

# Responsive Web Design (RWD)

Einführung in  
Web Engineering  
Teil 3  
*Manfred Kaul*



Quelle: [https://de.wikipedia.org/wiki/Responsive\\_Webdesign](https://de.wikipedia.org/wiki/Responsive_Webdesign)

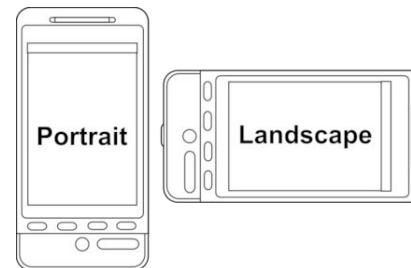


Quelle: [https://en.wikipedia.org/wiki/Cascading\\_Style\\_Sheets](https://en.wikipedia.org/wiki/Cascading_Style_Sheets)

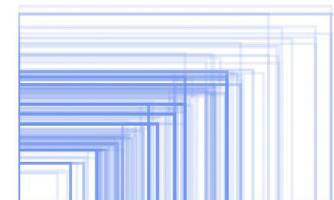
# Was ist der Unterschied zwischen PDF und HTML?

# Definition Responsive Webdesign (RWD)

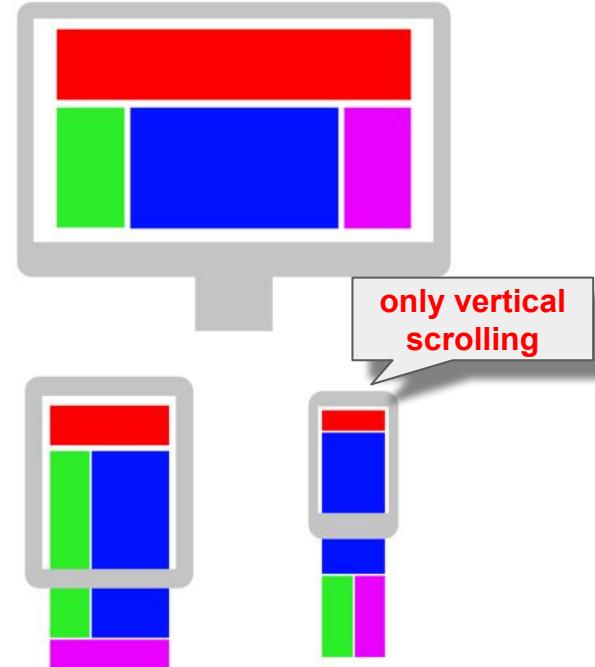
- Unter einer responsiven Gestaltung einer Webseite versteht man, dass sich das Layout der Seite an das Ausgabemedium selbstständig (möglichst nahtlos, fluide) anpasst.
- Der grafische Aufbau einer „responsiven“ Webseite folgt den Anforderungen des jeweiligen Gerätes dynamisch.



[https://en.wikipedia.org/wiki/Page\\_orientation](https://en.wikipedia.org/wiki/Page_orientation)



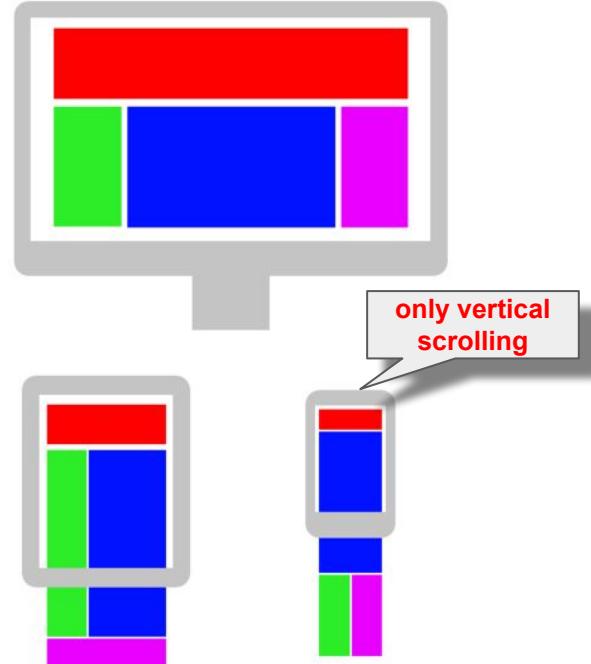
<https://opensignal.com/reports/fragmentation.php>



[https://en.wikipedia.org/wiki/Responsive\\_web\\_design](https://en.wikipedia.org/wiki/Responsive_web_design)

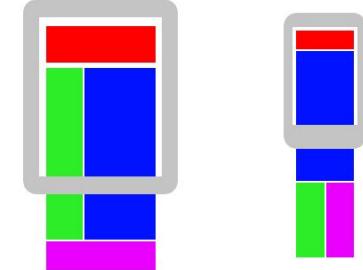
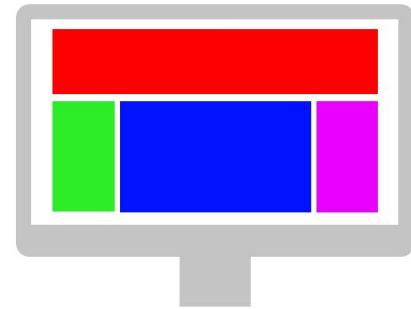
# Definitionen: responsiv - fluide - mobil - adaptiv - liquid

1. Responsive Web Design (RWD)  
**fluide** Anpassung an die Geräteeigenschaften
2. Mobile Webseite - optimiert für Mobilgeräte
3. Adaptive Webseite  
feste Anzahl von **Breakpoints**, wann Layout umgeschaltet wird - nicht fluide
4. Liquide Webseite  
zwar fluide Anpassung, aber immer mit gleicher Anordnung - keine Umbrüche, keine Übergänge, keine Breakpoints



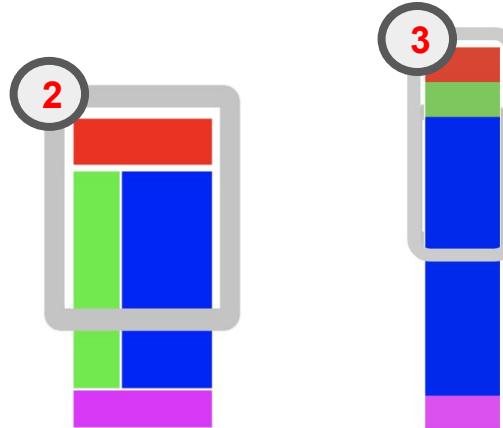
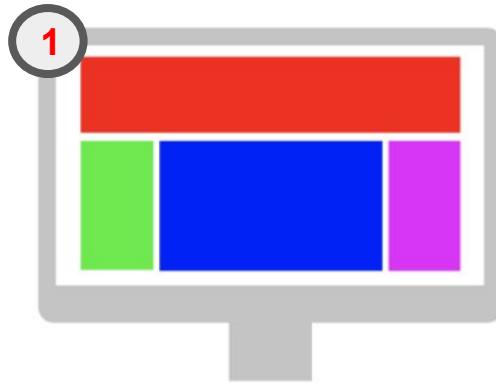
# Zwei Methoden für Responsive Design

1. Desktop First
2. Mobile First



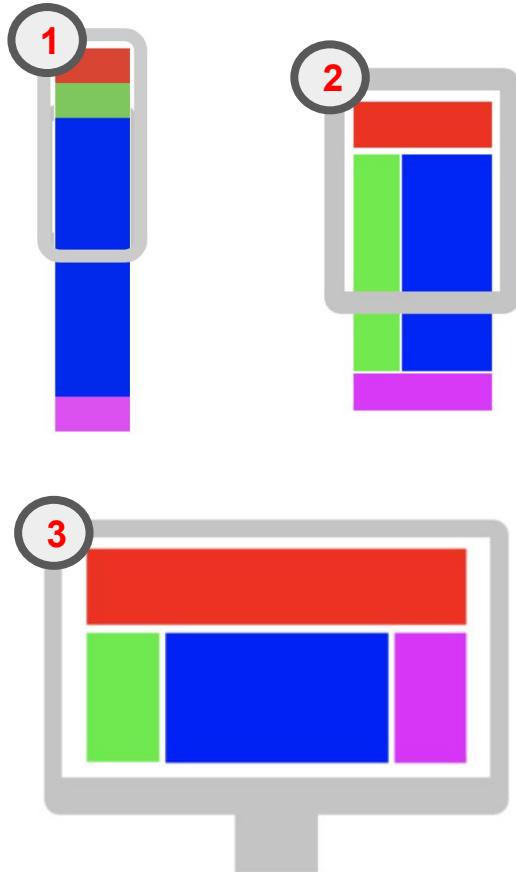
# Desktop First

- Erstellen Sie die Seite,
- schieben Sie Ihr Browserfenster zusammen
- Fügen Sie Layout-Regeln für kleinere Viewports hinzu



# Mobile First

- ziehen Sie das Browserfenster eng zusammen
- erstellen die Seite.
- In einem zweiten Schritt ziehen Sie das Fenster auseinander und fügen Layout-Regeln für größere Viewports hinzu



# Technische Realisierung von RWD mit CSS

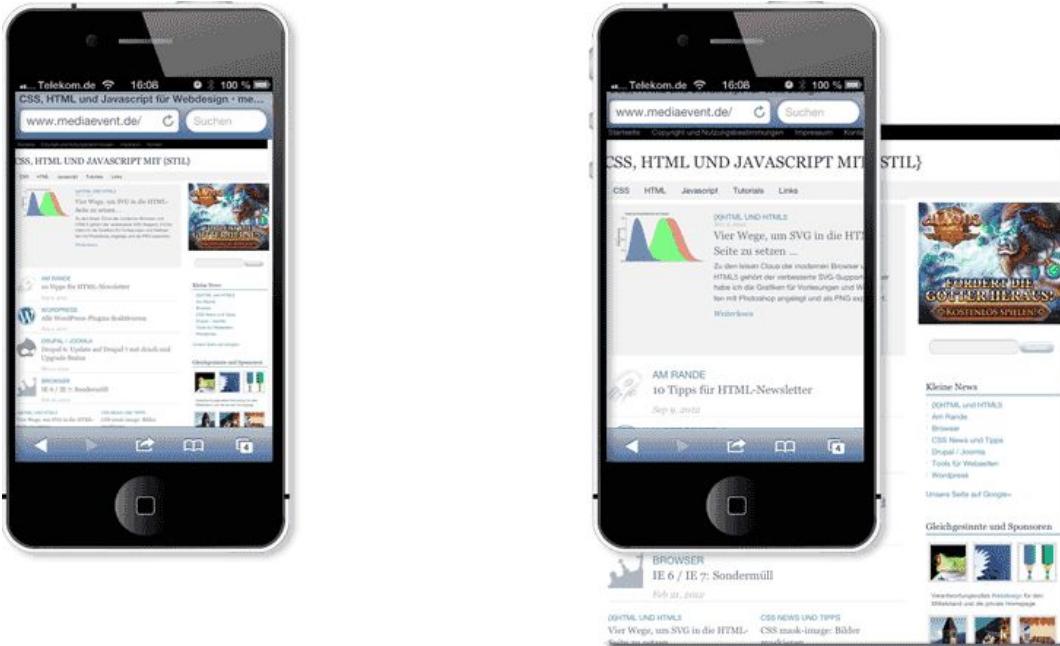
1. <meta>-Tag viewport
2. Media Queries
3. View Port Units (% , vw, vh)
4. **calc()** Function Expressions
5. CSS Custom Properties
6. Flex Layout
7. Grid Layout
8. Flexible Images
9. Flexible Videos
10. Responsive Font Size



Quelle: [https://de.wikipedia.org/wiki/Responsive\\_Webdesign](https://de.wikipedia.org/wiki/Responsive_Webdesign)

# 1. <meta>-Tag viewport

```
<meta name="viewport" content="width=device-width, initial-scale=1.0">
```



<https://www.mediaevent.de/css/media-queries.html>

# 1. <meta>-Tag viewport

```
<meta name="viewport" content="width=device-width, initial-scale=1.0">
```

Mobile devices come in all sizes, with screens of differing device pixel ratios. The mobile browser's viewport is the area of the window in which web content can be seen, which is not necessarily the same size as the rendered page. Mobile browsers render pages in a virtual window or viewport, generally at **960px**, which is usually wider than the screen, and then shrink the rendered result down so it can all be seen at once. Users can then pan and zoom to see different areas of the page. For example, if a mobile screen has a width of 320px, a website might be rendered with a virtual viewport of 960px, and then it will be shrunk down to fit into the 320px space, which, depending on the design, is illegible for many if not everyone. To tell a mobile browser to use the viewport width instead of the default 960px as the width of the screen, developers can include a viewport meta tag, like the following:

```
<meta name="viewport" content="width=device-width">
```

The `width` property controls the size of the viewport. It should preferably be set to `device-width`, which is the width of the screen in CSS pixels at a scale of 100%. There are other properties, including `maximum-scale`, `minimum-scale`, and `user-scalable`, which control whether users can zoom the page in or out, but the default values are the best for accessibility and user experience, so these can be omitted.

[https://developer.mozilla.org/en-US/docs/Web/CSS/Viewport\\_concepts](https://developer.mozilla.org/en-US/docs/Web/CSS/Viewport_concepts)

## 2. Media Queries

```
@media print {  
  * {  
    background-color: white;  
    font-family: "times new roman", times, serif;  
    text-align: justify;  
  }  
}  
  
@media screen and (max-width: 220px){ }  
@media (min-width: 30em) and (orientation: landscape) { }  
@media screen and (min-width: 30em) and (orientation: landscape) { }  
@media (min-height: 680px), screen and (orientation: portrait) { }  
@media (min-width: 30em) and (max-width: 50em) { }  
@media (not:hover)) { }
```

Druck mit Serifen, Bildschirm ohne

- 1 Medientypen
- 2 Medienmerkmale
  - 2.1 width
  - 2.2 height
  - 2.3 device-width
  - 2.4 device-height
  - 2.5 device-pixel-ratio
  - 2.6 orientation
  - 2.7 aspect-ratio
  - 2.8 device-aspect-ratio
  - 2.9 color
  - 2.10 color-index
  - 2.11 monochrome
  - 2.12 light-level
  - 2.13 pointer
  - 2.14 resolution
  - 2.15 scan
  - 2.16 grid

# Medientypen: Media Types in Media Queries

As such, the following **media types** are defined for use in **media queries**:

## all

Matches all devices.

## print

Matches printers, and devices intended to reproduce a printed display, such as a web browser showing a document in “Print Preview”.

## screen

Matches all devices that aren’t matched by **print** or **speech**.

## speech

Matches devices that similar devices that “read out” a page.

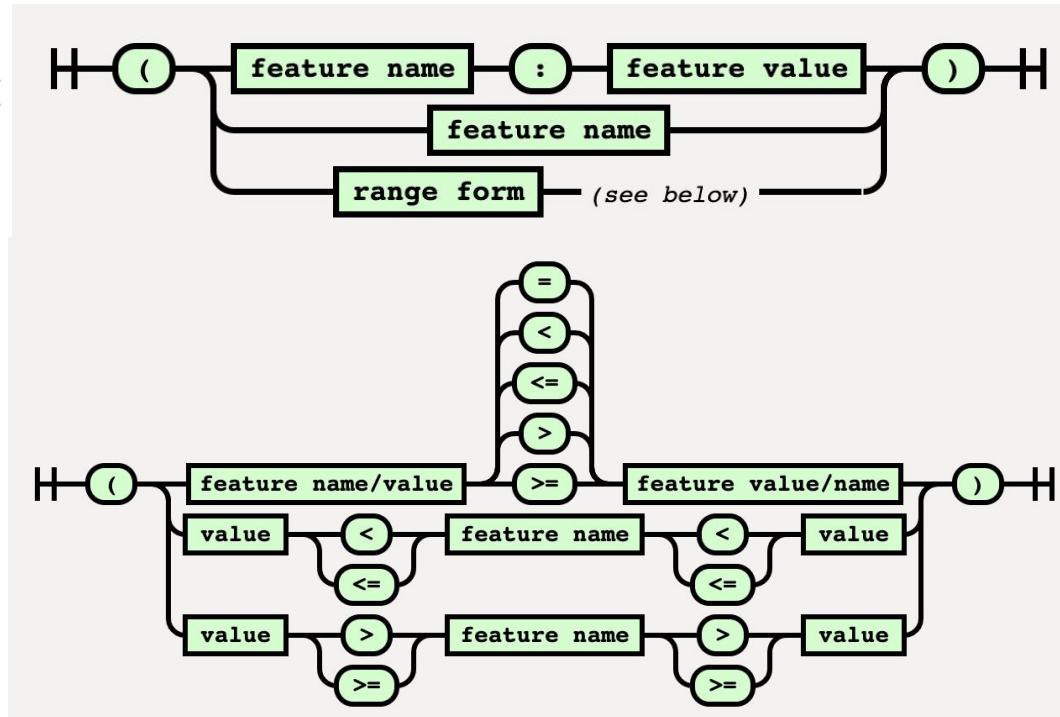
```
@media print {  
    body {  
        color: black;  
        background-color: white;  
    }  
    h1 {  
        font-size: 14pt;  
    }  
    .navigation {  
        display: none;  
    }  
}
```

Ausschalten von  
Teilbereichen

<https://drafts.csswg.org/mediaqueries-4/#media-types>

# Medienmerkmale: Media Features in Media Queries

```
@media (max-width: 220px){ }  
@media (min-width: 30em) and (orientation: landscape){ }  
@media (min-height: 680px) and (orientation: portrait){ }  
@media (min-width: 30em) and (max-width: 50em){ }  
@media (width: 600px){ }  
@media (aspect-ratio: 16/9){ }  
@media (orientation: portrait){ }  
@media (min-color: 3){ } /* 3 Bits per Pixel */  
@media (color-index: 16){ } /* 16 colors only */  
@media (monochrome: 1){ } /* 1 Bit Graustufen */  
@media (pointer: fine){ }  
@media (min-resolution: 200dpi){ }
```



# Medienmerkmale in Media Queries

Feature	Value	Min/Max	Description
color	integer	yes	number of bits per color component
color-index	integer	yes	number of entries in the color lookup table
device-aspect-ratio	integer/integer	yes	aspect ratio
device-height	length	yes	height of the output device
device-width	length	yes	width of the output device
grid	integer	no	true for a grid-based device
height	length	yes	height of the rendering surface
monochrome	integer	yes	number of bits per pixel in a monochrome frame buffer
resolution	resolution ("dpi" or "dpcm")	yes	resolution
scan	"progressive" or "interlaced"	no	scanning process of "tv" media types
width	length	yes	width of the rendering surface

```
<head>
  <link rel="stylesheet" type="text/css" href="css/all.css" />
  <link rel="stylesheet" type="text/css" media="all and (max-device-width: 480px)"
        href="css/smartphones.css" />
</head>
```

[https://de.wikipedia.org/wiki/Responsive\\_Webdesign](https://de.wikipedia.org/wiki/Responsive_Webdesign)

# Beispiel

## Media Query mit Medienmerkmal (Eigenschaftsabfrage)

```
#inhalt {  
    width: 800px;  
}  
  
@media screen and (max-width: 1024px) {  
    #inhalt {  
        width: 600px;  
    }  
  
    aside {  
        display: none;  
    }  
}
```

Mit Media Queries können die Eigenschaften des aktuellen Gerätes direkt abgefragt werden. Verfügbare Geräteeigenschaften sind zum Beispiel:

- Breite und Höhe des Browserfensters
- Breite und Höhe des Gerätes
- Orientierung (Quer- oder Hochformat)
- Bildschirmauflösung

Ausschalten von  
Teilbereichen

# Logische Kombinatoren für Media Queries

- and
- or
- not

```
<media-query> = <media-condition>
    | [ not | only ]? <media-type> [ and <media-condition-without-or> ]?
<media-type> = <ident>

<media-condition> = <media-not> | <media-in-parens> [ <media-and>* | <media-or>* ]
<media-condition-without-or> = <media-not> | <media-in-parens> <media-and>*
<media-not> = not <media-in-parens>
<media-and> = and <media-in-parens>
<media-or> = or <media-in-parens>
<media-in-parens> = ( <media-condition> ) | <media-feature> | <general-enclosed>

<media-feature> = ( [ <mf-plain> | <mf-boolean> | <mf-range> ] )
<mf-plain> = <mf-name> : <mf-value>
<mf-boolean> = <mf-name>
<mf-range> = <mf-name> <mf-comparison> <mf-value>
    | <mf-value> <mf-comparison> <mf-name>
    | <mf-value> <mf-lt> <mf-name> <mf-lt> <mf-value>
    | <mf-value> <mf-gt> <mf-name> <mf-gt> <mf-value>
<mf-name> = <ident>
<mf-value> = <number> | <dimension> | <ident> | <ratio>
<mf-lt> = '<' '='?
<mf-gt> = '>' '='?
<mf-eq> = '='
<mf-comparison> = <mf-lt> | <mf-gt> | <mf-eq>

<general-enclosed> = [ <function-token> <any-value> ) ] | ( <ident> <any-value> )
```

<https://drafts.csswg.org/mediaqueries-4/#mq-syntax>

# Beispiel Media Query for RWD mit float

```
/* Style the topnav links */  
.topnav a {  
    float: left;  
    display: block;  
    color: #f2f2f2;  
    text-align: center;  
    padding: 14px 16px;  
    text-decoration: none;  
}  
  
/* On screens that are 600px wide or less, make the menu links  
stack on top of each other instead of next to each other */
```

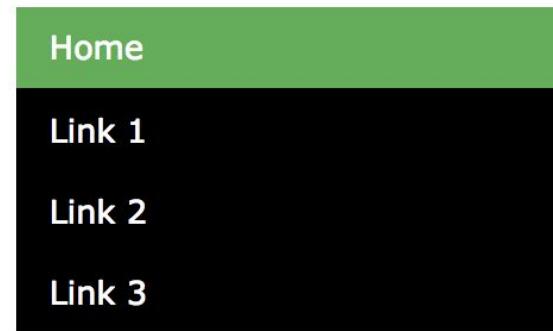
```
@media screen and (max-width: 600px) {  
.topnav a {  
    float: none;  
    width: 100%;  
}  
}
```



Large screens:



Small screens:



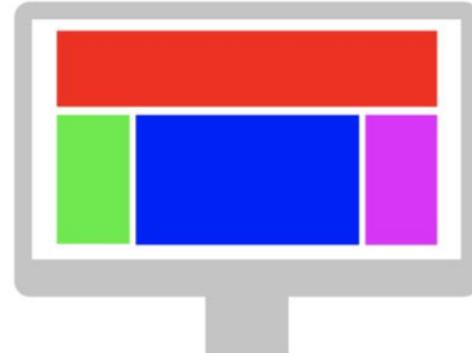
[https://www.w3schools.com/css/css3\\_mediaqueries\\_ex.asp](https://www.w3schools.com/css/css3_mediaqueries_ex.asp)

# Desktop First mit max

- Erstellen Sie die Seite,
- Schieben Sie Ihr Browserfenster zusammen
- Fügen Sie Media Queries für solche Breakpoints ein, bei denen Ihnen das Layout um die Ohren fliegt.
- Das nennt sich Desktop-First.

```
aside {  
  float: left;  
  width: 50%;  
}
```

```
@media (max-width: 50em) {  
  aside {  
    float: none;  
    width: auto;  
  }  
}
```

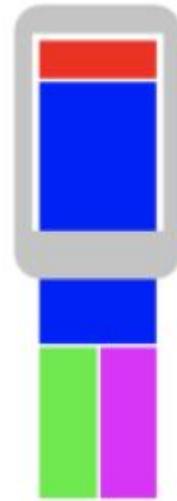


# Mobile First mit **min**

- Ziehen Sie das Browserfenster eng zusammen
- Erstellen die Seite.
- In einem zweiten Schritt ziehen Sie das Fenster auseinander und fügen Media Queries dort ein, wo die zusätzliche Breite sie sinnvoll machen.
- Das nennt sich dann Mobile-First.

```
aside {  
  /* nur Farb- und Hintergrundformatierungen  
 */  
}  
}
```

```
@media (min-width: 50em) {  
  aside {  
    float: left;  
    width: 50%;  
  }  
}
```



### 3. Viewport Units for RWD

## CSS-Maßeinheiten (*CSS Units*)

Relative  
CSS-Maßeinheiten

em, rem, ex,  
ch, vw, vh,  
**vmin, vmax,**  
%

Absolute  
CSS-Maßeinheiten

cm, mm, in,  
px, pt, pc

<https://www.w3.org/TR/css-values-3/>

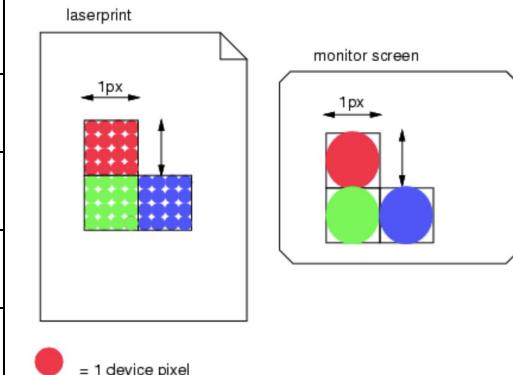
# Relative CSS-Maßeinheiten (*Relative CSS Units*)

Unit	Description
em	Relative to the font-size of the element (2em means 2 times the size of the current font)
ch	Relative to width of the "0" (zero)
rem	Relative to font-size of the root element
vw	Relative to 1% of the width of the viewport
vh	Relative to 1% of the height of the viewport
vmin	Relative to 1% of viewport's smaller dimension
vmax	Relative to 1% of viewport's larger dimension
%	Prozent

[https://www.w3schools.com/cssref/css\\_units.asp](https://www.w3schools.com/cssref/css_units.asp)

# Absolute CSS-Maßeinheiten (*Absolute CSS Units*)

Einheit	Beschreibung
cm	Zentimeter
mm	Millimeter
in	Inches ( $1\text{in} = 96\text{px} = 2.54\text{cm}$ )
px	Pixels ( $1\text{px} = 1/96\text{th of 1in} \approx$ (früher) ein Bildschirm-Pixel)
pt	Points ( $1\text{pt} = 1/72 \text{ of 1in}$ )
pc	Picas ( $1\text{pc} = 12 \text{ pt}$ )



<https://www.w3.org/TR/css-values-3/#absolute-lengths>

[https://www.w3schools.com/cssref/css\\_units.asp](https://www.w3schools.com/cssref/css_units.asp)

# 4. calc() Function Expressions

```
#div1 {  
    position: absolute;  
    left: 50px;  
    width: calc(100% - 22px);  
    border: 1px solid black;  
    background-color: yellow;  
    padding: 5px;  
    text-align: center;  
}
```

```
1 | h1 {  
2 |     font-size: calc(1.5rem + 3vw);  
3 | }
```

## Formal syntax

calc( <calc-sum> )

where

<calc-sum> = <calc-product> [ [ '+' | '-' ] <calc-product> ]\*

where

<calc-product> = <calc-value> [ '\*' <calc-value> | '/' <number> ]\*

where

<calc-value> = <number> | <dimension> | <percentage> | ( <calc-sum> )

<https://developer.mozilla.org/en-US/docs/Web/CSS/calc>

# 5. CSS Custom Properties (aka CSS Variables)

```
1 | .foo {  
2 |   --widthA: 100px;  
3 |   --widthB: calc(var(--widthA) / 2);  
4 |   --widthC: calc(var(--widthB) / 2);  
5 |   width: var(--widthC);  
6 | }
```

Deklaration mit `--name`

Verwendung mit `var(--name)`

# CSS Custom Properties (*CSS Variables*)

```
:root {  
    --main-bg-color: coral;  
    --main-txt-color: blue;  
    --main-padding: 15px;  
}
```

Deklaration mit `--xyz`

```
#div1 {  
    background-color: var(--main-bg-color);  
    color: var(--main-txt-color);  
    padding: var(--main-padding);  
}
```

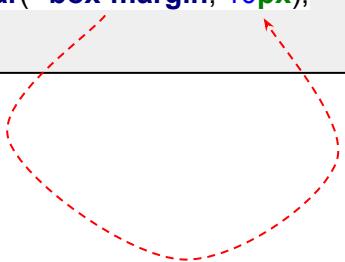
Verwendung mit `var()`

```
#div2 {  
    background-color: var(--main-bg-color);  
    color: var(--main-txt-color);  
    padding: var(--main-padding);  
}
```

# CSS Variables -2-

## Default Values

```
.box{  
  --box-color:#4d4e53;  
  --box-padding: 0 10px;  
  
  /* 10px is used because --box-margin is not  
   defined. */  
  margin: var(--box-margin, 10px);  
}
```



Da die Variable `--box-margin` nicht definiert ist, wird `10px` als Fallback genommen.

## Scope & Inheritance

```
:root {  
  --globalVar: 10px;  
}  
  
.enclosing {  
  --enclosingVar: 20px;  
}  
  
.enclosing .closure {  
  --closureVar: 30px;  
  
  font-size: calc(var(--closureVar)  
    + var(--enclosingVar) + var(--globalVar));  
  /* 60px for now */  
}
```

# CSS Variables in Media Queries

```
<!DOCTYPE html>
<head>
  <meta charset="UTF-8">
  <title>CSS Variables</title>
  <style>
    :root {
      --responsive-padding: 1rem;
    }
    @media (min-width: 576px) {
      :root {
        --responsive-padding: 2rem;
      }
    }
    .foo {
      padding: var(--responsive-padding);
    }
  </style>
</head>
<body>
  <h1 class="foo">Überschrift</h1>
</body>
```



```
<!DOCTYPE html>
<head>
  <meta charset="UTF-8">
  <title>CSS Variables as Multipliers</title>
  <style>
    :root {
      --multiplier: 1;
    }
    @media (min-width: 576px) {
      :root {
        --multiplier: 2;
      }
    }
    .foo {
      padding: calc( 1rem * var(--multiplier) );
    }
  </style>
</head>
<body>
  <h1 class="foo">Überschrift</h1>
</body>
```



# CSS Counter

```
<style>
  body {
    counter-reset: section;
  }

  h2::before {
    counter-increment: section;
    content: "Section " counter(section) ": ";
  }
</style>

<h1>Using CSS Counters:</h1>
<h2>HTML Tutorial</h2>
<h2>CSS Tutorial</h2>
<h2>JavaScript Tutorial</h2>
```

## Using CSS Counters:

**Section 1: HTML Tutorial**

**Section 2: CSS Tutorial**

**Section 3: JavaScript Tutorial**

[https://www.w3schools.com/css/css\\_counters.asp](https://www.w3schools.com/css/css_counters.asp)

# 6. Flexbox-Layout

Home    HTML    **CSS**    JavaScript    SQL    PHP    Bootstrap    How To    jQuery    W3.CSS    Angular

**CSS Advanced**

- CSS Rounded Corners
- CSS Border Images
- CSS Backgrounds
- CSS Colors
- CSS Gradients
- CSS Shadows
- CSS Text Effects
- CSS Web Fonts
- CSS 2D Transforms
- CSS 3D Transforms
- CSS Transitions
- CSS Animations
- CSS Tooltips
- CSS Style Images
- CSS object-fit
- CSS Buttons
- CSS Pagination
- CSS Multiple Columns
- CSS User Interface
- CSS Variables
- CSS Box Sizing
- CSS Flexbox**
- CSS Media Queries
- CSS MQ Examples

## The flex-wrap Property

The `flex-wrap` property specifies whether the flex items should wrap or not.

The examples below have 12 flex items, to better demonstrate the `flex-wrap` property.

1	2	3	4	5	6
9	10	11	12		

### Example

The `wrap` value specifies that the flex items will wrap if necessary:

```
.flex-container {  
    display: flex;  
    flex-wrap: wrap;  
}
```

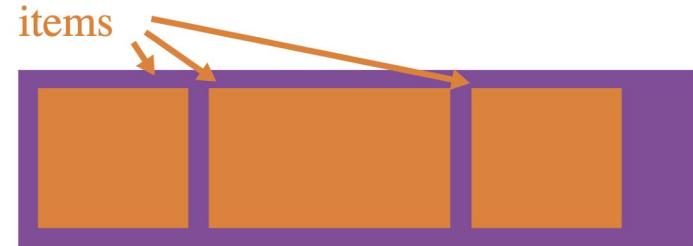
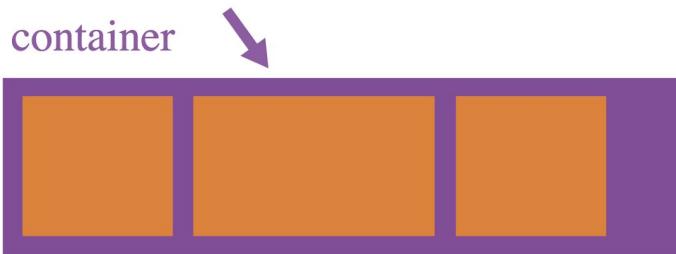
[Try it Yourself »](#)

[https://www.w3schools.com/css/css3\\_flexbox.asp](https://www.w3schools.com/css/css3_flexbox.asp)



# CSS Tricks Complete Guide to Flexbox

## ► Basics & Terminology



### Properties for the Parent (flex container)

#### #display

This defines a flex container; inline or block depending on the given value. It enables a flex context for all its direct children.

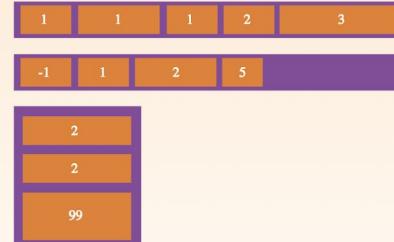
```
css
.container {
  display: flex; /* or inline-flex */
}
```

Note that CSS columns have no effect on a flex container.

#### #flex-direction

### Properties for the Children (flex items)

#### #order



# Beispiel: RWD Menu with Flexbox

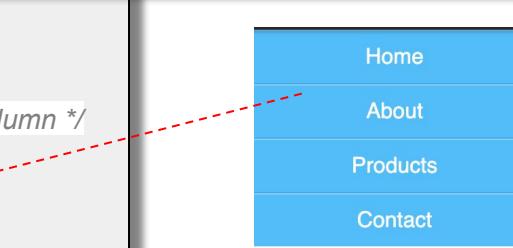
```
/* Large */
.navigation {
  display: flex;
  flex-flow: row wrap;
  /* This aligns items to the end line on main-axis */
  justify-content: flex-end;
}

/* Medium screens */
@media all and (max-width: 800px) {
  .navigation {
    /* When on medium sized screens, we center it */
    justify-content: space-around;
  }
}

/* Small screens */
@media all and (max-width: 500px) {
  .navigation {
    /* On small screens, we are no longer using row direction but column */
    flex-direction: column;
  }
}
```



```
<ul class="navigation">
  <li><a href="#">Home</a></li>
  <li><a href="#">About</a></li>
  <li><a href="#">Products</a></li>
  <li><a href="#">Contact</a></li>
</ul>
```



# Codepen Flexbox Playground

 Flexbox playground  
A PEN BY Gabi PRO

[Fork](#) [Change View](#) [Log In](#) [Sign Up](#)

## Properties for the flex container

**FLEX-DIRECTION** ( property of the flex container )

row:  row-reverse:  column:  column-reverse:



1 2 3 4 5

**FLEX-WRAP** ( property of the flex container )

nowrap:  wrap:  wrap-reverse:



1 2 3 4 5

<https://codepen.io/enxaneta/full/adLPwy>

# Exkurs: The box-sizing CSS property

**box-sizing: content-box;**

W3C box model



**box-sizing: border-box;**

Internet Explorer box model



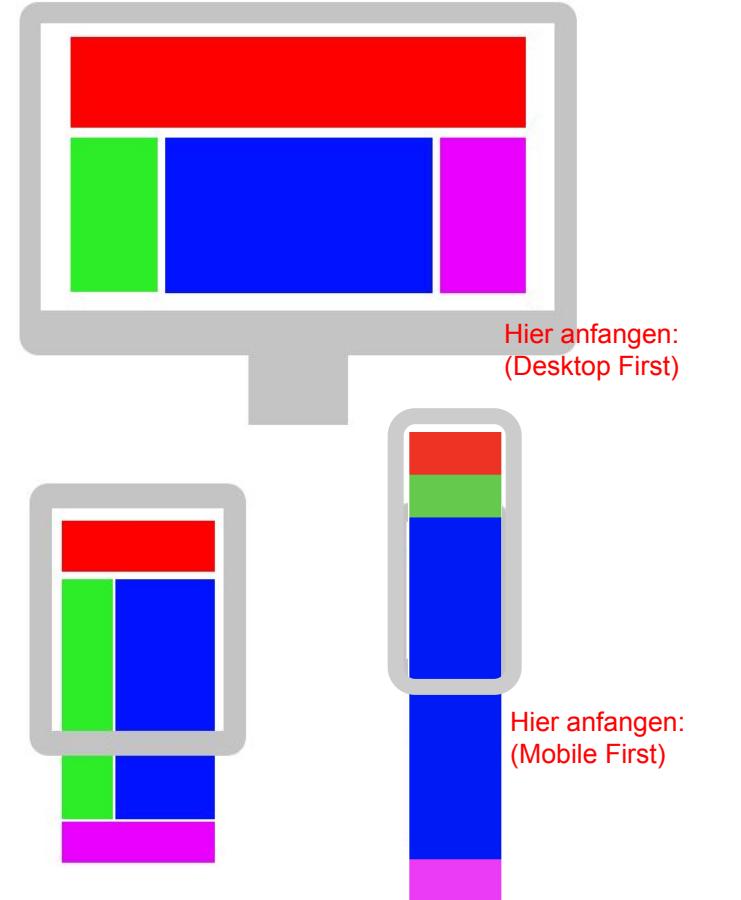
The **box-sizing** CSS property sets how the total width and height of an element is calculated.

<https://developer.mozilla.org/en-US/docs/Web/CSS/box-sizing>

# Übungen: Media Queries for RWD

- Führen Sie Breakpoints bei 600px und 300px ein, so dass folgende RWD-LAYOUTS entstehen:

```
<body>
<header></header>
<aside class="left"></aside>
<article></article>
<aside class="right"></aside>
</body>
```

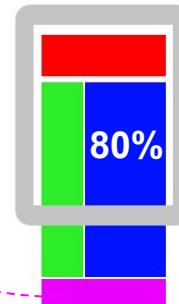
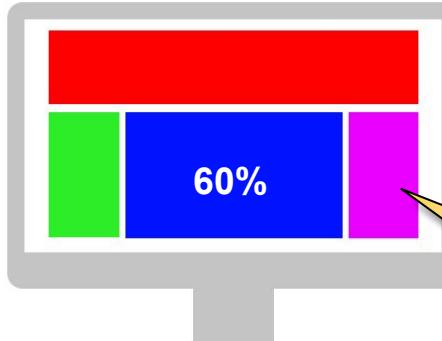


[https://en.wikipedia.org/wiki/Responsive\\_web\\_design](https://en.wikipedia.org/wiki/Responsive_web_design)

# Kann man in Flexbox die Reihenfolge tauschen?

```
* {  
  box-sizing: border-box;  
}  
  
body {  
  display: flex;  
  flex-wrap: wrap;  
}  
  
@media screen and (max-width: 600px) {  
  aside.right { width: 100%; height: 5rem; }  
  article { width: 80%; }  
}  
  
@media screen and (max-width: 300px) {  
  article { width: 100%; }  
  aside.left, aside.right { width: 50%; height: 5rem; }  
  article { order: 1; }  
  aside.left { order: 2; }  
  aside.right { order: 3; }  
}
```

order



rot - grün - blau - pink

rot - blau - grün - pink

# Grid-System mit Flexbox und CSS Variables

```
<style>
.container {
  display: flex;
  flex-wrap: wrap;
  margin: 0 auto;
}

.column {
  --columns: 12; /* Number of columns in the grid system */
  --width: 1; /* Default width of the element */

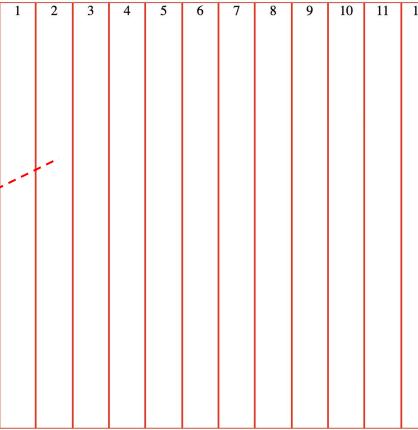
  flex-basis: calc(var(--width) / var(--columns) * 100% - 2px);
}

.header, .content, .sidebar {
  --width: 12;
}

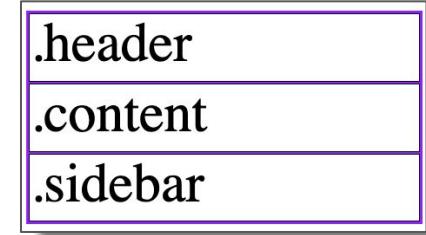
@media (min-width: 360px) {
  .content {
    --width: 6;
  }
  .sidebar {
    --width: 6;
  }
}

@media (min-width: 520px) {
  .content {
    --width: 8;
  }
  .sidebar {
    --width: 4;
  }
}
</style>
```

in 12 Teile zerlegt



ab 360px

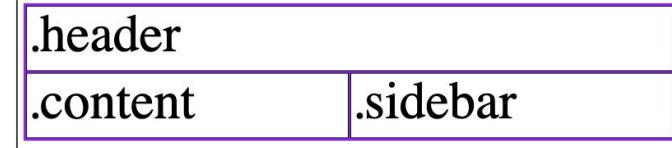


```
<body>


<header class="header column">
    .header
  </header>
  <main class="content column">
    .content
  </main>
  <aside class="sidebar column">
    .sidebar
  </aside>
</div>
</body>


```

ab 520px



# 7. Grid-Layout

HTML CSS JAVASCRIPT SQL PHP BOOTSTRAP HOW TO JQUERY W3.CSS ANGULAR MORE ▾ REFERENCES ▾

- CSS Pagination
- CSS Multiple Columns
- CSS User Interface
- CSS Variables
- CSS Box Sizing
- CSS Flexbox
- CSS Media Queries
- CSS MQ Examples
- CSS Responsive**
- RWD Intro
- RWD Viewport
- RWD Grid View
- RWD Media Queries
- RWD Images
- RWD Videos
- RWD Frameworks
- RWD Templates
- CSS Grid**
- Grid Intro
- Grid Container
- Grid Item**
- CSS Examples**
- CSS Templates
- CSS Examples
- CSS Quiz

## Naming Grid Items

The `grid-area` property can also be used to assign names to grid items.

Header				
Menu	Main	Right		
Footer				

Named grid items can be referred to by the `grid-template-areas` property of the grid container.

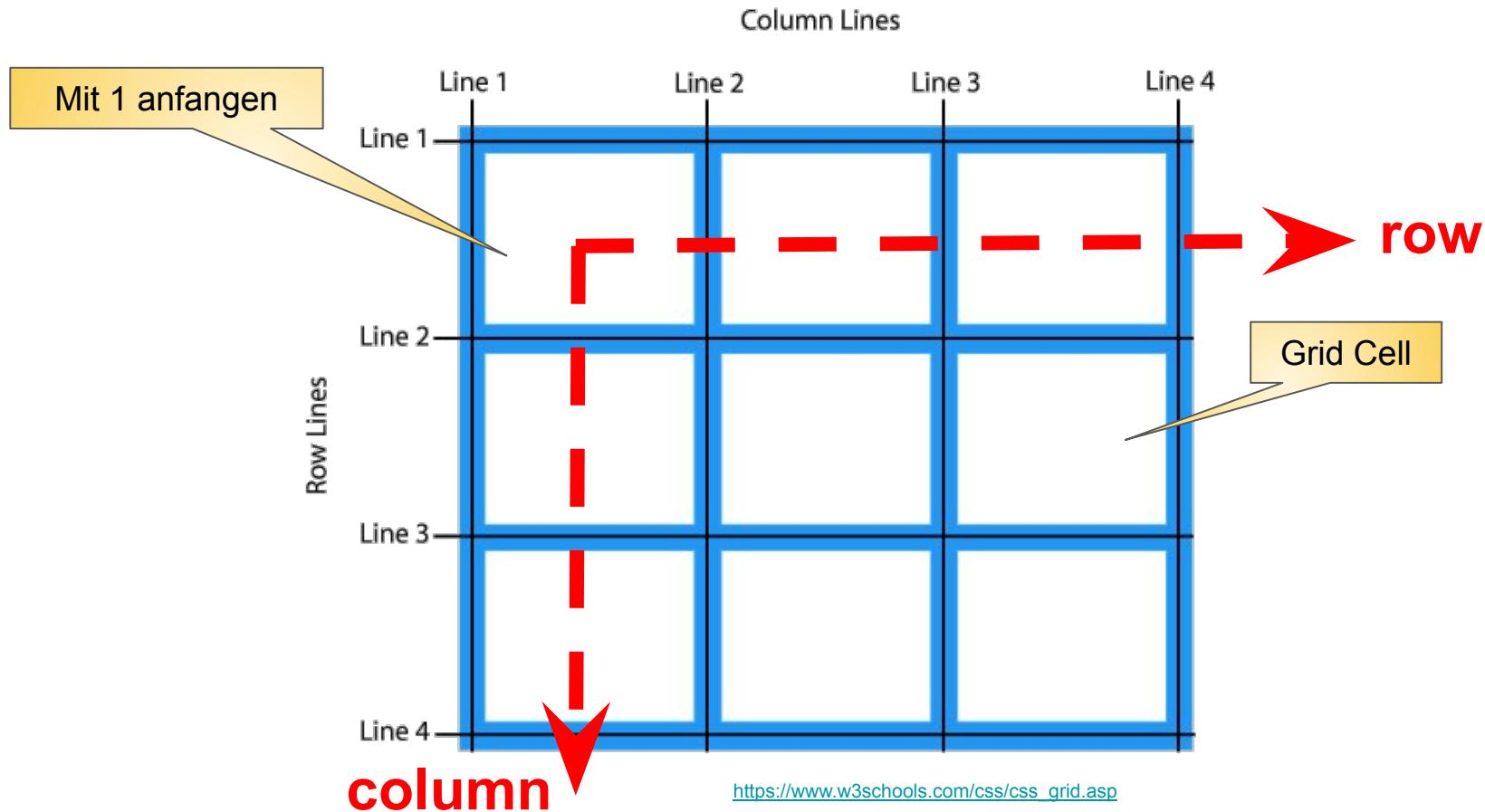
### Example

Item1 gets the name "myArea" and spans all five columns in a five columns grid layout:

```
.item1 {  
    grid-area: myArea;  
}  
.grid-container {  
    grid-template-areas: 'myArea myArea myArea myArea myArea';  
}
```

[Try it Yourself »](#)

# Grid-Terminologie



# Erzeugen von Grids mit repeat(3, 1fr)

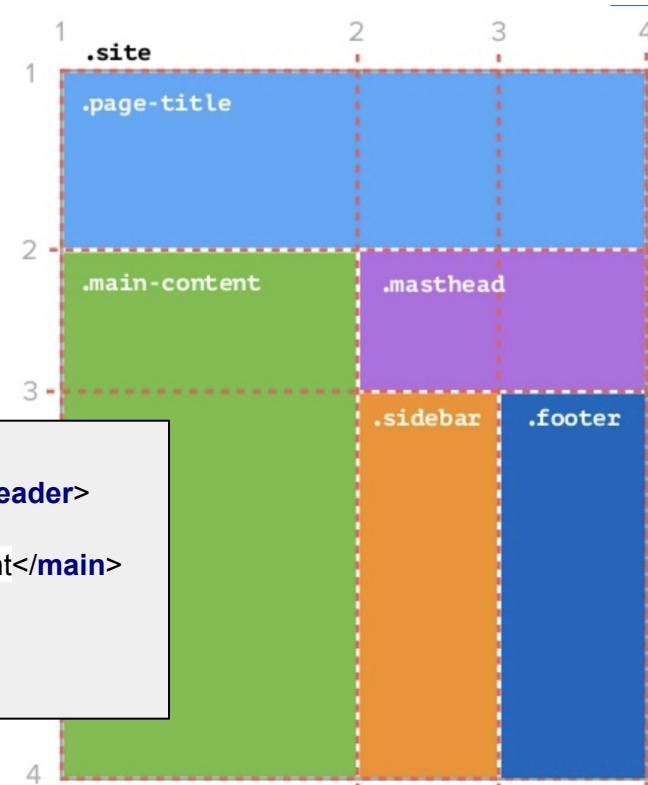
```
body {  
  display: grid;  
  grid-template-columns: repeat(3, 1fr);  
}  
  
@media (min-width: 400px) {  
  body {  
    grid-template-columns: repeat(4, 1fr);  
  }  
}
```

[https://wiki.selfhtml.org/wiki/CSS/Tutorials/Einstieg/Media\\_Queries#Mobile first](https://wiki.selfhtml.org/wiki/CSS/Tutorials/Einstieg/Media_Queries#Mobile_first)

# Grid mit "Anfang / Ende-Notation"

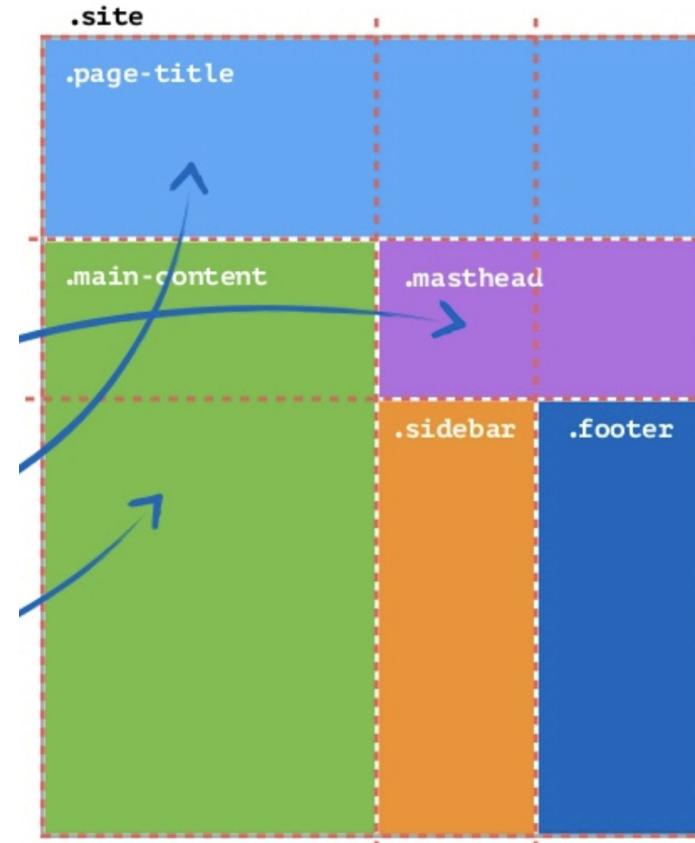
```
<style>
div * {
  border: thin dashed red;
  min-height: 10vh;
}
.site {
  display: grid;
  grid-template-columns: 2fr 1fr 1fr;
  grid-template-rows: auto 1fr 3fr;
}
.masthead {
  grid-column: 2/4;
  grid-row: 2/3;
}
.page-title {
  grid-column: 1/4;
  grid-row: 1/2;
}
.main-content {
  grid-column: 1/2;
  grid-row: 2/4;
}
...
</style>
```

```
<div class="site">
  <header class="masthead">masthead</header>
  <nav class="page-title">page-title</nav>
  <main class="main-content">main-content</main>
  <aside class="sidebar">sidebar</aside>
  <footer class="footer">footer</footer>
</div>
```



# Grid Template Areas

```
<style>
div * {
  border: thin dashed red;
  min-height: 10vh;
}
.site {
  display: grid;
  grid-template-columns: 2fr 1fr 1fr;
  grid-template-rows: auto 1fr 3fr;
  grid-template-areas:
    "title title title"
    "main header header"
    "main sidebar footer";
}
.masthead { grid-area: header; }
.page-title { grid-area: title; }
.main-content { grid-area: main; }
.sidebar { grid-area: sidebar; }
.footer { grid-area: footer; }
</style>
```



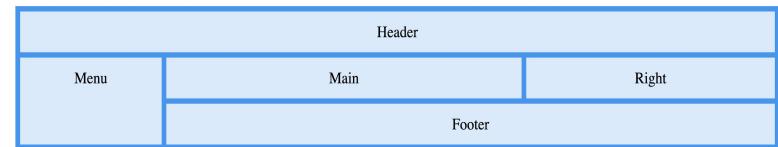
# Positionierung eines Items im Grid

```
.item8 {  
    grid-area: 1 / 2 / 5 / 6;  
}
```

Make "item8" start on row-line 1 and column-line 2, and end on row-line 5 and column line 6.

```
.item8 {  
    grid-area: 2 / 1 / span 2 / span 3;  
}
```

Make "item8" start on row-line 2 and column-line 1, and span 2 rows and 3 columns.



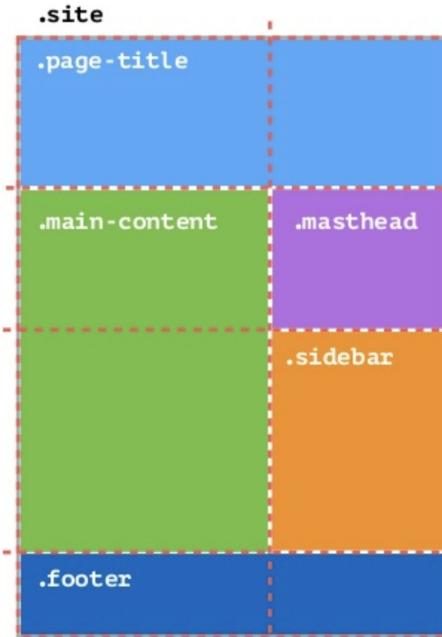
```
.item1 { grid-area: header; }  
.item2 { grid-area: menu; }  
.item3 { grid-area: main; }  
.item4 { grid-area: right; }  
.item5 { grid-area: footer; }  
  
.grid-container {  
    grid-template-areas:  
        'header header header header header header'  
        'menu main main main right right'  
        'menu footer footer footer footer footer';  
}
```

# RWD mit Grid

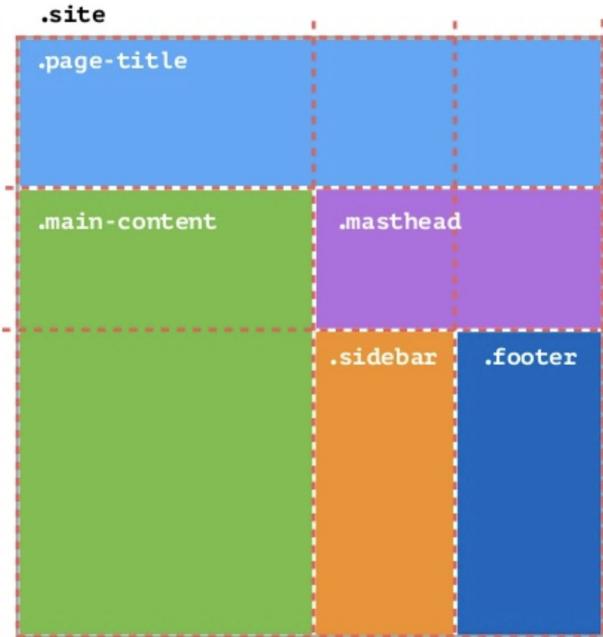
```
.site {  
    display: grid;  
    grid-template-columns: 1fr 1fr;  
    grid-template-rows: auto 1fr 3fr;  
    grid-template-areas:  
        "title title"  
        "main header"  
        "main sidebar"  
        "footer footer";  
}  
  
.masthead { grid-area: header; }  
.page-title { grid-area: title; }  
.main-content { grid-area: main; }  
.sidebar { grid-area: sidebar; }  
.footer { grid-area: footer; }  
  
@media screen and (min-width: 34em){  
    .site {  
        grid-template-columns: 2fr 1fr 1fr;  
        grid-template-areas:  
            "title title title"  
            "main header header"  
            "main sidebar footer";  
    }  
}
```



Two-column grid



Three-column grid



<https://www.slideshare.net/mor10/css-grid-changes-everything-about-web-layouts-wordcamp-europe-2017/42>

```

.site {
  display: grid;
  grid-template-columns: 1fr;
  grid-template-rows: auto 1fr 3fr 2fr 1fr;
  grid-template-areas:
    "header"
    "title"
    "main"
    "sidebar"
    "footer";
}

.masthead { grid-area: header; }
.page-title { grid-area: title; }
.main-content { grid-area: main; }
.sidebar { grid-area: sidebar; }
.footer { grid-area: footer; }

@media screen and (min-width: 34em){
  .site {
    grid-template-columns: 2fr 1fr;
    grid-template-areas:
      "title title"
      "main header"
      "main sidebar"
      "footer footer";
  }
}

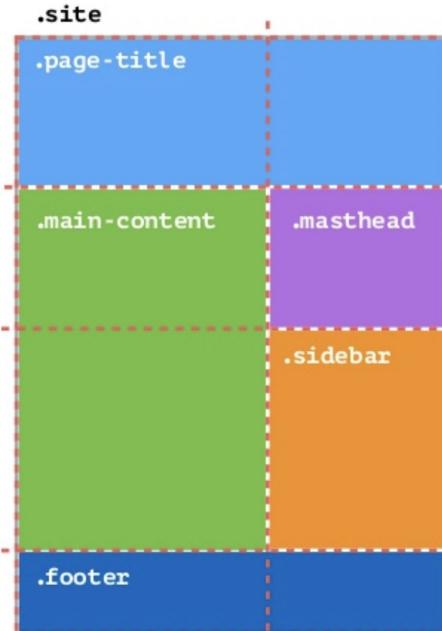
```

# RWD mit 1 & 2 Spalten

No grid



Two-column grid



<https://www.slideshare.net/mor10/css-grid-changes-everything-about-web-layouts-wordcamp-europe-2017/42>

# CSS Tricks Complete Guide to Grid

The screenshot shows the 'A Complete Guide to Grid' article on the CSS-Tricks website. The page has a dark blue header with white text. On the left, there's a vertical sidebar with icons for Home, Video, PDF, and Snippets. The main content area features a large title 'A Complete Guide to Grid' and author information 'BY CHRIS HOUSE LAST UPDATED ON APRIL 23, 2018'. Below the title, there's a paragraph about CSS Grid Layout.

CSS Grid Layout is the most powerful layout system available in CSS. It is a 2-dimensional system, meaning it can handle both columns and rows, unlike [flexbox](#) which is largely a 1-dimensional system. You work with Grid Layout by applying CSS rules both to a parent element (which becomes the Grid Container) and to that element's children (which become Grid Items).

<https://css-tricks.com/snippets/css/complete-guide-grid/>

# CodePen Grid Playground

The screenshot shows the CodePen interface for a 'CSS Grid playground' pen by Morten Rand-Hendriksen. The header includes the title, a 'Fork' button, 'Change View' button, and navigation links for 'Log In' and 'Sign Up'. The main content area features a large heading 'CSS Grid Playground' and a sub-heading 'Get to know the CSS Grid Layout Module'. Below this is a visual representation of a CSS grid layout with seven colored boxes labeled Box 1 through Box 7. The layout consists of two rows: the top row contains three boxes (Box 1, Box 2, Box 3) in a light blue, purple, and teal gradient; the bottom row contains three boxes (Box 4, Box 5, Box 7) in a yellow, orange, and grey gradient. All boxes are rounded rectangles with a thin black border.

<https://codepen.io/mor10/full/MEQJbM>

## 8. Flexible Images

Erstes Verfahren mit CSS-Regeln mit Media Queries:

```
/* For width smaller than 400px: */
body {
    background-image: url('img_smallflower.jpg');
}

/* For browser width 400px and larger: */
@media only screen and (min-width: 400px) {
    body {
        background-image: url('img_flowers.jpg');
    }
}
```

[https://www.w3schools.com/css/css\\_rwd\\_images.asp](https://www.w3schools.com/css/css_rwd_images.asp)

# Flexible Images

HTML CSS JAVASCRIPT SQL PHP BOOTSTRAP HOW TO JQUERY W3.CSS ANGULAR XML MORE ▾

CSS Gradients  
CSS Shadows  
CSS Text Effects  
CSS Web Fonts  
CSS 2D Transforms  
CSS 3D Transforms  
CSS Transitions  
CSS Animations  
CSS Tooltips  
CSS Style Images  
CSS object-fit  
CSS Buttons  
CSS Pagination  
CSS Multiple Columns  
CSS User Interface  
CSS Variables  
CSS Box Sizing  
CSS Flexbox  
CSS Media Queries  
CSS MQ Examples

CSS Responsive  
RWD Intro  
RWD Viewport  
RWD Grid View  
RWD Media Queries

RWD Images

RWD Videos  
RWD Frameworks  
RWD Templates

CSS Grid  
Grid Intro  
Grid Container  
Grid Item

CSS Examples  
CSS Templates  
CSS Examples  
CSS Quiz  
CSS Exercises

## Responsive Web Design - Images

◀ Previous



Resize the browser window to see how the image scales to fit the page.

```
<figure>  
  
    
  
  <figcaption>Bildunterschrift</figcaption>  
  
</figure>
```

## Using The width Property

If the `width` property is set to 100%, the image will be responsive and scale

### Example

```
img {  
  width: 100%;  
  height: auto;  
}
```

```
<picture> /* For browser width 400px and smaller: */  
  <source srcset="img_smallflower.jpg" media="(max-width: 400px)">  
  <source srcset="img_flowers.jpg">  
    
</picture>
```

# 9. Flexible Videos

```
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<style>
  video {
    width: 100%;
    height: auto;
  }
</style>

<video controls>
  <source src="https://www.w3schools.com/css/mov_bbb.mp4" type="video/mp4">
  <source src="https://www.w3schools.com/css/mov_bbb.ogv" type="video/ogg">
  Your browser does not support HTML5 video.
</video>

<p>Resize the browser window to see how the size of the video player will scale.</p>
```

[https://www.w3schools.com/css/css\\_rwd\\_videos.asp](https://www.w3schools.com/css/css_rwd_videos.asp)

# 10. Responsive Font Size

Aufgabe: Text-Zeilen sollen bei ViewPort-Änderungen ungefähr erhalten bleiben

```
<style>
  p {
    /* Fallback */
    font-size: 18px;

    /* Responsive */
    font-size: 1.29vw;
  }
</style>
```

**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. 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. 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.



<http://emilolsson.com/tools/vw-unit-calc-an-online-responsive-css-font-size-calculator/>

# Wie lernt man Responsive Design?



[https://wiki.selfhtml.org/wiki/HTML/Tutorials/responsive\\_Webdesign](https://wiki.selfhtml.org/wiki/HTML/Tutorials/responsive_Webdesign)

w3schools.com

THE WORLD'S LARGEST

HTML CSS JAVASCRIPT SQL PHP BOOTSTRAP HOW TO JQUERY W3.CSS ANGULAR MORE REFERENCES ▾

CSS User Interface  
CSS Variables  
CSS Box Sizing  
CSS Flexbox  
CSS Media Queries  
CSS MQ Examples

RWD Intro  
RWD Viewport  
RWD Grid View  
RWD Media Queries  
RWD Images  
RWD Videos  
RWD Frameworks  
RWD Templates

CSS Grid  
Grid Intro  
Grid Container  
Grid Item

CSS Examples  
CSS Templates  
CSS Examples  
CSS Quiz  
CSS Exercises  
CSS Certificate

## Responsive Web Design - Introduction

◀ Previous Next ▶

### What is Responsive Web Design?

Responsive web design makes your web page look good on all devices.

Responsive web design uses only HTML and CSS.

Responsive web design is not a program or a JavaScript.

### Designing For The Best Experience For All Users

Web pages can be viewed using many different devices: desktops, tablets, and phones. Your web page should look good, and be easy to use, regardless of the device.

Web pages should not leave out information to fit smaller devices, but rather adapt its content to fit any device:



[https://www.w3schools.com/css/css\\_rwd\\_intro.asp](https://www.w3schools.com/css/css_rwd_intro.asp)

# Videos on RWD

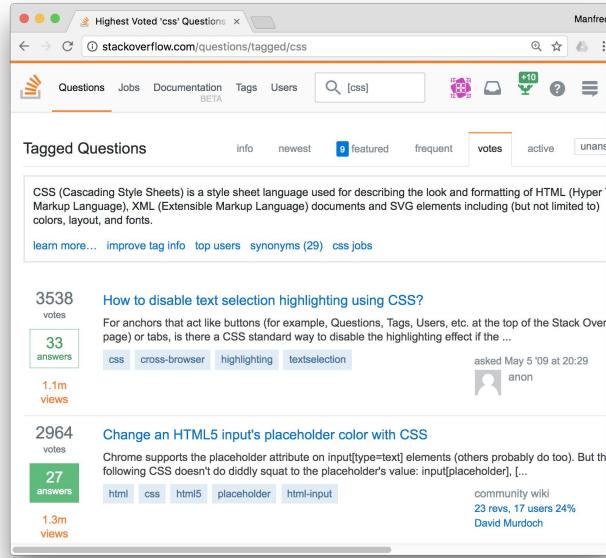
YOU CAN  
ADD & SUBTRACT  
CSS UNITS  
WITH A  
SIMILAR TYPE

```
.example-1 {  
  font-size: calc(10px + 2rem);  
}  
  
.example-2 {  
  font-size: calc(100px - 50px);  
}
```

@MIKEBIRTHMULLER

<https://vimeo.com/235428198>

# Wo bekommt man Antworten?



A screenshot of a web browser window titled "Highest Voted 'css' Questions". The URL in the address bar is "stackoverflow.com/questions/tagged/css". The page shows a list of tagged questions under the heading "Tagged Questions". The first question is "How to disable text selection highlighting using CSS?", which has 3538 votes, 33 answers, and 1.1m views. The second question is "Change an HTML5 input's placeholder color with CSS", which has 2964 votes, 27 answers, and 1.3m views.



<http://stackoverflow.com/questions/tagged/css>

# Browser-Abhängigkeiten: Can I use column-break-inside?

CSS3 Multiple column layout  - CR

Method of flowing information in multiple columns

Current aligned Usage relative Date relative Show all

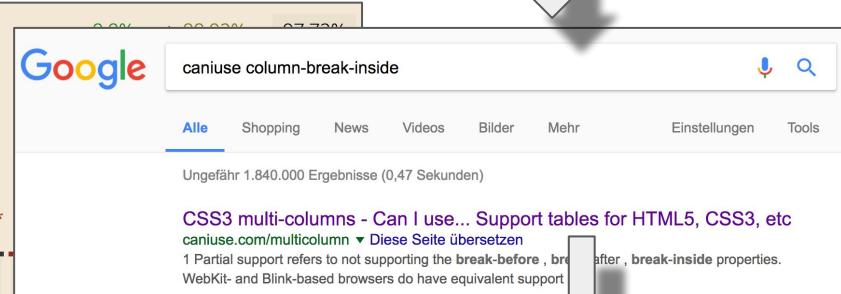
IE	Edge	Firefox	Chrome	Safari	Opera	iOS Safari
			1 2 49 1 55			
		1 51 1 52	1 56 1 57	1 10	1 43	1 9.3 1 10.2
11	15	1 53 1 54 1 55 1 56	1 58 1 59 1 60 1 61	1 10.1	1 44	1 10.3
				TP	1 45 1 46	1 4.4 1 4.4.4
						1 56 1 57

Global unprefixed: Germany unprefixed:

Notes Known issues (9) Resources (10) Feedback

<sup>1</sup> Partial support refers to not supporting the `break-before`, `break-after`, `break-inside` properties. WebKit- and Blink-based browsers do have equivalent support for the non-standard `-webkit-column-break-*` properties to accomplish the same result (but only the `auto` and `always` values). Firefox does not support `break-*`.

<sup>2</sup> Partial support refers to not supporting the `column-fill` property.



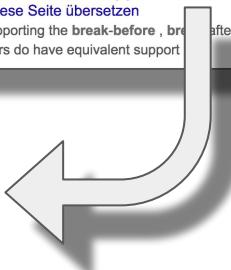
Google caniuse column-break-inside

Alle Shopping News Videos Bilder Mehr Einstellungen Tools

Ungefähr 1.840.000 Ergebnisse (0,47 Sekunden)

CSS3 multi-columns - Can I use... Support tables for HTML5, CSS3, etc caniuse.com/multicolumn ▾ Diese Seite übersetzen

1 Partial support refers to not supporting the `break-before`, `break-after`, `break-inside` properties. WebKit- and Blink-based browsers do have equivalent support for the non-standard `-webkit-column-break-*` properties to accomplish the same result (but only the `auto` and `always` values). Firefox does not support `break-*`.



<http://caniuse.com/#feat=multicolumn>

54

# Holy Grail Layout “Tableless”, “Responsive”

