt außerhalb der Bereichsgrenzen. Ein Wert kann außerhalb eines Bereichs liegen, falls dies der Fall ist entweder kleiner oder größer als die maximalen und minimalen Einstellwerte.
`:placeholder-shown`	**Experimental.** Gilt für jedes Formularelement, das aktuell Platzhaltertext anzeigt.
`:read-only`	Gilt für Elemente, die vom Benutzer nicht bearbeitet werden können.
`:read-write`	Gilt für jedes Element, das von einem Benutzer bearbeitet werden kann, z. B. `<input>` -Elemente.
`:right`	In Verbindung mit der `@page` Regel werden alle richtigen Seiten in a ausgewählt gedrucktes Dokument.
`:root`	stimmt mit dem Stammelement einer Baumstruktur überein, die das Dokument darstellt.
`:scope`	Die CSS-Pseudoklasse entspricht den Elementen, die eine Referenz sind Punkt für Selektoren, gegen die ein Match ausgeführt werden soll.
`:target`	Wählt das aktuell aktive #news-Element aus (auf eine URL geklickt) diesen Ankernamen enthalten)
`:visited`	Gilt für alle Links, die vom Benutzer besucht wurden.

Die `:visited` Pseudoklasse kann in vielen modernen Browsern nicht mehr zum Stilisieren verwendet werden, da dies eine Sicherheitslücke darstellt. Siehe diesen Link als Referenz.

Grundlegende Selektoren

Wähler	Beschreibung
*	Universalauswahl (alle Elemente)

Wähler	Beschreibung
`div`	Tag-Selektor (alle `<div>` -Elemente)
`.blue`	Klassenauswahl (alle Elemente mit Klasse `blue`)
`.blue.red`	Alle Elemente mit der Klasse `blue` **und** `red` (eine Art Zusammensetzungsauswahl)
`#headline`	ID-Selektor (das Element mit "id" -Attribut auf `headline`)
`:pseudo-class`	Alle Elemente mit Pseudoklasse
`::pseudo-element`	Element, das mit dem Pseudoelement übereinstimmt
`:lang(en)`	Entsprechendes Element: lang-Deklaration, zum Beispiel `<span lang="en">`
`div > p`	Kinderauswahl

> **Hinweis:** Der Wert einer ID muss auf einer Webseite eindeutig sein. Es ist eine Verletzung des HTML-Standards , den Wert einer ID mehr als einmal in derselben Dokumentstruktur zu verwenden.

Eine vollständige Liste der Selektoren finden Sie in der Spezifikation von CSS Selectors Level 3 .

So gestalten Sie eine Range-Eingabe

HTML

```
<input type="range"></input>
```

CSS

Bewirken	Pseudo Selector
Daumen	`input[type=range]::-webkit-slider-thumb, input[type=range]::-moz-range-thumb, input[type=range]::-ms-thumb`
Spur	`input[type=range]::-webkit-slider-runnable-track, input[type=range]::-moz-range-track, input[type=range]::-ms-track`
Im Fokus	`input[type=range]:focus`
Unterer Teil der Spur	`input[type=range]::-moz-range-progress, input[type=range]::-ms-fill-lower` (derzeit nicht in WebKit-Browsern möglich - JS erforderlich)

Global boolean mit Checkbox: angehakt und ~ (allgemeiner Geschwister-Kombinator)

Mit dem ~ Selektor können Sie problemlos ein global zugängliches Boolean ohne JavaScript implementieren.

Fügen Sie boolean als Kontrollkästchen hinzu

Fügen Sie ganz am Anfang Ihres Dokuments mit einer eindeutigen `id` und dem `hidden` Attributsatz so viele Booleans hinzu, wie Sie möchten:

```
<input type="checkbox" id="sidebarShown" hidden />
<input type="checkbox" id="darkThemeUsed" hidden />

<!-- here begins actual content, for example: -->
<div id="container">
    <div id="sidebar">
        <!-- Menu, Search, ... -->
    </div>

    <!-- Some more content ... -->
</div>

<div id="footer">
    <!-- ... -->
</div>
```

Ändern Sie den Wert des Booleschen Werts

Sie können den Booleschen Wert umschalten, indem Sie eine `label` mit dem Attribut " `for` hinzufügen:

```
<label for="sidebarShown">Show/Hide the sidebar!</label>
```

Zugriff auf boolesche Werte mit CSS

Die normale `.color-red` (wie `.color-red`) gibt die Standardeigenschaften an. Sie können durch folgende `true` / `false` Selektoren überschrieben werden:

```
/* true: */
<checkbox>:checked ~ [sibling of checkbox & parent of target] <target>

/* false: */
<checkbox>:not(:checked) ~ [sibling of checkbox & parent of target] <target>
```

Beachten Sie, dass `<checkbox>` , `[sibling ...]` und `<target>` durch die richtigen Selektoren ersetzt werden sollten. `[sibling ...]` kann ein bestimmter Selektor sein (oft, wenn Sie faul sind) `*` oder nichts, wenn das Ziel bereits ein Bruder des Kontrollkästchens ist.

Beispiele für die obige HTML-Struktur wären:

```
#sidebarShown:checked ~ #container #sidebar {
    margin-left: 300px;
}

#darkThemeUsed:checked ~ #container,
#darkThemeUsed:checked ~ #footer {
    background: #333;
}
```

In Aktion

Siehe diese Geige für eine Implementierung dieser globalen Booleans.

CSS3: Beispiel für den Bereichsauswahlbereich

```
<style>
input:in-range {
    border: 1px solid blue;
}
</style>



<input type="number" min="10" max="20" value="15">
<p>The border for this value will be blue</p>
```

Die CSS-Pseudoklasse `:in-range` stimmt überein, wenn ein Wertattribut eines Elements innerhalb der angegebenen Bereichsbeschränkungen für dieses Element liegt. Dadurch kann die Seite eine Rückmeldung geben, dass der aktuell mithilfe des Elements definierte Wert innerhalb der Bereichsgrenzen liegt. [1]

Kind Pseudo-Klasse

> Msgstr "Die: nth - child (an + b) - CSS - Pseudoklasse stimmt mit einem Element überein, das in der Dokumentstruktur ein + b - 1 - gleichgeordnetes Element enthält, und zwar für einen bestimmten positiven **oder Nullwert** für n" - MDN: nth - child

Pseudo-Selektor	1	2	3	4	5	6	7	8	9	10
`:first-child`	✔									
`:nth-child(3)`			✔							
`:nth-child(n+3)`			✔	✔	✔	✔	✔	✔	✔	✔
`:nth-child(3n)`			✔			✔			✔	
`:nth-child(3n+1)`	✔			✔			✔			✔

Pseudo-Selektor	1	2	3	4	5	6	7	8	9	10
`:nth-child(-n+3)`	✔	✔	✔							
`:nth-child(odd)`	✔		✔		✔		✔		✔	
`:nth-child(even)`		✔		✔		✔		✔		✔
`:last-child`										✔
`:nth-last-child(3)`								✔		

Wählen Sie das Element anhand seiner ID ohne die hohe Spezifität des ID-Selektors aus

Mit diesem Trick können Sie ein Element auswählen, das die ID als Wert für einen Attributselektor verwendet, um die hohe Spezifität des ID-Selektors zu vermeiden.

HTML:

```
<div id="element">...</div>
```

CSS

```
#element { ... } /* High specificity will override many selectors */

[id="element"] { ... } /* Low specificity, can be overridden easily */
```

A. Das: keine Pseudoklassenbeispiel & B.: focus-within CSS-Pseudoklasse

A. Die Syntax ist oben dargestellt.

Der folgende Selektor stimmt mit allen `<input>` -Elementen in einem HTML-Dokument überein, die nicht deaktiviert sind und keine Klasse haben `.example` :

HTML:

```
<form>
    Phone: <input type="tel" class="example">
    E-mail: <input type="email" disabled="disabled">
    Password: <input type="password">
</form>
```

CSS:

```
input:not([disabled]):not(.example){
   background-color: #ccc;
}
```

Die `:not()` Pseudo-Klasse unterstützt in Selectors Level 4 auch durch Kommas getrennte Selektoren:

CSS:

```
input:not([disabled], .example){
   background-color: #ccc;
}
```

Live Demo auf JSBin

Siehe Hintergrundsyntax hier .

B. Die: focus-within CSS-Pseudoklasse

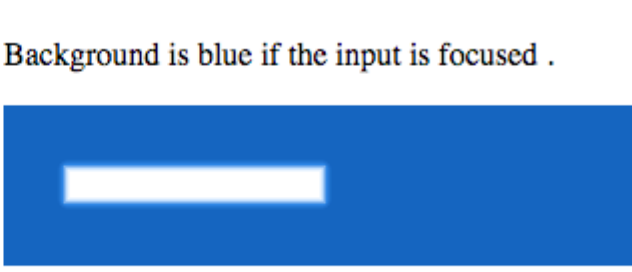
HTML:

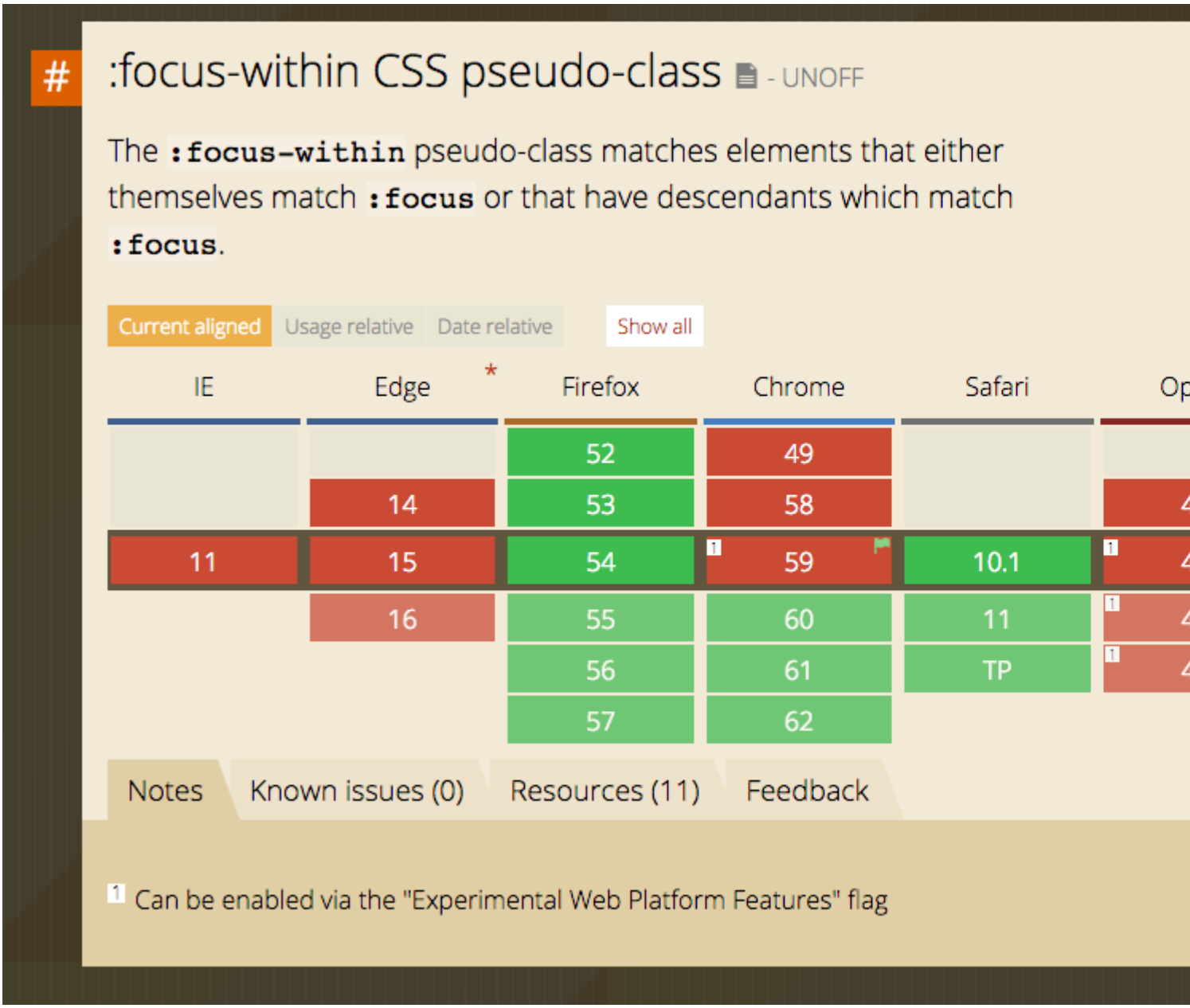
```
  <h3>Background is blue if the input is focused .</p>
  <div>
    <input type="text">
  </div>
```

CSS:

```
div {
  height: 80px;
}
input{
  margin:30px;
}
div:focus-within {
  background-color: #1565C0;
}
```

```
div {
  height: 80px;
}
input{
  margin:30px;
}
div:focus-within {
  background-color: #1565C0;
}
```





Das Beispiel: Nur-Kind-Pseudo-Klassenauswahl

Die `:only-child CSS` -Pseudoklasse stellt jedes Element dar, das das einzige untergeordnete Element des übergeordneten Elements ist.

HTML:

```
<div>
  <p>This paragraph is the only child of the div, it will have the color blue</p>
</div>

<div>
  <p>This paragraph is one of the two children of the div</p>
  <p>This paragraph is one of the two children of its parent</p>
</div>
```

CSS:

```
p:only-child {
  color: blue;
}
```

Das obige Beispiel wählt das Element `<p>` , das das eindeutige `<p>` Element darstellt, in diesem Fall ein `<div>` .

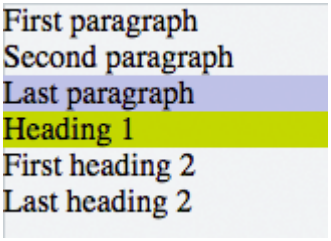
Live Demo auf JSBin

Die: letzte Auswahl

Der `:last-of-type` wählt das Element, das das letzte untergeordnete Element eines bestimmten Typs seines übergeordneten Elements ist. In dem folgenden Beispiel wählt die CSS den letzten Absatz und die letzte Überschrift `h1` .

```
p:last-of-type {
  background: #C5CAE9;
}
h1:last-of-type {
  background: #CDDC39;
}
```

```
<div class="container">
    <p>First paragraph</p>
    <p>Second paragraph</p>
    <p>Last paragraph</p>
    <h1>Heading 1</h1>
    <h2>First heading 2</h2>
    <h2>Last heading 2</h2>
</div>
```



jsFiddle

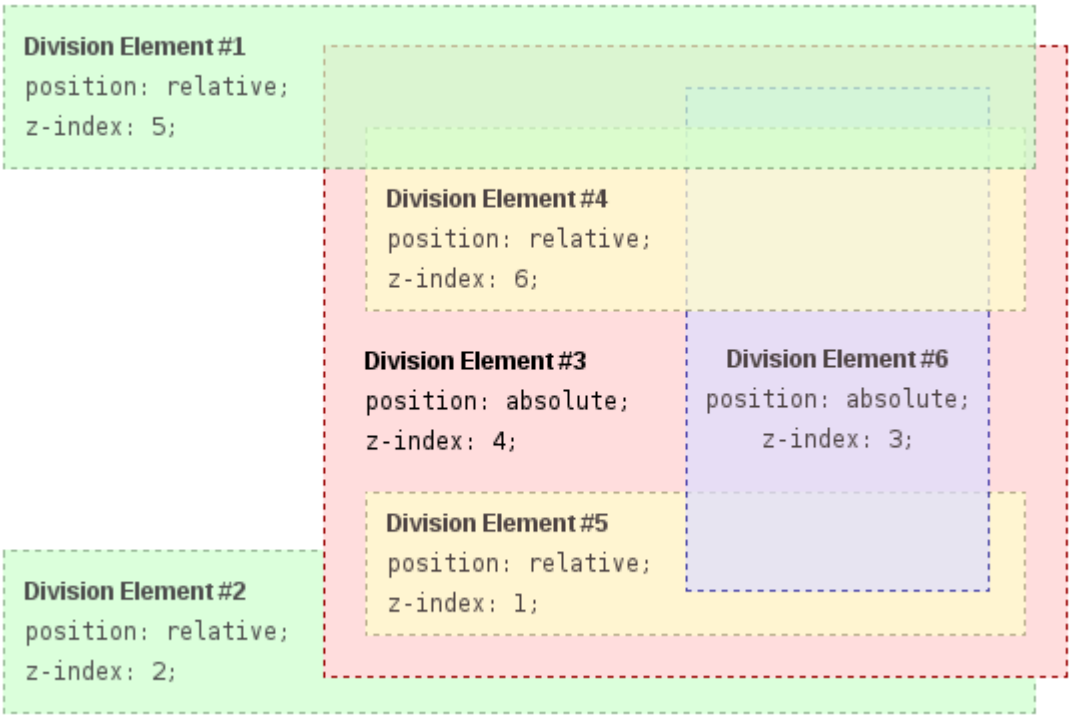
Selektoren online lesen: https://riptutorial.com/de/css/topic/611/selektoren

Kapitel 46: Stapelkontext

Examples

Stapelkontext

In diesem Beispiel erstellt jedes positionierte Element aufgrund seiner Positionierungs- und Z-Index-Werte einen eigenen Stapelkontext. Die Hierarchie der Stapelkontexte ist wie folgt organisiert:



- Wurzel
  - DIV # 1
  - DIV # 2
  - DIV # 3
  - DIV # 4
  - DIV # 5
  - DIV # 6

Es ist wichtig zu beachten, dass DIV # 4, DIV # 5 und DIV # 6 Kinder von DIV # 3 sind. Das Stapeln dieser Elemente wird daher innerhalb von DIV # 3 vollständig aufgelöst. Sobald das Stapeln und Rendern in DIV # 3 abgeschlossen ist, wird das gesamte DIV # 3-Element im Root-Element in Bezug auf das DIV seines Geschwisters zum Stapeln übergeben.

HTML:

```
<div id="div1">
  <h1>Division Element #1</h1>
```

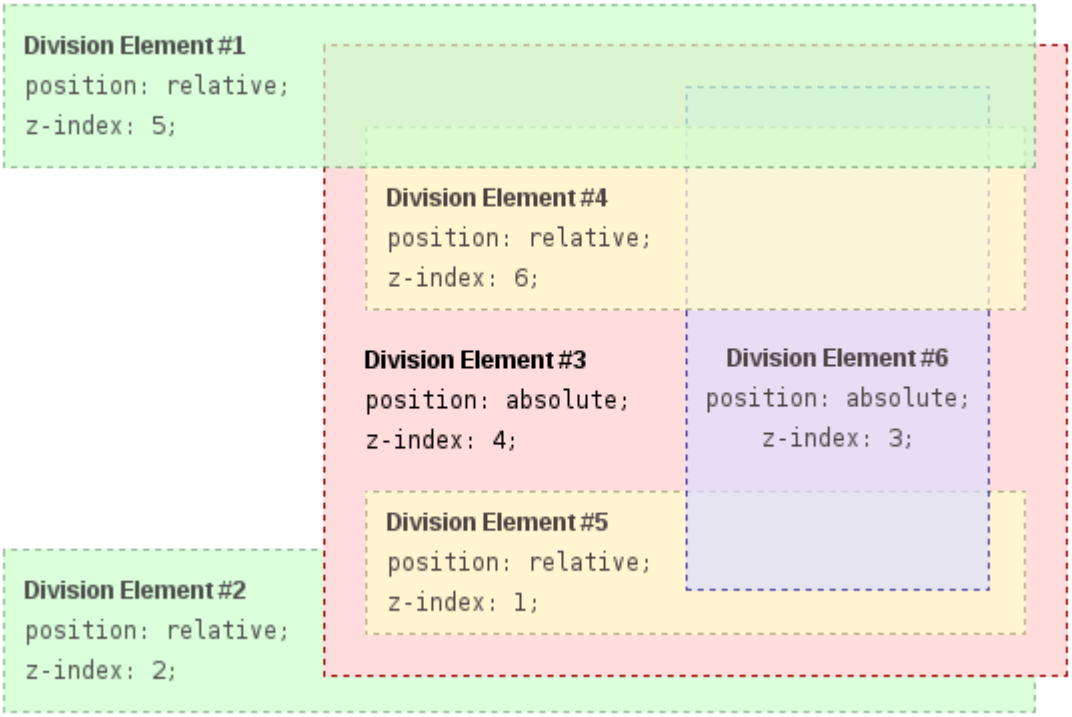
```
  <code>position: relative;<br/>
  z-index: 5;</code>
</div>
<div id="div2">
  <h1>Division Element #2</h1>
  <code>position: relative;<br/>
  z-index: 2;</code>
</div>
<div id="div3">
  <div id="div4">
    <h1>Division Element #4</h1>
    <code>position: relative;<br/>
    z-index: 6;</code>
  </div>
  <h1>Division Element #3</h1>
  <code>position: absolute;<br/>
  z-index: 4;</code>
  <div id="div5">
    <h1>Division Element #5</h1>
    <code>position: relative;<br/>
    z-index: 1;</code>
  </div>
  <div id="div6">
    <h1>Division Element #6</h1>
    <code>position: absolute;<br/>
    z-index: 3;</code>
  </div>
</div>
```

CSS:

```
* {
    margin: 0;
}
html {
    padding: 20px;
    font: 12px/20px Arial, sans-serif;
}
div {
    opacity: 0.7;
    position: relative;
}
h1 {
    font: inherit;
    font-weight: bold;
}
#div1,
#div2 {
    border: 1px dashed #696;
    padding: 10px;
    background-color: #cfc;
}
#div1 {
    z-index: 5;
    margin-bottom: 190px;
}
#div2 {
    z-index: 2;
}
#div3 {
```

```
    z-index: 4;
    opacity: 1;
    position: absolute;
    top: 40px;
    left: 180px;
    width: 330px;
    border: 1px dashed #900;
    background-color: #fdd;
    padding: 40px 20px 20px;
}
#div4,
#div5 {
    border: 1px dashed #996;
    background-color: #ffc;
}
#div4 {
    z-index: 6;
    margin-bottom: 15px;
    padding: 25px 10px 5px;
}
#div5 {
    z-index: 1;
    margin-top: 15px;
    padding: 5px 10px;
}
#div6 {
    z-index: 3;
    position: absolute;
    top: 20px;
    left: 180px;
    width: 150px;
    height: 125px;
    border: 1px dashed #009;
    padding-top: 125px;
    background-color: #ddf;
    text-align: center;
}
```

Ergebnis:



Quelle: https://developer.mozilla.org/en-US/docs/Web/CSS/CSS_Positioning/Understanding_z_index/The_stacking_context .

Stapelkontext online lesen: https://riptutorial.com/de/css/topic/5037/stapelkontext

Kapitel 47: Struktur und Formatierung einer CSS-Regel

Bemerkungen

Um die Lesbarkeit zu erleichtern, lassen Sie alle Deklarationen um eine Ebene von ihrem Selector eingerückt und die schließende geschweifte Klammer in einer eigenen Zeile. Fügen Sie nach Selektoren und Doppelpunkten ein einzelnes Leerzeichen ein und setzen Sie nach der Enddeklaration immer ein Semikolon.

Gut

```
p {
    color: maroon;
    font-size: 16px;
}
```

Schlecht

```
p{
 color: maroon;
font-size:16px }
```

Einzeiler

Wenn es nur eine oder zwei Deklarationen gibt, kommen Sie *möglicherweise* mit dieser Deklaration davon. In den meisten Fällen nicht empfohlen. Seien Sie immer konsequent wenn möglich.

```
p { color: maroon; font-size: 16px; }
```

Examples

Regeln, Selektoren und Deklarationsblöcke

Eine CSS- **Regel** besteht aus einem **Selektor** (zB `h1`) und einem **Deklarationsblock** (`{}`).

```
h1 {}
```

Immobilienlisten

Einige Eigenschaften können mehrere Werte annehmen, die zusammen als **Eigenschaftsliste bezeichnet werden** .

```
/* Two values in this property list */
span {
    text-shadow: yellow 0 0 3px, green 4px 4px 10px;
}

/* Alternate Formatting */
span {
    text-shadow:
        yellow 0 0 3px,
        green 4px 4px 10px;
}
```

Multiple Selektoren

Wenn Sie CSS-Selektoren gruppieren, wenden Sie dieselben Stile auf verschiedene Elemente an, ohne die Stile in Ihrem Stylesheet zu wiederholen. Verwenden Sie ein Komma, um mehrere gruppierte Selektoren voneinander zu trennen.

```
div, p { color: blue }
```

Die blaue Farbe gilt also für alle `<div>` -Elemente und alle `<p>` -Elemente. Ohne das Komma wären nur `<p>` -Elemente, die ein `<div>` , rot.

Dies gilt auch für alle Arten von Selektoren.

```
p, .blue, #first, div span{ color : blue }
```

Diese Regel gilt für:

- `<p>`
- Elemente der `blue` Klasse
- Element mit der ID `first`
- jedes `<span>` innerhalb eines `<div>`

Struktur und Formatierung einer CSS-Regel online lesen:
https://riptutorial.com/de/css/topic/4313/struktur-und-formatierung-einer-css-regel

Kapitel 48: Tabellen

Syntax

- Tabellenlayout: *auto* | Fest;
- Grenzzusammenbruch: *separate* | Zusammenbruch;
- Randabstand: <Länge> | <Länge> <Länge>;
- leere Zellen: *show* | verbergen;
- Bildunterschriftseite: *oben* | Unterseite;

Bemerkungen

Diese Eigenschaften gelten sowohl für `<table>` -Elemente (*) als auch für HTML-Elemente, die als `display: table` oder `display: inline-table` angezeigt werden

(*) `<table>` -Elemente werden offensichtlich von UA / Browsern nativ als `display: table`

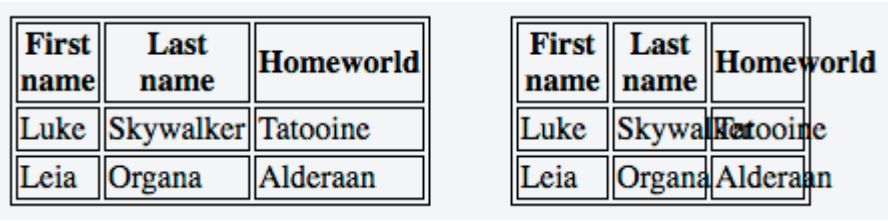
HTML-Tabellen sind semantisch für Tabellendaten gültig. Es wird nicht empfohlen, Tabellen für das Layout zu verwenden. Verwenden Sie stattdessen CSS.

Examples

Tabellenlayout

Die `table-layout` Eigenschaft ändert den Algorithmus, der für das Layout einer Tabelle verwendet wird.

Nachfolgend ein Beispiel für zwei Tabellen, deren `width: 150px` :



Die Tabelle links hat ein `table-layout: auto` während die `table-layout: fixed` rechts ein `table-layout: fixed` hat `table-layout: fixed` . Ersteres ist breiter als die angegebene Breite (210px statt 150px), aber der Inhalt passt. Letzteres nimmt die definierte Breite von 150px an, unabhängig davon, ob der Inhalt überläuft oder nicht.

Wert	Beschreibung
Auto	Dies ist der Standardwert. Sie definiert das Layout der Tabelle, das durch den Inhalt der Zellen bestimmt wird.

Wert	Beschreibung
Fest	Dieser Wert legt das Tabellenlayout so fest, dass es von der in der Tabelle angegebenen width -Eigenschaft bestimmt wird. Wenn der Inhalt einer Zelle diese Breite überschreitet, ändert sich die Größe der Zelle nicht, sondern der Inhalt wird überlaufen.

Grenzzusammenbruch

Die Eigenschaft `border-collapse` legt fest, ob die Grenzen einer Tabelle getrennt oder zusammengefügt werden sollen.

Nachfolgend ein Beispiel für zwei Tabellen mit unterschiedlichen Werten für die Eigenschaft `border-collapse` :

First name	Last name	Homeworld
Luke	Skywalker	Tatooine
Leia	Organa	Alderaan

First name	Last name	Homeworld
Luke	Skywalker	Tatooine
Leia	Organa	Alderaan

Die Tabelle auf der linken Seite hat `border-collapse: separate` während die Tabelle auf der rechten Seite `border-collapse: collapse`

Wert	Beschreibung
trennen	Dies ist der Standardwert. Dadurch werden die Grenzen der Tabelle voneinander getrennt.
Zusammenbruch	Dieser Wert legt fest, dass die Rahmen der Tabelle zusammengefügt werden, anstatt dass sie voneinander getrennt sind.

Randabstand

Die `border-spacing` bestimmt den Abstand zwischen Zellen. Dies hat keine Auswirkungen , wenn `border-collapse` gesetzt zu `separate` .

Unten finden Sie ein Beispiel für zwei Tabellen mit unterschiedlichen Werten für die Eigenschaft " `border-spacing` :

First name	Last name	Homeworld
Luke	Skywalker	Tatooine
Leia	Organa	Alderaan

First name	Last name	Homeworld
Luke	Skywalker	Tatooine
Leia	Organa	Alderaan

Die Tabelle links hat einen `border-spacing: 2px` (Standardeinstellung), während die rechte Tabelle einen `border-spacing: 2px` hat `border-spacing: 8px` .

Wert	Beschreibung
<Länge>	Dies ist das Standardverhalten, obwohl der genaue Wert zwischen Browsern variieren kann.
<Länge> <Länge>	Diese Syntax ermöglicht die Angabe separater horizontaler und vertikaler Werte.

leere Zellen

Die Eigenschaft `empty-cells` legt fest, ob Zellen ohne Inhalt angezeigt werden sollen oder nicht. Dies hat keine Auswirkungen , wenn `border-collapse` gesetzt zu `separate` .

Nachfolgend ein Beispiel mit zwei Tabellen mit unterschiedlichen Werten für die Eigenschaft `empty-cells` :

First name	Last name	Homeworld
Luke		Tatooine
Leia	Organa	

First name	Last name	Homeworld
Luke		Tatooine
Leia	Organa	

Die Tabelle links enthält `empty-cells: show` während die rechte Tabelle `empty-cells: hide` . Ersteres zeigt die leeren Zellen an, letzteres jedoch nicht.

Wert	Beschreibung
Show	Dies ist der Standardwert. Es zeigt Zellen auch wenn sie leer sind.
verbergen	Dieser Wert verbirgt eine Zelle insgesamt, wenn die Zelle keinen Inhalt enthält.

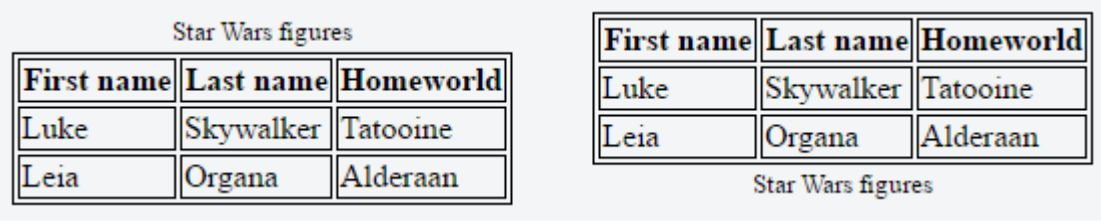
Mehr Informationen:

- https://www.w3.org/TR/CSS21/tables.html#empty-cells
- https://developer.mozilla.org/en-US/docs/Web/CSS/empty-cells
- http://codepen.io/SitePoint/pen/yfhtq
- https://css-tricks.com/almanac/properties/e/empty-cells/

Bildunterschriften

Die Eigenschaft `caption-side` bestimmt die vertikale Positionierung des Elements `<caption>` innerhalb einer Tabelle. Dies hat keine Auswirkungen, wenn ein solches Element nicht vorhanden ist.

Nachfolgend ein Beispiel mit zwei Tabellen mit unterschiedlichen Werten für die Eigenschaft `caption-side` :



Die Tabelle auf der linken Seite hat eine `caption-side: top` während die Tabelle rechts eine `caption-side: bottom` hat `caption-side: bottom` .

Wert	Beschreibung
oben	Dies ist der Standardwert. Die Beschriftung wird über der Tabelle platziert.
Unterseite	Dieser Wert platziert die Beschriftung unter der Tabelle.

Tabellen online lesen: https://riptutorial.com/de/css/topic/1074/tabellen

Kapitel 49: Typografie

Syntax

- font: [*font-style*] *[font-variant] [font-weight] Schriftgröße [/ Zeilenhöhe] Schriftfamilie* ;
- font-style: *font-style*
- Font-Variante: *Font-Variante*
- Schriftgewicht: *Schriftgewicht* ;
- Schriftgröße: *Schriftgröße* ;
- Zeilenhöhe: *Zeilenhöhe* ;
- Schriftfamilie: *Schriftfamilie* ;
- Farbe: *Farbe* ;
- Anführungszeichen: *keine* | *string* | *initial* |
- font-stretch: *font-stretch* ;
- Text ausrichten: *Text ausrichten* ;
- text-indent: *length* | *initial* |
- Textüberlauf: *Clip* | *Auslassungszeichen* | *Zeichenfolge* | *Anfang* | *Vererbung* ;
- Text-Transformation: *keine* | *Großschreibung* | *Großbuchstaben* | *Kleinschreibung* | *Anfangs* | *Vererbung* ;
- Text-Schatten: *h-Schatten v-Schatten-Unschärfe-Radius Farbe* | *keine* | *anfänglich* | *erben* ;
- font-size-adjust: *number* | *none* | *initial* | *vererben;*
- Font-Stretch: *ultrakondensiert* | *extrakondensiert* | *kondensiert* | *halbkondensiert* | *normal* | *halb expandiert* | *expandiert* | *extra expandiert* | *ultra expandiert* | *initial* |
- Bindestriche: *keine* | *Handbuch* | *auto* ;
- Tab-Größe: *Nummer* | *Länge* | *Anfang* | *erben* ;
- Buchstabenabstand: *Normal* | *Länge* | *Anfangs* | *Vererbung* ;
- Wortabstand: *Normal* | *Länge* | *Anfang* | *Vererbung* ;

Parameter

Parameter	Einzelheiten
Schriftstil	`italics` oder `oblique`
Schriftvariante	`normal` oder `small-caps`
Schriftgewicht	`normal` , `bold` oder numerisch von 100 bis 900.
Schriftgröße	Die Schriftgröße, angegeben in `%` , `px` , `em` oder einer anderen gültigen CSS-Messung
Zeilenhöhe	Die Zeilenhöhe in `%` , `px` , `em` oder eine andere gültige CSS-Messung
Schriftfamilie	Hiermit wird der Familienname definiert.

Parameter	Einzelheiten
Farbe	Jede gültige CSS-Farbdarstellung , wie `red` , `#00FF00` , `hsl(240, 100%, 50%)` usw.
Font-Stretch	Gibt an, ob eine konfektionierte oder erweiterte Fläche von der Schriftart verwendet werden soll. Gültige Werte sind `normal` , `ultra-condensed` , `extra-condensed` , `condensed` , `semi-condensed` , `semi-expanded` , `expanded` , `extra-expanded` oder `ultra-expanded`
Textausrichtung	`start` , `end` , `left` , `right` , `center` , `justify` , `match-parent`
Textdekoration	`none` , `underline` , `overline` , `line-through` , `initial` , `inherit` ;

Bemerkungen

- Die `text-shadow` Eigenschaft wird von Versionen von Internet Explorer weniger als 10 unterstützt.

Examples

Schriftgröße

HTML:

```
<div id="element-one">Hello I am some text.</div>
<div id="element-two">Hello I am some smaller text.</div>
```

CSS:

```
#element-one {
    font-size: 30px;
}

#element-two {
    font-size: 10px;
}
```

Der Text in `#element-one` wird `30px` groß sein, während der Text in `#element-two` `10px` groß ist.

Die Schriftkurzschrift

Mit der Syntax:

```
element {
    font: [font-style] [font-variant] [font-weight] [font-size/line-height] [font-family];
}
```

Sie können alle Ihre Schriftstile in einer Deklaration mit der Kurzschrift für `font` . Verwenden Sie einfach die `font` Eigenschaft und ordnen Sie Ihre Werte in der richtigen Reihenfolge an.

Wenn Sie beispielsweise alle `p` Elemente mit einer Schriftgröße von 20px fett formatieren und Arial als Schriftfamilie verwenden möchten, müssen Sie sie normalerweise wie folgt codieren:

```
p {
    font-weight: bold;
    font-size: 20px;
    font-family: Arial, sans-serif;
}
```

Mit der Schriftkurzschrift kann sie jedoch wie folgt komprimiert werden:

```
p {
    font: bold 20px Arial, sans-serif;
}
```

Hinweis : Da `font-style` , `font-variant` , `font-weight` und `line-height` optional sind, werden die drei in diesem Beispiel übersprungen. Es ist wichtig zu beachten , dass die Verknüpfung mit **setzt** die anderen Attribute nicht gegeben. Ein weiterer wichtiger Punkt ist , dass die beiden notwendigen Eigenschaften für die Schriftart Verknüpfung zu arbeiten , sind `font-size` und `font-family` . Wenn beide nicht enthalten sind, wird die Verknüpfung ignoriert.

Anfangswert für jede der Eigenschaften:

- `font-style: normal;`
- `font-variant: normal;`
- `font-weight: normal;`
- `font-stretch: normal;`
- `font-size: medium;`
- `line-height: normal;`
- `font-family` - hängt vom Benutzeragenten ab

Schriftenstapel

```
font-family: 'Segoe UI', Tahoma, sans-serif;
```

Der Browser versucht, die Schriftart "Segoe UI" auf die Zeichen in den Elementen anzuwenden, auf die die obige Eigenschaft abzielt. Wenn diese Schriftart nicht verfügbar ist oder die Schriftart keine Glyphe für das erforderliche Zeichen enthält, greift der Browser auf Tahoma und gegebenenfalls auf alle serifenlosen Schriftarten auf dem Computer des Benutzers zurück. Beachten Sie, dass für Schriftnamen mit mehr als einem Wort wie "Segoe UI" einfache oder doppelte Anführungszeichen verwendet werden müssen.

```
font-family: Consolas, 'Courier New', monospace;
```

Der Browser versucht, die Schriftart "Consolas" auf die Zeichen in den Elementen anzuwenden, auf die sich die oben genannte Eigenschaft bezieht. Wenn diese Schriftart nicht verfügbar ist oder

die Schriftart keine Glyphe für das erforderliche Zeichen enthält, wird der Browser auf "Courier New" und gegebenenfalls eine Monospace-Schriftart auf dem Computer des Benutzers zurückgesetzt.

Buchstaben-Abstand

```
h2 {
  /* adds a 1px space horizontally between each letter;
     also known as tracking */
  letter-spacing: 1px;
}
```

Mit der Eigenschaft für den Buchstabenabstand wird der Abstand zwischen den Zeichen in einem Text angegeben.

! Buchstabenabstand unterstützt auch negative Werte:

```
p {
  letter-spacing: -1px;
}
```

Ressourcen: https://developer.mozilla.org/en-US/docs/Web/CSS/letter-spacing

Text umwandeln

Mit `text-transform` Eigenschaft `text-transform` können Sie die Großschreibung von Text ändern. Gültige Werte sind: `uppercase` , `capitalize` , `lowercase` , `initial` , `inherit` , und `none`

CSS:

```
.example1 {
    text-transform: uppercase;
}
.example2 {
    text-transform: capitalize;
}
.example3 {
    text-transform: lowercase;
}
```

HTML

```
<p class="example1">
    all letters in uppercase  <!-- "ALL LETTERS IN UPPERCASE" -->
</p>
<p class="example2">
    all letters in capitalize <!-- "All Letters In Capitalize (Sentence Case)" -->
</p>
<p class="example3">
    all letters in lowercase  <!-- "all letters in lowercase" -->
</p>
```

Texteinzug

```
p {
    text-indent: 50px;
}
```

Die Eigenschaft `text-indent` gibt an, um wie viel horizontaler Leerzeichen Text vor dem Anfang der ersten Zeile des Textinhalts eines Elements verschoben werden soll.

Ressourcen:

- Einrücken nur der ersten Textzeile in einem Absatz?
- https://www.w3.org/TR/CSS21/text.html#propdef-text-indent
- https://developer.mozilla.org/en-US/docs/Web/CSS/text-indent

Textdekoration

Die `text-decoration` wird zum Festlegen oder Entfernen von Dekorationen aus Text verwendet.

```
h1 { text-decoration: none; }
h2 { text-decoration: overline; }
h3 { text-decoration: line-through; }
h4 { text-decoration: underline; }
```

Textdekoration kann in Kombination mit Textdekorationsstil und Textdekorationsfarbe als Abkürzungseigenschaft verwendet werden:

```
.title { text-decoration: underline dotted blue; }
```

Dies ist eine Kurzfassung von

```
.title {
  text-decoration-style: dotted;
  text-decoration-line: underline;
  text-decoration-color: blue;
}
```

Es ist zu beachten, dass die folgenden Eigenschaften nur in Firefox unterstützt werden

- Text-Deko-Farbe
- Textdekorationslinie
- Text-Deko-Stil
- Textdekoration-Überspringen

Textüberlauf

Die Eigenschaft `text-overflow` beschäftigt sich damit, wie überlaufender Inhalt Benutzern angezeigt werden soll. In diesem Beispiel stehen die `ellipsis` für abgeschnittenen Text.

```
.text {
  overflow: hidden;
  text-overflow: ellipsis;
}
```

`text-overflow: ellipsis` funktionieren leider nur für eine einzelne Textzeile. Es gibt keine Möglichkeit, Ellipsen in der letzten Zeile des Standard-CSS zu unterstützen. Dies kann jedoch mit einer nicht standardmäßigen Webkit-Implementierung von Flexboxen erreicht werden.

```
.giveMeEllipsis {
   overflow: hidden;
   text-overflow: ellipsis;
   display: -webkit-box;
   -webkit-box-orient: vertical;
   -webkit-line-clamp: N; /* number of lines to show */
   line-height: X;        /* fallback */
   max-height: X*N;       /* fallback */
}
```

Beispiel (in Chrome oder Safari öffnen):

http://jsfiddle.net/csYjC/1131/

Ressourcen:

https://www.w3.org/TR/2012/WD-css3-ui-20120117/#text-overflow0

Wortabstand

Die Wortabstandseigenschaft gibt das Abstandsverhalten zwischen Tags und Wörtern an.

Mögliche Werte

- eine positive oder negative *Länge* (mit `em` `px` `vh` `cm` usw.) oder *Prozentsatz* (mit `%`)
- Das Schlüsselwort `normal` verwendet den voreingestellten Wortabstand der Schriftart
- Das Schlüsselwort `inherit` übernimmt den Wert des übergeordneten Elements

CSS

```
.normal    { word-spacing: normal; }
.narrow    { word-spacing: -3px; }
.extensive { word-spacing: 10px; }
```

HTML

```
<p>
  <span class="normal">This is an example, showing the effect of "word-spacing".</span><br>
  <span class="narrow">This is an example, showing the effect of "word-spacing".</span><br>
  <span class="extensive">This is an example, showing the effect of "word-spacing".</span><br>
</p>
```

Online-Demo

Versuch es selber

Lesen Sie weiter:

- Wortabstand - MDN
- Wortabstand - w3.org

Textrichtung

```
div {
    direction: ltr; /* Default, text read read from left-to-right */
}
.ex {
    direction: rtl; /* text read from right-to-left */
}
.horizontal-tb {
    writing-mode: horizontal-tb; /* Default, text read from left-to-right and top-to-bottom.
*/
}
.vertical-rtl {
    writing-mode: vertical-rl; /* text read from right-to-left and top-to-bottom */
}
.vertical-ltr {
    writing-mode: vertical-rl; /* text read from left-to-right and top to bottom */
}
```

Die `direction` Eigenschaft wird verwendet, um die horizontale Textrichtung eines Elements zu ändern.

Syntax: `direction: ltr | rtl | initial | inherit;`

Der `writing-mode` Eigenschaft ändert die Ausrichtung des Textes , so dass es von oben nach unten oder von links nach rechts gelesen werden, abhängig von der Sprache.

Syntax: `direction: horizontal-tb | vertical-rl | vertical-lr;`

Schriftvariante

Attribute:

normal

Standardattribut der Schriftarten.

Small Caps

Setzt jeden Buchstaben in Großbuchstaben, **sondern** macht die Kleinbuchstaben (ab ursprünglichem Text) in der Größe kleiner als die Buchstaben , die ursprünglich in Großbuchstaben.

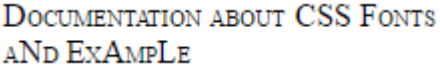
CSS:

```
.smallcaps{
   font-variant: small-caps;
}
```

HTML:

```
<p class="smallcaps">
    Documentation about CSS Fonts
    <br>
    aNd ExAmpLe
</p>
```

OUPUT:



Anmerkung: Die Eigenschaft font-variant ist eine Abkürzung für die Eigenschaften: font-variant-caps, font-variant-numerisch, font-variant-alternates, font-variant-ligaturen und font-variant-east-asian.

Zitate

Die `quotes` Eigenschaft wird verwendet, um die öffnenden und schließenden Anführungszeichen des `<q>` -Tags anzupassen.

```
q {
    quotes: "«" "»";
}
```

Textschatten

Um dem `text-shadow` hinzuzufügen, verwenden Sie die Eigenschaft `text-shadow` . Die Syntax lautet wie folgt:

```
text-shadow: horizontal-offset vertical-offset blur color;
```

Schatten ohne Unschärfekreis

```
h1 {
  text-shadow: 2px 2px #0000FF;
}
```

Dies erzeugt einen blauen Schatteneffekt um eine Überschrift

Schatten mit Unschärfekreis

Um einen Weichzeichnungseffekt hinzuzufügen, fügen Sie ein Argument für den `blur radius`

```
h1 {
  text-shadow: 2px 2px 10px #0000FF;
}
```

Mehrere Schatten

Um einem Element mehrere Schatten zu geben, trennen Sie diese durch Kommas

```
h1 {
    text-shadow: 0 0 3px #FF0000, 0 0 5px #0000FF;
}
```

Typografie online lesen: https://riptutorial.com/de/css/topic/427/typografie

Kapitel 50: Übergänge

Syntax

- Übergang: [Übergangseigenschaft] [Übergangsdauer] [Übergangszeitfunktion] [Übergangsverzögerung];

Parameter

Parameter	Einzelheiten
Übergangseigenschaft	Die spezifische CSS-Eigenschaft, deren Wertänderung umgestellt werden muss (oder `all` , wenn alle umschaltbaren Eigenschaften umgestellt werden müssen.
Übergangszeit	Die Dauer (oder Periode) in Sekunden (`s`) oder Millisekunden (`ms`), über die der Übergang stattfinden muss.
Übergangszeitfunktion	Eine Funktion, die beschreibt, wie die Zwischenwerte während des Übergangs berechnet werden. Häufig verwendete Werte sind `ease` , `ease-in` , `ease-out` , `ease-in-out` , `linear` , `cubic-bezier()` , die `steps()` . Weitere Informationen zu den verschiedenen Timing-Funktionen finden Sie in den W3C-Spezifikationen .
Übergangsverzögerung	Die Zeit, die vergangen sein muss, bevor der Übergang beginnen kann. Kann in Sekunden (`s`) oder Millisekunden (`ms`) angegeben werden

Bemerkungen

Einige ältere Browser unterstützen nur `transition` Herstellerpräfix :

- `-webkit` : Chrome 25-, Safari 6-, Safari & Chrome für iOS 6.1-, Android 4.3- Browser, Blackberry Browser 7-, UC Browser 9.9- für Android.
- `-moz` : Firefox 15-.
- `-o` : Opera 11.5-, Opera Mobile 12-.

Beispiel:

```
-webkit-transition: all 1s;
-moz-transition: all 1s;
-o-transition: all 1s;
transition: all 1s;
```

Examples

Übergangskurzschrift

CSS

```
div{
    width: 150px;
    height:150px;
    background-color: red;
    transition: background-color 1s;
}
div:hover{
    background-color: green;
}
```

HTML

```
<div></div>
```

In diesem Beispiel wird die Hintergrundfarbe geändert, wenn das div angezeigt wird.

Transition (Longhand)

CSS

```
div {
  height: 100px;
  width: 100px;
  border: 1px solid;
  transition-property: height, width;
  transition-duration: 1s, 500ms;
  transition-timing-function: linear;
  transition-delay: 0s, 1s;
}
div:hover {
  height: 200px;
  width: 200px;
}
```

HTML

```
<div></div>
```

- **Übergangseigenschaft** : Gibt die CSS-Eigenschaften an, für die der Übergangseffekt gilt. In diesem Fall wird das div sowohl horizontal als auch vertikal erweitert, wenn es im Hover-Modus ist.
- **Übergangsdauer** : Gibt die Zeitdauer an, für die ein Übergang abgeschlossen ist. Im obigen

Beispiel dauern die Höhen- und Breitenübergänge 1 Sekunde bzw. 500 Millisekunden.

- **Transition-Timing-Funktion** : Bestimmt die Geschwindigkeitskurve des Übergangseffekts. Ein *linearer* Wert zeigt an, dass der Übergang vom Anfang bis zum Ende die gleiche Geschwindigkeit hat.
- **Übergangsverzögerung** : Gibt an, wie lange es dauert, bis der Übergangseffekt beginnt. In diesem Fall beginnt die Höhe sofort mit dem Übergang, während die Breite 1 Sekunde wartet.

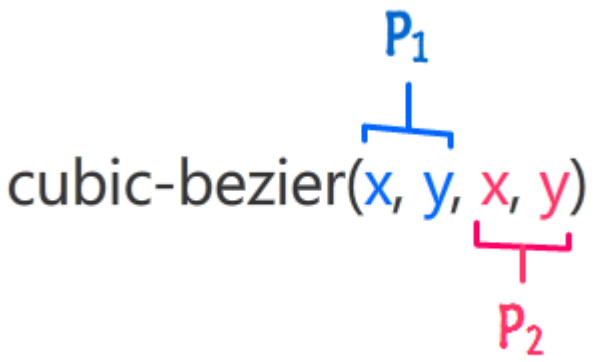
Kubik-Bezier

Die `cubic-bezier` Funktion ist eine Übergangszeitfunktion, die häufig für benutzerdefinierte und glatte Übergänge verwendet wird.

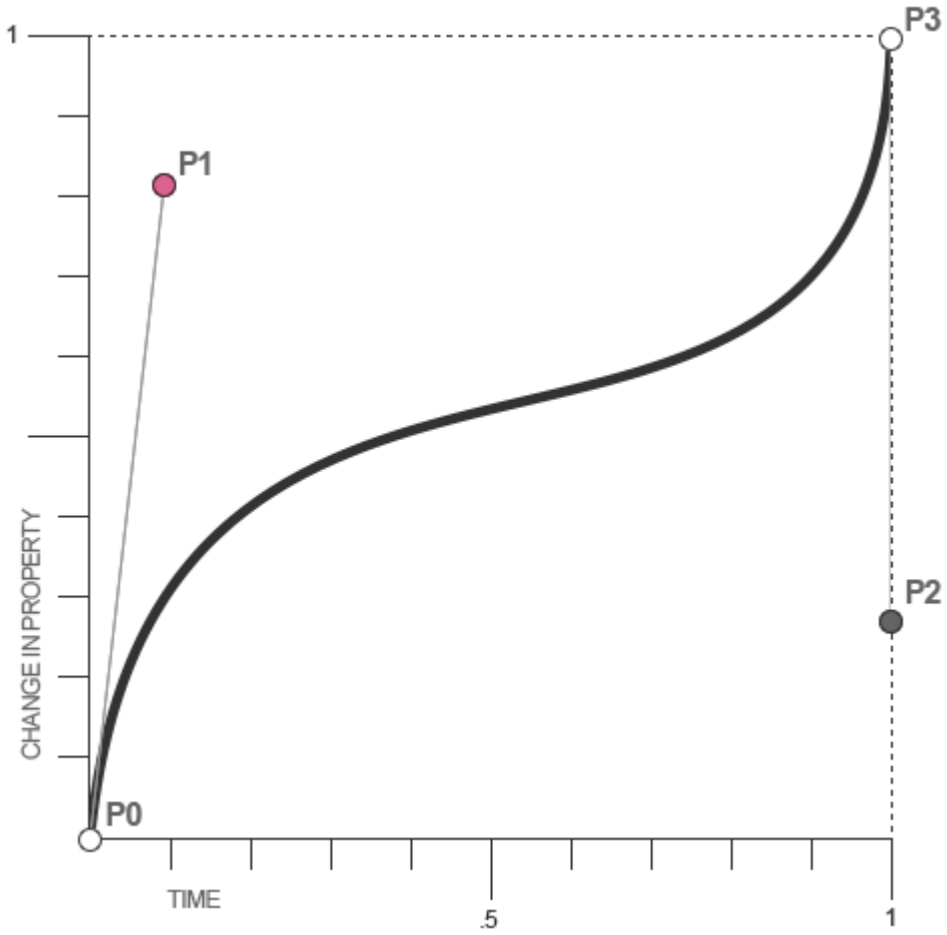
```
transition-timing-function: cubic-bezier(0.1, 0.7, 1.0, 0.1);
```

Die Funktion benötigt vier Parameter:

```
cubic-bezier(P1_x, P1_y, P2_x, P2_y)
```



Diese Parameter werden auf Punkte abgebildet, die Teil einer Bézier-Kurve sind :



Bei CSS-Bézier-Kurven liegen P0 und P3 immer an derselben Stelle. P0 liegt bei (0,0) und P3 bei (1,1), was bedeutet, dass die an die Cubic-Bezier-Funktion übergebenen Parameter nur zwischen 0 und 1 liegen können.

Wenn Sie Parameter übergeben, die nicht in diesem Intervall liegen, verwendet die Funktion standardmäßig einen `linear` Übergang.

Da Cubic-Bezier der flexibelste Übergang in CSS ist, können Sie alle anderen Übergangszeitfunktionen in Cubic-Bezier-Funktionen übersetzen:

`linear` : `cubic-bezier(0,0,1,1)`

`ease-in` : `cubic-bezier(0.42, 0.0, 1.0, 1.0)`

`ease-out` : `cubic-bezier(0.0, 0.0, 0.58, 1.0)`

`cubic-bezier(0.42, 0.0, 0.58, 1.0)` `ease-in-out` : `cubic-bezier(0.42, 0.0, 0.58, 1.0)`

Übergänge online lesen: https://riptutorial.com/de/css/topic/751/ubergange

Kapitel 51: Überlauf

Syntax

- Überlauf: sichtbar | versteckt | scrollen | auto | initial | erben;

Parameter

`Overflow`	Einzelheiten
`visible`	Zeigt alle überlaufenden Inhalte außerhalb des Elements an
`scroll`	Blendet den überlaufenden Inhalt aus und fügt eine Bildlaufleiste hinzu
`hidden`	Blendet den überlaufenden Inhalt aus, beide Bildlaufleisten werden ausgeblendet und die Seite wird fixiert
`auto`	Das gleiche wie `scroll` , wenn der Inhalt überläuft, aber keine Bildlaufleiste , wenn der Inhalt passt hinzufügen
`inherit`	Inherit ist der Wert des übergeordneten Elements für diese Eigenschaft

Bemerkungen

> Die `overflow` gibt an, ob Inhalt beschnitten, Bildlaufleisten gerendert oder ein Container gestreckt werden soll, um den Inhalt anzuzeigen, wenn der Blockcontainer überläuft.

Was passiert, wenn ein Element zu klein ist, um seinen Inhalt anzuzeigen? Standardmäßig wird der Inhalt nur überlaufen und außerhalb des Elements angezeigt. Das macht deine Arbeit schlecht. Sie möchten, dass Ihre Arbeit gut aussieht, so dass Sie die Überlaufeigenschaft so einstellen, dass der überlaufende Inhalt auf gewünschte Weise behandelt wird.

Die Werte für die `overflow` sind identisch mit denen für die Eigenschaften `overflow-x` und `overflow-y` , es sei denn, sie gelten für jede Achse

Die `overflow-wrap` Eigenschaft wurde auch als `word-wrap` Eigenschaft bezeichnet.

> Wichtiger Hinweis: Wenn Sie die Überlaufeigenschaft mit einem anderen Wert als sichtbar verwenden, wird ein neuer Blockformatierungskontext erstellt.

Examples

Überlauf: blättern

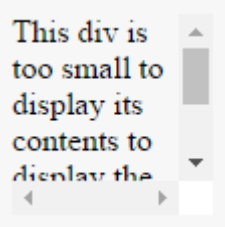
HTML

```
<div>
    This div is too small to display its contents to display the effects of the overflow
property.
</div>
```

CSS

```
div {
    width:100px;
    height:100px;
    overflow:scroll;
}
```

Ergebnis



Der oben aufgeführte Inhalt wird in ein 100 x 100 Pixel großes Feld geschnitten, wobei Bildlauf zur Anzeige überlaufender Inhalte möglich ist.

Die meisten Desktop-Browser zeigen sowohl horizontale als auch vertikale Bildlaufleisten an, unabhängig davon, ob Inhalte abgeschnitten werden. Dadurch können Probleme mit Bildlaufleisten in einer dynamischen Umgebung vermieden werden. Drucker drucken möglicherweise überlaufende Inhalte.

Überlaufwickel

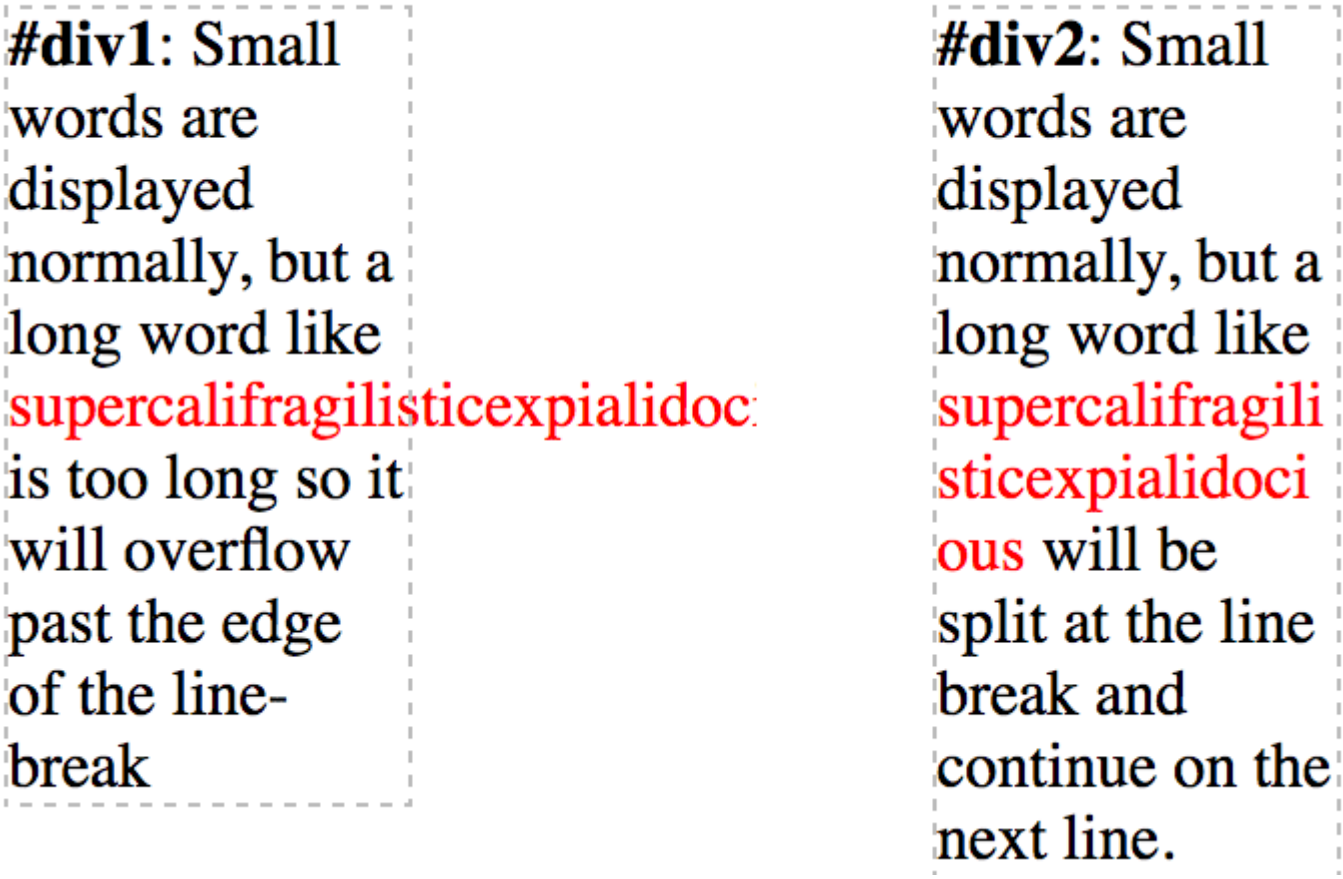
`overflow-wrap` teilt einem Browser mit, dass er eine Textzeile innerhalb eines Zielelements in mehrere Zeilen an einem ansonsten unzerbrechlichen Ort aufteilen kann. Hilfreich bei der Vermeidung einer langen Textfolge, die Layoutprobleme verursacht, weil der Container überläuft.

CSS

```
div {
    width:100px;
    outline: 1px dashed #bbb;
}

#div1 {
    overflow-wrap:normal;
}

#div2 {
    overflow-wrap:break-word;
}
```

HTML

```
<div id="div1">
      <strong>#div1</strong>: Small words are displayed normally, but a long word like <span
style="red;">supercalifragilisticexpialidocious</span> is too long so it will overflow past
the edge of the line-break
</div>

<div id="div2">
      <strong>#div2</strong>: Small words are displayed normally, but a long word like <span
style="red;">supercalifragilisticexpialidocious</span> will be split at the line break and
continue on the next line.
</div>
```



`overflow-wrap` - Wert	Einzelheiten
`normal`	Lässt ein Wort überlaufen, wenn es länger als die Zeile ist
`break-word`	Teilt ein Wort ggf. in mehrere Zeilen auf
`inherit`	Erbt den Wert des übergeordneten Elements für diese Eigenschaft

Überlauf: sichtbar

HTML

```
<div>
     Even if this div is too small to display its contents, the content is not clipped.
</div>
```

CSS

```
div {
     width:50px;
     height:50px;
     overflow:visible;
}
```

Ergebnis

Inhalt wird nicht abgeschnitten und außerhalb des Inhaltsrahmens angezeigt, wenn er die Containergröße überschreitet.

Block-Formatierungskontext, der mit Überlauf erstellt wurde

Wenn Sie die `overflow` Eigenschaft mit einem anderen Wert als `visible` wird ein neuer **Blockformatierungskontext erstellt** . Dies ist nützlich, wenn Sie ein Blockelement neben einem schwebenden Element ausrichten möchten.

CSS

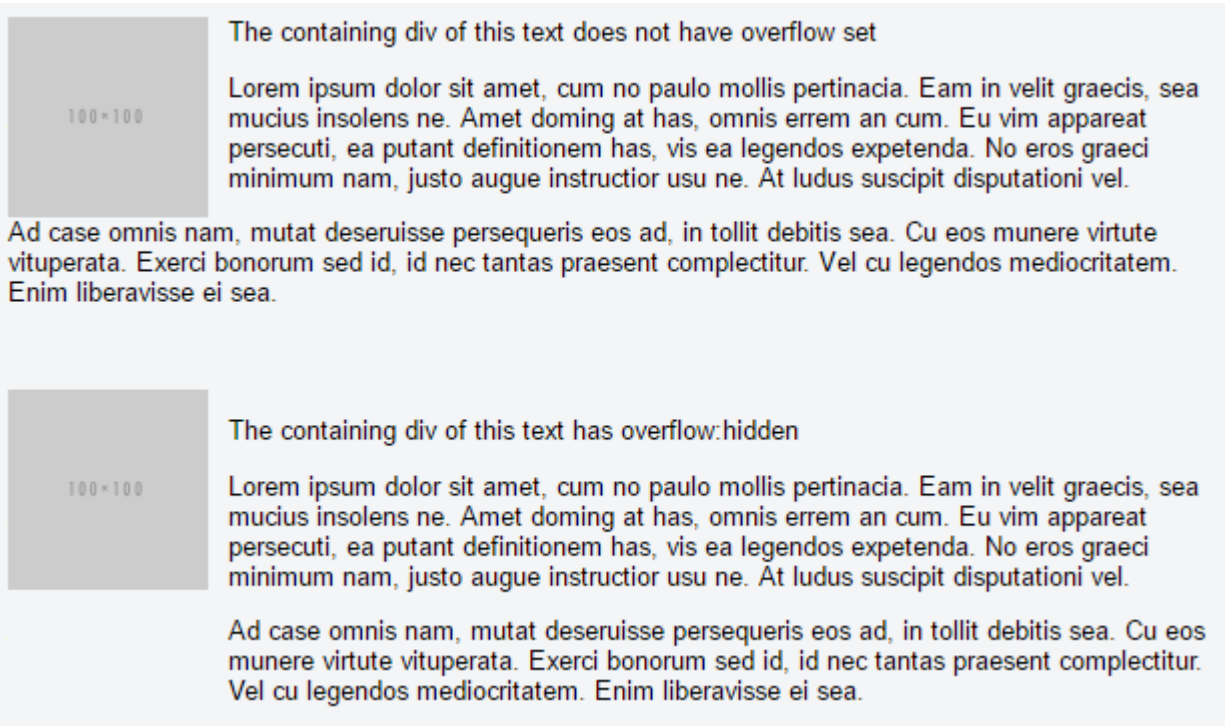
```
img {
    float:left;
    margin-right: 10px;
}
 div {
     overflow:hidden; /* creates block formatting context */
}
```

HTML

```
<img src="http://placehold.it/100x100">
<div>
  <p>Lorem ipsum dolor sit amet, cum no paulo mollis pertinacia.</p>
  <p>Ad case omnis nam, mutat deseruisse persequeris eos ad, in tollit debitis sea.</p>
```

```
</div>
```

Ergebnis



Dieses Beispiel zeigt, wie Absätze innerhalb eines div mit der Eigenschaft " `overflow` mit einem schwebenden Bild interagieren.

Überlauf-x und Überlauf-y

Diese beiden Eigenschaften arbeiten auf ähnliche Weise wie die `overflow` und akzeptieren dieselben Werte. Der Parameter `overflow-x` funktioniert nur für die x- oder die Links-Rechts-Achse. Der `overflow-y` arbeitet auf der y-Achse oder von oben nach unten.

HTML

```
<div id="div-x">
    If this div is too small to display its contents,
    the content to the left and right will be clipped.
</div>

<div id="div-y">
    If this div is too small to display its contents,
    the content to the top and bottom will be clipped.
</div>
```

CSS

```
div {
    width: 200px;
    height: 200px;
}
```

```
#div-x {
    overflow-x: hidden;
}

#div-y {
    overflow-y: hidden;
}
```

Überlauf online lesen: https://riptutorial.com/de/css/topic/4947/uberlauf

Kapitel 52: Umrisse

Syntax

- Umriss: Umrissfarbe Umrissbreite | initial | erben;
- Umrissbreite: mittel | dünn | dick | Länge | initial | erben;
- Gliederungsstil: keine | versteckt | gepunktet | gestrichelt | fest | doppelt | Nut | Grat | Einschub | Anfang | initial | erben;

Parameter

Parameter	Einzelheiten
gepunktet	gepunktete Kontur
gestrichelt	gestrichelte Kontur
solide	feste Kontur
doppelt	doppelter Umriss
Rille	3D-gerillte Umrisse hängen vom Umrissfarbenwert ab
Grat	Gerippte 3D-Konturen hängen vom Konturfarbenwert ab
Einsatz	Die 3D-Umrisslinie hängt vom Umrissfarbenwert ab
Anfang	Die 3D-Anfangsumrisslinie hängt vom Umrissfarbenwert ab
keiner	keine Gliederung
versteckt	versteckter Umriss

Bemerkungen

`outline` wird nun in Basic UI , einem CSS-Modul Level 3 (es wurde bereits in REC CSS2.1 beschrieben) beschrieben.

Die Gliederungseigenschaft ist in Browsern standardmäßig für fokussierbare Elemente in `:focus` Fokusstatus definiert.
Es sollte nicht entfernt werden, siehe http://outlinenone.com. Dort heißt es:

> **Was macht die Gliederungseigenschaft?**
>
> Es bietet visuelles Feedback für Links, die beim Navigieren in einem Webdokument mit

> der TAB-Taste (oder einer gleichwertigen Funktion) "Fokus" haben. Dies ist besonders nützlich für Leute, die keine Maus verwenden oder eine Sehbehinderung haben. Wenn Sie die Gliederung entfernen, machen Sie Ihre Site für diese Personen unzugänglich. (…)

Interessante verwandte Beispiele zum Stack Overflow:

- So entfernen Sie die Umrandung eines Eingabetextelements
- Wie entferne ich die gepunktete Umrisslinie von Firefox sowie die Links?

Examples

Überblick

Gliederung ist eine Linie, die um das Element außerhalb der Grenze verläuft. Im Gegensatz zu `border` , umreißt jeden Raum in dem Box - Modell nicht nehmen Sie. Das Hinzufügen einer Gliederung zu einem Element hat also keinen Einfluss auf die Position des Elements oder anderer Elemente.

Außerdem können Konturen in einigen Browsern nicht rechteckig sein. Dies kann passieren, wenn die `outline` auf ein `span` Element angewendet wird, in dem sich Text mit anderen Eigenschaften für die `font-size` . Im Gegensatz zu Rändern dürfen Konturen *keine* abgerundeten Ecken haben.

Die wesentlichen Bestandteile von `outline` sind `outline-color` , `outline-style` und `outline-width` .

Die Definition einer Gliederung entspricht der Definition einer Grenze:

> Eine Kontur ist eine Linie um ein Element. Es wird um den Rand des Elements angezeigt. Es unterscheidet sich jedoch von der Grenzeigenschaft.

```
outline: 1px solid black;
```

Umriss-Stil

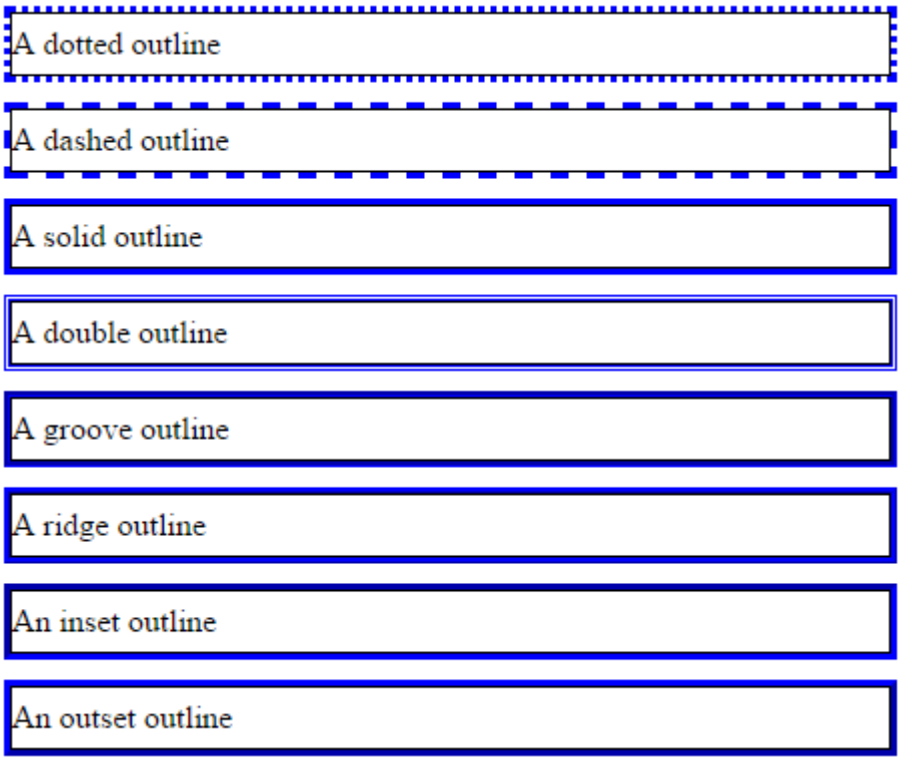
Mit `outline-style` Eigenschaft "Konturstil" wird der Stil der Kontur eines Elements festgelegt.

```
p {
  border: 1px solid black;
  outline-color:blue;
  line-height:30px;
}
.p1{
  outline-style: dotted;
}
.p2{
  outline-style: dashed;
}
.p3{
  outline-style: solid;
}
.p4{
```

```
  outline-style: double;
}
.p5{
  outline-style: groove;
}
.p6{
  outline-style: ridge;
}
.p7{
  outline-style: inset;
}
.p8{
  outline-style: outset;
}
```

HTML

```
<p class="p1">A dotted outline</p>
<p class="p2">A dashed outline</p>
<p class="p3">A solid outline</p>
<p class="p4">A double outline</p>
<p class="p5">A groove outline</p>
<p class="p6">A ridge outline</p>
<p class="p7">An inset outline</p>
<p class="p8">An outset outline</p>
```



Umrisse online lesen: https://riptutorial.com/de/css/topic/4258/umrisse

Kapitel 53: Vertikale Zentrierung

Bemerkungen

Dies wird verwendet, wenn die Abmessungen (`width` und `height`) des Elements nicht bekannt oder dynamisch sind.

Verwenden Sie **Flexbox lieber** als alle anderen Optionen, da sie für das Design der Benutzeroberfläche optimiert ist.

Examples

Zentrieren mit Display: Tabelle

HTML:

```
<div class="wrapper">
    <div class="outer">
        <div class="inner">
            centered
        </div>
    </div>
</div>
```

CSS:

```
.wrapper {
  height: 600px;
  text-align: center;
}
.outer {
  display: table;
  height: 100%;
  width: 100%;
}
.outer .inner {
  display: table-cell;
  text-align: center;
  vertical-align: middle;
}
```

Zentrieren mit Transformation

HTML:

```
<div class="wrapper">
    <div class="centered">
        centered
    </div>
</div>
```

CSS:

```
.wrapper {
  position: relative;
  height: 600px;
}
.centered {
  position: absolute;
  z-index: 999;
  transform: translate(-50%, -50%);
  top: 50%;
  left: 50%;
}
```

Zentrieren mit Flexbox

HTML:

```
<div class="container">
    <div class="child"></div>
</div>
```

CSS:

```
.container {
  height: 500px;
  width: 500px;
  display: flex;               // Use Flexbox
  align-items: center;         // This centers children vertically in the parent.
  justify-content: center;     // This centers children horizontally.
  background: white;
}

.child {
  width: 100px;
  height: 100px;
  background: blue;
}
```

Text mit Zeilenhöhe zentrieren

HTML:

```
<div class="container">
    <span>vertically centered</span>
</div>
```

CSS:

```
.container{
    height: 50px;           /* set height */
    line-height: 50px;      /* set line-height equal to the height */
    vertical-align: middle; /* works without this rule, but it is good having it explicitly
set */
```

```
}
```

Hinweis: Bei dieser Methode wird nur eine *einzelne Textzeile* vertikal *zentriert* . Blockelemente werden nicht korrekt zentriert, und wenn der Text in eine neue Zeile übergeht, werden zwei sehr lange Textzeilen angezeigt.

Zentrieren mit Position: absolut

HTML:

```
<div class="wrapper">
  <img src="http://cdn.sstatic.net/Sites/stackoverflow/company/img/logos/so/so-
icon.png?v=c78bd457575a">
</div>
```

CSS:

```
.wrapper{
   position:relative;
   height: 600px;
}
.wrapper img {
  position: absolute;
  top: 0;
  left: 0;
  right: 0;
  bottom: 0;
  margin: auto;
}
```

Wenn Sie andere Bilder zentrieren möchten, müssen Sie diesem Element Höhe und Breite zuweisen.

HTML:

```
<div class="wrapper">
  <div class="child">
     make me center
  </div>
</div>
```

CSS:

```
.wrapper{
   position:relative;
   height: 600px;
}
.wrapper .child {
  position: absolute;
  top: 0;
  left: 0;
  right: 0;
  bottom: 0;
```

```
  margin: auto;
  width: 200px;
  height: 30px;
  border: 1px solid #f00;
}
```

Zentrierung mit Pseudoelement

HTML:

```
<div class="wrapper">
  <div class="content"></div>
</div>
```

CSS:

```
.wrapper{
   min-height: 600px;
}

.wrapper:before{
  content: "";
  display: inline-block;
  height: 100%;
  vertical-align: middle;
}

.content {
  display: inline-block;
  height: 80px;
  vertical-align: middle;
}
```

Diese Methode wird am besten in Fällen verwendet, in denen der `.content` in `.wrapper` zentriert ist. und Sie möchten, `.wrapper` die Höhe von `.content` erweitert wird, wenn die Höhe von `.wrapper` die `.wrapper` Höhe von `.wrapper` überschreitet.

Vertikale Zentrierung online lesen: https://riptutorial.com/de/css/topic/5070/vertikale-zentrierung

Kapitel 54: Zähler

Syntax

- Gegensatz: [<Gegenname> <Ganzzahl>?] + | keiner
- counter-reset: [<countername> <integer>?] + | keiner
- Counter-Inkrement: [<Counter-Name> <Ganzzahl>?] + | keiner
- counter (<counter-name> [, <counter-style>]?)
- Zähler (<Counter-Name>, <Connector-String> [, <Counter-Style>]?)

Parameter

Parameter	Einzelheiten
Gegenname	Dies ist der Name des Zählers, der erstellt oder inkrementiert oder gedruckt werden muss. Es kann ein beliebiger benutzerdefinierter Name sein, wie es der Entwickler wünscht.
ganze Zahl	Diese Ganzzahl ist ein optionaler Wert, der, wenn er neben dem Zählernamen angegeben wird, den Anfangswert des Zählers (in den `counter-set` `counter-reset` Eigenschaften) oder den Wert darstellt, um den der Zähler erhöht werden soll (in `counter-increment`).
keiner	Dies ist der Anfangswert für alle 3 `counter-*` -Eigenschaften. Wenn dieser Wert für die `counter-increment` , ist der Wert von keinem der Zähler betroffen. Wenn dies für die anderen beiden verwendet wird, wird kein Zähler erstellt.
Gegenstil	Dies gibt den Stil an, in dem der Zählerwert angezeigt werden muss. Es werden alle Werte unterstützt, die von der Eigenschaft `list-style-type` unterstützt werden. Wenn `none` wird, wird der Zählerwert überhaupt nicht gedruckt.
Verbindungszeichenfolge	Dies ist die Zeichenfolge, die zwischen den Werten von zwei verschiedenen Zählerständen (wie "." In "2.1.1") platziert werden muss.

Bemerkungen

Zähler sind kein neues Thema in CSS. Es war Teil der CSS Level 2-Spezifikationen (Revision 1,

um genau zu sein) und bietet daher eine sehr hohe Browserunterstützung.

Alle Browser außer IE6 und IE7 unterstützen CSS-Zähler.

Examples

Anwenden von römischen Zahlen auf die Zählerausgabe

CSS

```
body {
  counter-reset: item-counter;
}

.item {
  counter-increment: item-counter;
}

.item:before {
  content: counter(item-counter, upper-roman) ". "; /* by specifying the upper-roman as style
the output would be in roman numbers */
}
```

HTML

```
<div class='item'>Item No: 1</div>
<div class='item'>Item No: 2</div>
<div class='item'>Item No: 3</div>
```

Im obigen Beispiel würde die Ausgabe des Zählers als I, II, III (römische Zahlen) anstelle der üblichen 1, 2, 3 angezeigt, da der Entwickler den Stil des Zählers explizit angegeben hat.

Nummerieren Sie jeden Artikel mit dem CSS-Zähler

CSS

```
body {
  counter-reset: item-counter; /* create the counter */
}
.item {
  counter-increment: item-counter; /* increment the counter every time an element with class
"item" is encountered */
}
.item-header:before {
  content: counter(item-counter) ". "; /* print the value of the counter before the header and
append a "." to it */
}
```

```
/* just for demo */

.item {
  border: 1px solid;
  height: 100px;
  margin-bottom: 10px;
}
.item-header {
  border-bottom: 1px solid;
  height: 40px;
  line-height: 40px;
  padding: 5px;
}
.item-content {
  padding: 8px;
}
```

HTML

```
<div class='item'>
  <div class='item-header'>Item 1 Header</div>
  <div class='item-content'>Lorem Ipsum Dolor Sit Amet....</div>
</div>
<div class='item'>
  <div class='item-header'>Item 2 Header</div>
  <div class='item-content'>Lorem Ipsum Dolor Sit Amet....</div>
</div>
<div class='item'>
  <div class='item-header'>Item 3 Header</div>
  <div class='item-content'>Lorem Ipsum Dolor Sit Amet....</div>
</div>
```

Das obige Beispiel nummeriert jedes "Element" auf der Seite und fügt die Nummer des Elements vor seinem Header hinzu (unter Verwendung der `content` von `.item-header` Element `:before` Pseudo). Eine Live-Demo dieses Codes ist hier verfügbar.

Implementierung der mehrstufigen Nummerierung mithilfe von CSS-Zählern

CSS

```
ul {
  list-style: none;
  counter-reset: list-item-number; /* self nesting counter as name is same for all levels */
}
li {
  counter-increment: list-item-number;
}
li:before {
  content: counters(list-item-number, ".") " "; /* usage of counters() function means value of
counters at all higher levels are combined before printing */
}
```

HTML

```
<ul>
  <li>Level 1
    <ul>
      <li>Level 1.1
        <ul>
          <li>Level 1.1.1</li>
        </ul>
      </li>
    </ul>
  </li>
  <li>Level 2
    <ul>
      <li>Level 2.1
        <ul>
          <li>Level 2.1.1</li>
          <li>Level 2.1.2</li>
        </ul>
      </li>
    </ul>
  </li>
  <li>Level 3</li>
</ul>
```

Das Obige ist ein Beispiel für die mehrstufige Nummerierung mit CSS-Zählern. Es nutzt das ***Selbstverschachtelungskonzept*** der Zähler. Selbstverschachtelung ist ein Konzept, bei dem ein Element, das bereits einen Zähler mit dem angegebenen Namen hat, einen anderen erstellen muss, es als untergeordnetes Element des vorhandenen Zählers erstellt. Hier erbt die zweite Ebene `ul` bereits den Zähler für die `list-item-number` von ihrem übergeordneten `list-item-number` , muss dann jedoch eine eigene `list-item-number` (für ihr untergeordnetes `li`) erstellen und erstellt so die `list-item-number[1]` (Zähler für zweite Ebene) und verschachtelt es unter der `list-item-number[0]` (Zähler für die erste Ebene). Dadurch wird die mehrstufige Nummerierung erreicht.

Der Ausgang wird gedruckt unter Verwendung der `counters()` Funktion anstelle der `counter()` Funktion , da die `counters()` Funktion ausgelegt ist , den Wert aller höheren Ebene Zähler (parent) vorangestellt , wenn der Ausgangsdruck.

Zähler online lesen: https://riptutorial.com/de/css/topic/2575/zahler

Kapitel 55: Zentrierung

Examples

CSS-Transformation verwenden

CSS-Transformationen basieren auf der Größe der Elemente. Wenn Sie also nicht wissen, wie groß oder breit Ihr Element ist, können Sie es absolut 50% von oben und links von einem relativen Container positionieren und um 50% nach links und nach oben verschieben vertikal und horizontal zentrieren.

Beachten Sie, dass das Element bei dieser Technik an einer nicht ganzzahligen Pixelgrenze gerendert werden könnte, wodurch es verschwommen wirkt. Siehe diese Antwort in SO für eine Problemumgehung.

HTML

```
<div class="container">
  <div class="element"></div>
</div>
```

CSS

```
.container {
  position: relative;
}

.element {
  position: absolute;
  top: 50%;
  left: 50%;
  transform: translate(-50%, -50%);
}
```

Beispiel in JSFiddle anzeigen

CROSS BROWSER-KOMPATIBILITÄT

Die Transformationseigenschaft benötigt Präfixe, die von älteren Browsern unterstützt werden. Präfixe werden für Chrome <= 35, Safari <= 8, Opera <= 22, Android Browser <= 4.4.4 und IE9 benötigt. CSS-Umwandlungen werden von IE8 und älteren Versionen nicht unterstützt.

Hier ist eine allgemeine Transformationsdeklaration für das vorherige Beispiel:

```
-webkit-transform: translate(-50%, -50%); /* Chrome, Safari, Opera, Android */
    -ms-transform: translate(-50%, -50%); /* IE 9 */
        transform: translate(-50%, -50%);
```

Weitere Informationen finden Sie unter canIuse .

MEHR INFORMATIONEN

- Das Element wird gemäß dem ersten nicht statischen übergeordneten Element positioniert (`position: relative` , `absolute` oder `fixed`). Erfahren Sie mehr in dieser Geige und in diesem Dokumentationsthema .
- Zum horizontalen Zentrieren verwenden Sie `left: 50%` und `transform: translateX(-50%)` . Das Gleiche gilt für die Nur-Vertikal-Zentrierung: Mitte mit `top: 50%` und `transform: translateY(-50%)` .
- Die Verwendung von nicht statischen Breiten- / Höhenelementen mit dieser Zentriermethode kann dazu führen, dass das zentrierte Element gestaucht erscheint. Dies geschieht meistens bei Elementen, die Text enthalten, und kann durch Hinzufügen von: `margin-right: -50%;` und `margin-bottom: -50%;` . Weitere Informationen finden Sie in dieser Geige .

Flexbox verwenden

HTML:

```
<div class="container">
  <img src="http://lorempixel.com/400/200" />
</div>
```

CSS:

```
html, body, .container {
  height: 100%;
}
.container {
  display: flex;
  justify-content: center; /* horizontal center */
}
img {
  align-self: center; /* vertical center */
}
```

Zeige Ergebnis

HTML:

```
<img src="http://lorempixel.com/400/200" />
```

CSS:

```
html, body {
  height: 100%;
}
body {
```

```
  display: flex;
  justify-content: center; /* horizontal center */
  align-items: center;     /* vertical center */
}
```

Zeige Ergebnis

Weitere Informationen zu Flexbox und zu den Stilen finden Sie unter Dynamische vertikale und horizontale Zentrierung in der Flexbox- Dokumentation.

Browser-Unterstützung

Flexbox wird von allen gängigen Browsern unterstützt, mit Ausnahme von IE-Versionen vor 10 .

Für einige aktuelle Browserversionen wie Safari 8 und IE10 sind Herstellerpräfixe erforderlich.

Für die schnelle Erstellung von Präfixen gibt es Autoprefixer , ein Tool von Drittanbietern.

Für ältere Browser (wie IE 8 & 9) ist ein Polyfill verfügbar .

Weitere Informationen zur Unterstützung von Flexbox-Browsern finden Sie in dieser Antwort .

Position verwenden: absolut

Arbeiten in alten Browsern (IE> = 8)

Automatische Ränder, gepaart mit Nullwerten für den `left` und `right` oder `top` und `bottom` Versatz, zentrieren absolut positionierte Elemente innerhalb des übergeordneten Elements.

Zeige Ergebnis

HTML

```
<div class="parent">
  <img class="center" src="http://lorempixel.com/400/200/" />
</div>
```

CSS

```
.parent {
  position: relative;
  height: 500px;
}

.center {
  position: absolute;
  margin: auto;
  top: 0;
  right: 0;
  bottom: 0;
  left: 0;
}
```

Elemente, die keine eigene implizite Breite und Höhe haben wie Bilder, benötigen diese Werte.

Andere Ressourcen: Absolute Zentrierung in CSS

Geisterelementtechnik (Michał Czernows Hack)

Diese Technik funktioniert auch, wenn die Abmessungen des Containers unbekannt sind.

Richten Sie ein "Ghost" -Element innerhalb des Containers mit einer Höhe von 100% ein. Verwenden Sie dann `vertical-align: middle` für das Element und das zu zentrierende Element.

CSS

```
/* This parent can be any width and height */
.block {
  text-align: center;

  /* May want to do this if there is risk the container may be narrower than the element
inside */
  white-space: nowrap;
}

/* The ghost element */
.block:before {
  content: '';
  display: inline-block;
  height: 100%;
  vertical-align: middle;

  /* There is a gap between ghost element and .centered,
  caused by space character rendered. Could be eliminated by
  nudging .centered (nudge distance depends on font family),
  or by zeroing font-size in .parent and resetting it back
  (probably to 1rem) in .centered. */
  margin-right: -0.25em;
}

/* The element to be centered, can also be of any width and height */
.centered {
  display: inline-block;
  vertical-align: middle;
  width: 300px;
  white-space: normal; /* Resetting inherited nowrap behavior */
}
```

HTML

```
<div class="block">
  <div class="centered"></div>
</div>
```

Verwenden Sie Textausrichtung

Die häufigste und einfachste Art der Zentrierung sind die Textzeilen in einem Element. CSS hat die Regel `text-align: center` für diesen Zweck:

HTML

```
<p>Lorem ipsum</p>
```

CSS

```
p {
    text-align: center;
}
```

Dies funktioniert nicht zum Zentrieren ganzer Blockelemente . `text-align` Kontrollen nur Ausrichtung von Inline - Inhalt wie Text in seinem übergeordneten Blockelement.

Weitere `text-align` im Abschnitt Typografie .

Zentrierung in Bezug auf einen anderen Artikel

Wir werden sehen, wie Inhalte basierend auf der Höhe eines nahen Elements zentriert werden.

Kompatibilität: IE8 +, alle anderen modernen Browser.

HTML

```
<div class="content">
  <div class="position-container">
    <div class="thumb">
      <img src="http://lorempixel.com/400/200/">
    </div>
    <div class="details">
      <p class="banner-title">text 1</p>
      <p class="banner-text">content content content content content content content content
content content content content content content</p>
      <button class="btn">button</button>
    </div>
  </div>
</div>
```

CSS

```
.content * {
  box-sizing: border-box;
}
.content .position-container {
  display: table;
}
.content .details {
  display: table-cell;
  vertical-align: middle;
  width: 33.333333%;
  padding: 30px;
  font-size: 17px;
  text-align: center;
}
.content .thumb {
```

```
  width: 100%;
}
.content .thumb img {
  width: 100%;
}
```

Link zu JSFiddle

Die Hauptpunkte sind die 3. `.thumb` , `.details` und `.position-container` :

- Der `.position-container` muss `.position-container` `display: table` .
- Die `.details` müssen die tatsächliche Breite der `width: ....` und `display: table-cell` , `vertical-align: middle` .
- Die `.thumb` muss `width: 100%` wenn Sie möchten, dass der gesamte verbleibende Speicherplatz belegt wird und die Breite der `.details` beeinflusst wird.
- Das Bild (wenn Sie ein Bild haben) in `.thumb` sollte eine `width: 100%` , ist aber nicht erforderlich, wenn Sie korrekte Proportionen haben.

Vertikale Ausrichtung mit 3 Codezeilen

Unterstützt von IE11 +

Zeige Ergebnis

Verwenden Sie diese 3 Linien, um praktisch alles vertikal auszurichten. Stellen Sie einfach sicher, dass das div / image, auf das Sie den Code anwenden, ein übergeordnetes Element mit einer Höhe hat.

CSS

```
div.vertical {
  position: relative;
  top: 50%;
  transform: translateY(-50%);
}
```

HTML

```
<div class="vertical">Vertical aligned text!</div>
```

Bild vertikal innerhalb von div ausrichten

HTML

```
<div class="wrap">
    <img src="http://lorempixel.com/400/200/" />
</div>
```

CSS

```
.wrap {
    height: 50px;/* max image height */
    width: 100px;
    border: 1px solid blue;
    text-align: center;
}
.wrap:before {
  content:"";
  display: inline-block;
  height: 100%;
  vertical-align: middle;
  width: 1px;
}

img {
    vertical-align: middle;
}
```

Horizontale und vertikale Zentrierung mit Tabellenlayout

Ein untergeordnetes Element kann leicht mit Hilfe der `table` zentriert werden.

HTML

```
<div class="wrapper">
    <div class="parent">
        <div class="child"></div>
    </div>
</div>
```

CSS

```
.wrapper {
    display: table;
    vertical-align: center;
    width: 200px;
    height: 200px;
    background-color: #9e9e9e;
}
.parent {
    display: table-cell;
    vertical-align: middle;
    text-align: center;
}
.child {
    display: inline-block;
    vertical-align: middle;
    text-align: center;
    width: 100px;
    height: 100px;
    background-color: teal;
}
```

Calc () verwenden

Die Funktion calc () ist Teil einer neuen Syntax in CSS3, in der Sie mathematisch berechnen können, welche Größe / Position Ihr Element mit verschiedenen Werten wie Pixeln, Prozentwerten usw. belegt `calc(100% - 80px)` immer auf den Abstand zwischen zwei Werten `calc(100% - 80px)` .

CSS

```
.center {
    position: absolute;
    height: 50px;
    width: 50px;
    background: red;
    top: calc(50% - 50px / 2); /* height divided by 2*/
    left: calc(50% - 50px / 2); /* width divided by 2*/
}
```

HTML

```
<div class="center"></div>
```

Dynamische Höhenelemente vertikal ausrichten

Die intuitive Anwendung von css führt nicht zu den gewünschten Ergebnissen, weil

- `vertical-align:middle` ***ist nicht*** auf Elemente auf Blockebene anwendbar
- `margin-top:auto` und `margin-bottom:auto` verwendete Werte würden als ***Null*** berechnet werden
- `margin-top:-50%` prozentuale Randwerte werden relativ zur ***Breite*** des enthaltenden Blocks berechnet

Für die breiteste Browserunterstützung eine Problemumgehung mit Hilfselementen:

HTML

```
<div class="vcenter--container">
  <div class="vcenter--helper">
    <div class="vcenter--content">
      <!--stuff-->
    </div>
  </div>
</div>
```

CSS

```
.vcenter--container {
  display: table;
  height: 100%;
  position: absolute;
  overflow: hidden;
  width: 100%;
}
.vcenter--helper {
  display: table-cell;
  vertical-align: middle;
}
```

```
.vcenter--content {
  margin: 0 auto;
  width: 200px;
}
```

jsfiddle aus der ursprünglichen Frage . Dieser Ansatz

- arbeitet mit dynamischen Höhenelementen
- respektiert den Fluss der Inhalte
- wird von älteren Browsern unterstützt

Zeilenhöhe verwenden

Sie können die Zeilenhöhe auch verwenden `line-height` um eine einzelne Textzeile in einem Container vertikal zu zentrieren:

CSS

```
div {
    height: 200px;
    line-height: 200px;
}
```

Das ist ziemlich hässlich, kann aber in einem `<input />` -Element nützlich sein. Die `line-height` -Eigenschaft funktioniert nur , wenn der Text zentriert eine einzige Zeile erstreckt werden. Wenn der Text in mehrere Zeilen umbrochen wird, wird die resultierende Ausgabe nicht zentriert.

Vertikal und horizontal zentrieren, ohne sich um Höhe oder Breite zu kümmern

Mit der folgenden Technik können Sie einem HTML-Element Inhalte hinzufügen und sowohl horizontal als auch vertikal **zentrieren, ohne sich um die Höhe oder Breite zu kümmern** .

Der äußere Behälter

- sollte `display: table;`

Der innere Behälter

- sollte `display: table-cell;`
- sollte `vertical-align: middle;`
- sollte `text-align: center;`

Das Inhaltsfeld

- sollte `display: inline-block;`
- sollte die horizontale Textausrichtung z. `text-align: left;` oder `text-align: right;` , sofern

Sie nicht möchten, dass der Text zentriert wird

Demo

HTML

```
<div class="outer-container">
   <div class="inner-container">
     <div class="centered-content">
        You can put anything here!
     </div>
   </div>
</div>
```

CSS

```
body {
    margin : 0;
}

.outer-container {
    position : absolute;
    display: table;
    width: 100%; /* This could be ANY width */
    height: 100%; /* This could be ANY height */
    background: #ccc;
}

.inner-container {
    display: table-cell;
    vertical-align: middle;
    text-align: center;
}

.centered-content {
    display: inline-block;
    text-align: left;
    background: #fff;
    padding: 20px;
    border: 1px solid #000;
}
```

Siehe auch **diese Geige** !

Zentrierung mit fester Größe

Wenn die Größe Ihres Inhalts festgelegt ist, können Sie die absolute Positionierung auf 50% setzen, wobei der `margin` die Hälfte der Breite und Höhe Ihres Inhalts verringert:

HTML

```
<div class="center">
    Center vertically and horizontally
</div>
```

CSS

```
.center {
    position: absolute;
    background: #ccc;

    left: 50%;
    width: 150px;
    margin-left: -75px;  /* width * -0.5 */

    top: 50%;
    height: 200px;
    margin-top: -100px;  /* height * -0.5 */
}
```

Horizontale Zentrierung nur mit fester Breite

Sie können das Element horizontal zentrieren, auch wenn Sie die Höhe des Inhalts nicht kennen:

HTML

```
<div class="center">
    Center only horizontally
</div>
```

CSS

```
.center {
    position: absolute;
    background: #ccc;

    left: 50%;
    width: 150px;
    margin-left: -75px;  /* width * -0.5 */
}
```

Vertikale Zentrierung mit fester Höhe

Sie können das Element vertikal zentrieren, wenn Sie die Höhe des Elements kennen:

HTML

```
<div class="center">
    Center only vertically
</div>
```

CSS

```
.center {
    position: absolute;
    background: #ccc;
```

```
    top: 50%;
    height: 200px;
    margin-top: -100px;  /* width * -0.5 */
}
```

Mit Marge: 0 auto;

Objekte können mit dem `margin: 0 auto;` zentriert werden `margin: 0 auto;` wenn sie Blockelemente sind und eine definierte Breite haben.

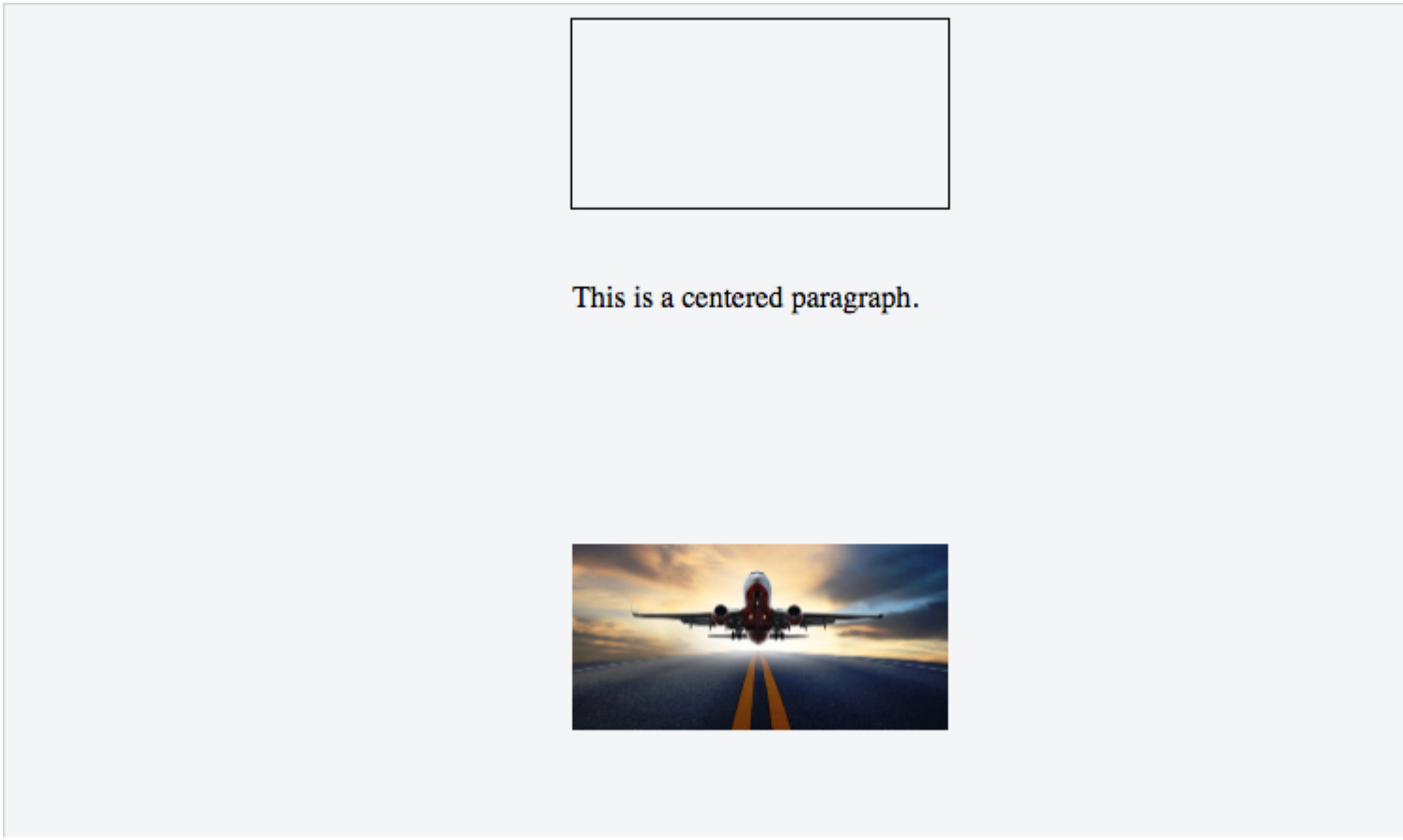
HTML

```
<div class="containerDiv">
    <div id="centeredDiv"></div>
</div>

<div class="containerDiv">
    <p id="centeredParagraph">This is a centered paragraph.</p>
</div>

<div class="containerDiv">
    <img id="centeredImage" src="https://i.kinja-img.com/gawker-media/image/upload/s--
c7Q9b4Eh--/c_scale,fl_progressive,q_80,w_800/qqyvc3bkpyl3mfhr8all.jpg" />
</div>
```

CSS

```
.containerDiv {
    width: 100%;
    height: 100px;
    padding-bottom: 40px;
}

#centeredDiv {
    margin: 0 auto;
    width: 200px;
    height: 100px;
    border: 1px solid #000;
}

#centeredParagraph {
    width: 200px;
    margin: 0 auto;
}

#centeredImage {
    display: block;
    width: 200px;
    margin: 0 auto;
}
```

Ergebnis:



JSFiddle-Beispiel: Objekte mit Rand zentrieren: 0 auto;

Zentrierung online lesen: https://riptutorial.com/de/css/topic/299/zentrierung

Kapitel 56: Zersplitterung

Syntax

- Seitenumbruch: auto | immer | vermeiden | links | richtig | initial | erben;
- Seitenumbruch: auto | immer | vermeiden | links | richtig | initial | erben;
- Seitenumbruch innen: auto | vermeiden | initial | erben;

Parameter

Wert	Beschreibung
Auto	Standard. Automatische Seitenumbrüche
immer	Fügen Sie immer einen Seitenumbruch ein
vermeiden	Seitenumbruch vermeiden (wenn möglich)
links	Fügen Sie Seitenumbrüche ein, sodass die nächste Seite als linke Seite formatiert wird
Recht	Fügen Sie Seitenumbrüche ein, damit die nächste Seite als rechte Seite formatiert wird
Initiale	Setzt diese Eigenschaft auf ihren Standardwert.
erben	Übernimmt diese Eigenschaft von ihrem übergeordneten Element.

Bemerkungen

Es gibt keine Seitenumbrücheigenschaft in CSS. Nur die 3 Eigenschaften (**Seitenwechsel** , **Seitenwechsel** , **Seitenwechsel innerhalb**).

Verwandte: `orphans` , `widows` .

Examples

Medien drucken Seitenumbruch

```
@media print {
  p {
    page-break-inside: avoid;
  }
  h1 {
    page-break-before: always;
```

```
  }
  h2 {
    page-break-after: avoid;
  }
}
```

Dieser Code macht 3 Dinge:

- Es verhindert einen Seitenumbruch innerhalb von p-Tags, dh ein Absatz wird niemals auf zwei Seiten gebrochen, wenn möglich.
- es erzwingt einen Seitenumbruch in allen h1-Überschriften, was bedeutet, dass vor jedem Auftreten von h1 ein Seitenumbruch erfolgt.
- es verhindert Seitenumbrüche direkt nach h2

Zersplitterung online lesen: https://riptutorial.com/de/css/topic/4316/zersplitterung

Credits

S. No	Kapitel	Contributors
1	Erste Schritte mit CSS	adamboro, animuson, Ashwin Ramaswami, awe, Boysenb3rry , Chris, Community, csx.cc, darrylyeo, FelipeAls, Gabriel R., Garconis, Gerardas, GoatsWearHats, G-Wiz, Harish Gyanani, Heri Hehe Setiawan, J Atkin, Jmh2013, joe_young, Jose Gomez, Just a student, Lambda Ninja, Marjorie Pickard, Nathan Arthur, patelarpan, RamenChef, Rocket Risa, Saroj Sasmal, ScientiaEtVeritas, selvassn, Sverri M. Olsen, Teo Dragovic, Todd, TylerH, Vivek Ghaisas, Xinyang Li, ZaneDickens
2	2D-Transformationen	Charlie H, Christiaan Maks, Harry, Hors Sujet, John Slegers, Luke Taylor, Madalina Taina, mnoronha, PaMaDo, Praveen Kumar, RamenChef, web-tiki, zer00ne
3	3D-Transformationen	Harry, Luka Kerr, Madalina Taina, mnoronha, Mr. Meeseeks, Nhan, RamenChef, web-tiki
4	Animationen	Aeolingamenfel, apaul, Dex Star, Jasmin Solanki, Nathan Arthur, RamenChef, Richard Hamilton, TylerH
5	Bemerkungen	animuson, bdkopen, coderfin, Madalina Taina, Nick
6	Benutzerdefinierte Eigenschaften (Variablen)	animuson, Brett DeWoody, Community, Daniel Käfer, Muthu Kumaran, Obsidian, RamenChef, RedRiderX, TylerH
7	Beschneiden und Maskieren	Andre Lopes, Harry, RamenChef, ScientiaEtVeritas, web-tiki
8	Block-Formatierungskontexte	Madalina Taina, Milan Laslop, RamenChef
9	Box Schatten	Hristo, Madalina Taina, RamenChef
10	Browserstile normalisieren	andre mcgruder, Confused One, Grant Palin, MMachinegun, mnoronha, RamenChef, SeinopSys
11	Browserunterstützung und Präfixe	Andrew, animuson, Braiam, Nhan, Obsidian, RamenChef, ScientiaEtVeritas, Shaggy, TylerH
12	CSS Image Sprites	Elegant.Scripting, Jmh2013, RamenChef, Ted Goas, Ulrich Schwarz

13	CSS-Entwurfsmuster	John Slegers
14	CSS-Objektmodell (CSSOM)	Alohci, animuson, feeela, Paul Sweatte, RamenChef, rishabh dev
15	Cursor-Styling	cone56, Madalina Taina, ScientiaEtVeritas, Squazz
16	Das Boxmodell	Arjan Einbu, Ben Rhys-Lewis, Benolot, BiscuitBaker, FelipeAls, James Donnelly, Lambda Ninja, Nathan Arthur, Ortomala Lokni, RamenChef, ScientiaEtVeritas, Sergej, V-Kopio
17	Eigenschaft filtern	Jeffery Tang, Nathan, RamenChef
18	Einzelelementformen	andreas, animuson, Brett DeWoody, Chiller, J F, Nick, RamenChef, ScientiaEtVeritas, TylerH
19	Erbe	Chris, mnoronha, RamenChef
20	Farben	andreas, animuson, Arjan Einbu, Brett DeWoody, Community, cuervoo, darrylyeo, designcise, H. Pauwelyn, Jasmin Solanki, John Slegers, Kuhan, Marc, Michael Moriarty, Miro, Nathan Arthur, niyasc, RamenChef, Richard Hamilton, ScientiaEtVeritas, SeinopSys, Stewartside, user007, Wolfgang, X-27
21	Flexibles Boxenlayout (Flexbox)	Ahmad Alfy, Asim K T, FelipeAls, fzzylogic, James Donnelly, Jef, Lambda Ninja, Marc-Antoine Leclerc, Nathan Arthur, Nhan, Ori Marash, RamenChef, Randy, Squazz, takeradi, Ted Goas, Timothy Morris, TylerH
22	Formen für Schwimmer	animuson, Harry, RamenChef, ScientiaEtVeritas
23	Funktionen	animuson, Brett DeWoody, cone56, dodopok, H. Pauwelyn, haim770, jaredsk, khawarPK, Kobi, RamenChef, SeinopSys, TheGenie OfTruth, TylerH
24	Funktionsabfragen	Andrew Myers, RamenChef
25	Gitter	Chris Spittles, FelipeAls, Jonathan Zúñiga, Mike McCaughan, Nhan, Praveen Kumar, RamenChef, Sebastian Zartner
26	Hintergründe	4444, Ahmad Alfy, animuson, Asim K T, Ben Rhys-Lewis, Boris, CalvT, cdm, Charlie H, CocoaBean, Dan Devine, Dan Eastwell, Daniel G. Blázquez, Daniel Stradowski, Darthstroke, designcise, Devid Farinelli, dippas, fcalderan, FelipeAls, Goose, Horst Jahns, Hynes, Jack, Jacob Gray, James Taylor, John Slegers, Jon Chan, Jonathan Zúñiga, Kevin Montrose, Louis St-Amour, Madalina Taina, Maximillian Laumeister, Michael Moriarty, Mr. Alien, mtb, Nate, Nathan Arthur, Nhan,

		Persijn, Praveen Kumar, RamenChef, Richard Hamilton, ScientiaEtVeritas, Sergey Denisov, Shaggy, Sourav Ghosh, Stewartside, Stratboy, think123, Timothy, Trevor Clarke, TylerH, Zac, Zeta, Zze
27	Inline-Block-Layout	Marten Koetsier, RamenChef
28	Internet Explorer-Hacks	Aeolingamenfel, animuson, Elizaveta Revyakina, feeela, John Slegers, LiLacTac, Praveen Kumar, RamenChef, ScientiaEtVeritas, Timothy Miller, TylerH
29	Kaskadierung und Spezifität	amflare, Arjan Einbu, brandaemon, DarkAjax, dippas, dmkerr, geeksal, Grant Palin, G-Wiz, James Donnelly, jehna1, John Slegers, kingcobra1986, Matas Vaitkevicius, Nathan Arthur, Nhan, Ortomala Lokni, RamenChef, ScientiaEtVeritas, Sunnyok, Ze Rubeus
30	Längeneinheiten	4dgaurav, A B, animuson, Epodax, geeksal, J F, Marc, Milche Patern, Ortomala Lokni, RamenChef, Richard Hamilton, rmondesilva, Robert Koritnik, Robotnicka, ScientiaEtVeritas, StefanBob, Stewartside, Thomas Altmann, Toby, user2622348 , vladdobra, Zakaria Acharki, zer00ne
31	Layoutsteuerung	Arjun Iv, Chathuranga Jayanath, Chris, Jmh2013, Kevin Katzke, Kurtis Beavers, Madalina Taina, mnoronha, Niek Brouwer, RamenChef, Sander Koedood, ScientiaEtVeritas, SeinopSys
32	Listenstile	animuson, Madalina Taina, Marten Koetsier, RamenChef, Ted Goas
33	Medien-Anfragen	amflare, Chathuranga Jayanath, darrylyeo, Demeter Dimitri, dodopok, James Donnelly, Jmh2013, joe_young, joejoe31b, John Slegers, Matas Vaitkevicius, Mattia Astorino, Maximillian Laumeister, Nathan Arthur, Praveen Kumar, RamenChef, ScientiaEtVeritas, srikarg, Teo Dragovic, Viktor
34	Mehrere Spalten	bipon, Sebastian Zartner
35	Objektanpassung und Platzierung	4444, Miles, RamenChef
36	Opazität	Andrew, animuson, Hillel Tech, Madalina Taina, RamenChef
37	Performance	mnoronha, RamenChef, TrungDQ
38	Polsterung	Andy G, CalvT, Felix A J, Madalina Taina, Mehdi Dehghani, Nathan, Paul Sweatte, pixelbandito, RamenChef, StefanBob, Will DiFruscio

39	Positionierung	Abhishek Singh, Alohci, animuson, Blunderfest, CalvT, Cassidy Williams, Chetan Joshi, GingerPlusPlus, Jacob Gray, Lambda Ninja, Madalina Taina, Matthew Beckman, Mattia Astorino, Rahul Nanwani, RamenChef, Sourav Ghosh, Stephen Leppik, Theodore K., TylerH
40	Pseudo-Elemente	4dgaurav, ankit, Chris, Community, dippas, dmnsgn, geek1011, geeksal, Gofilord, kevin vd, Marcatectura, Milche Patern, Nathan, Pat, Praveen Kumar, RamenChef, ScientiaEtVeritas, Shaggy, Stewartside, Sunnyok
41	Rand	andreas, Cassidy Williams, doctorsherlock, FelipeAls, Gnietschow, Harry, jaredsk, Madalina Taina, Nobal Mohan, RamenChef, ScientiaEtVeritas, Trevor Clarke
42	Ränder	Arjan Einbu, cdm, Chris Spittles, Community, J F, Madalina Taina, Mr_Green, Nathan Arthur, RamenChef, rejnev, Sun Qingyao, Sunnyok, Tot Zam, Trevor Clarke
43	Säulen	Brett DeWoody, Madalina Taina, RamenChef
44	Schwimmt	demonofthemist, FelipeAls, Madalina Taina, Nathan Arthur, RamenChef, vishak
45	Selektoren	75th Trombone, A.J, Aaron, abaracedo, Ahmad Alfy, Alex Filatov, amflare, Anil, animuson, Araknid, Arjan Einbu, Ashwin Ramaswami, BoltClock, Cerbrus, Charlie H, Chris, Chris Nager, Clinton Yeboah, Community, CPHPython, darrylyeo, Dave Everitt, David Fullerton, Demeter Dimitri, designcise, Devid Farinelli, Devon Bernard, Dinidu, dippas, Erenor Paz, Felix Edelmann, Felix Schütz, flyingfisch, Forty, fracz, Frits, gandreadis, geeksal, George Bailey, George Grigorita, H. Pauwelyn, HansCz, henry, Hugo Buff, Hynes, J Atkin, J F, Jacob Gray, James Donnelly, James Taylor, Jasha, jehna1, Jmh2013, joejoe31b, Joël Bonet Rodríguez, John Slegers, Kurtis Beavers, Madalina Taina, Marc, Mark Perera, Matas Vaitkevicius, Matsemann, Michael_B, Milan Laslop, Naeem Shaikh, Nathan Arthur, Nick, Ortomala Lokni, Persijn, Praveen Kumar, RamenChef, rdans, Richard Hamilton, Rion Williams, Robert Koritnik, RockPaperLizard, RoToRa, Sbats, ScientiaEtVeritas, Shaggy, Siavas, Stewartside, sudo bangbang, Sumner Evans, Sunnyok, ThatWeirdo, theB, Thomas Gerot, TylerH, xpy, Yury Fedorov, Zac, Zaffy, Zaz, Ze Rubeus, Zze
46	Stapelkontext	Nemanja Trifunovic, RamenChef
47	Struktur und Formatierung einer	Alon Eitan, darrylyeo, Marjorie Pickard

	CSS-Regel	
48	Tabellen	Casper Spruit, Chris, FelipeAls, JHS, Madalina Taina
49	Typografie	Alex Morales, Alohci, andreas, Arjan Einbu, ChaoticTwist, Evgeny, Felix A J, Goulven, Hynes, insertusernamehere, James Donnelly, joe_young, Jon Chan, Madalina Taina, Michael Moriarty, Nathan Arthur, Nhan, Praveen Kumar, RamenChef, Richard Hamilton, rmondesilva, Ryan, Sourav Ghosh, Suhaib Janjua, Ted Goas, Toby, ToniB, Trevor Clarke, user2622348, Vlusion, Volker E.
50	Übergänge	Christiaan Maks, dippas, Harry, Praveen Kumar, RamenChef, Richard Hamilton, ScientiaEtVeritas, Sergey Denisov, web-tiki
51	Überlauf	Andrew, bdkopen, jgh, Madalina Taina, Miles, Qaz, RamenChef, ScientiaEtVeritas, Ted Goas, Zac
52	Umrisse	Arif, Casey, Chathuranga Jayanath, Daniel Nugent, FelipeAls, Madalina Taina, RamenChef, ScientiaEtVeritas
53	Vertikale Zentrierung	animuson, AVAVT, bocanegra, Chiller, Chris, jaredsk, leo_ap, mmativ, patelarpan, Phil, Praveen Kumar, RamenChef
54	Zähler	Harry, RamenChef
55	Zentrierung	abaracedo, Alex Morales, Alohci, andreas, animuson, apaul, Christiaan Maks, Daniel Käfer, Devid Farinelli, Diego V, dippas , Eliran Malka, Emanuele Parisio, Euan Williams, F. Müller, Farzad YZ, Felix A J, geek1011, insertusernamehere, JedaiCoder, jehna1, JHS, John Slegers, Jonathan Argentiero, Kilian Stinson, Kyle Ratliff, Lambda Ninja, Lucia Bentivoglio, Luke Taylor, Madalina Taina, maho, Maxouhell, Michael_B, Mifeet, Mod Proxy, Mohammad Usman, Nathan Arthur, Nhan, o.v., Ortomala Lokni, paaacman, Paul Kozlovitch, Praveen Kumar, Sandeep Tuniki, ScientiaEtVeritas, Sergej, Siavas, smonff, Someone, Stewartside, Sunnyok, Taylor, TylerH, web-tiki, Ze Rubeus, zeel
56	Zersplitterung	animuson, dodopok, Madalina Taina, Milan Laslop, RamenChef