

Responsive Web Design (RWD)

Einführung in
Web Engineering
CSS Teil 2
Manfred Kaul



Quelle: https://de.wikipedia.org/wiki/Responsive_Webdesign



Quelle: https://en.wikipedia.org/wiki/Cascading_Style_Sheets

Was ist der Unterschied zwischen PDF und HTML?

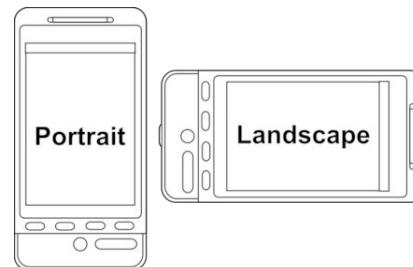
<http://entnemdept.ufl.edu/walker/pdfvhtml.htm>

Definition Responsive Webdesign (RWD)

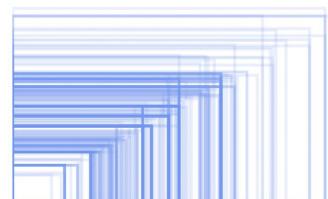
- Unter einer responsiven Gestaltung einer Webseite versteht man, dass sich das Layout der Seite an das Ausgabemedium selbstständig (möglichst nahtlos, fluide) anpasst.
- Der grafische Aufbau einer „responsiven“ Webseite folgt den Anforderungen des jeweiligen Gerätes dynamisch.



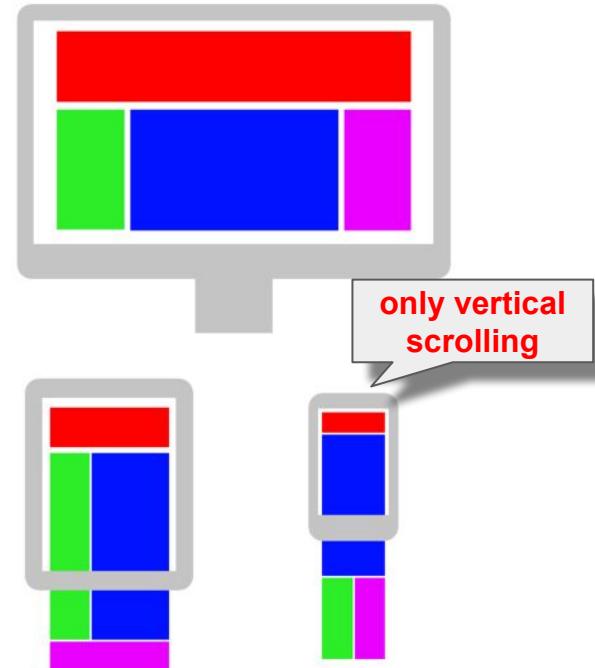
https://wiki.selfhtml.org/wiki/HTML/Tutorials/responsive_Webdesign



https://en.wikipedia.org/wiki/Page_orientation



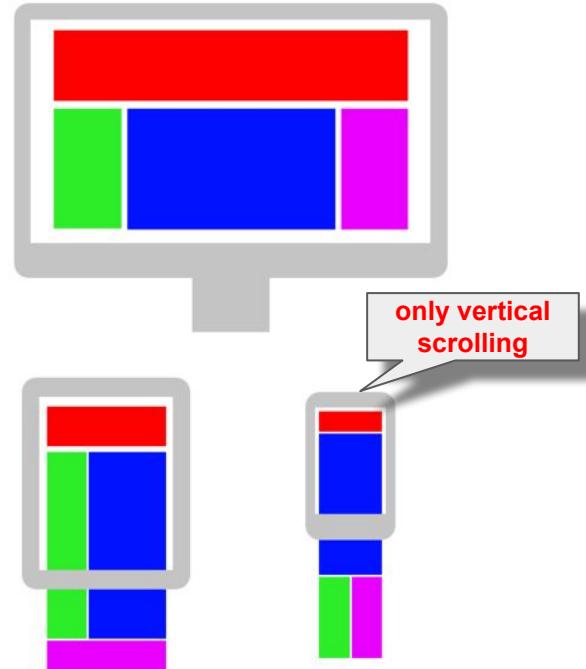
Viewportauflösungen heutiger Geräte
<https://opensignal.com/reports/fragmentation.php>



https://en.wikipedia.org/wiki/Responsive_web_design

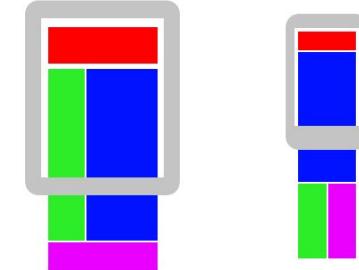
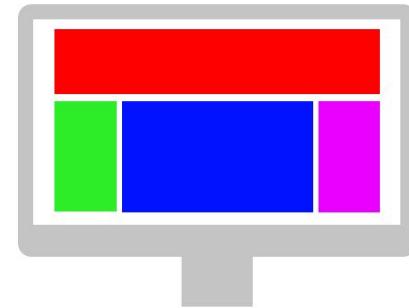
Definitionen: responsiv - fluide - mobil - adaptiv - liquid

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



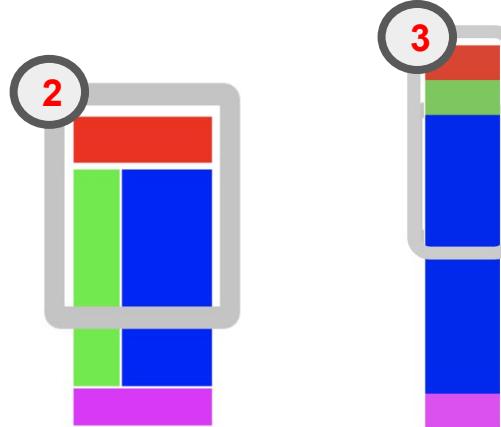
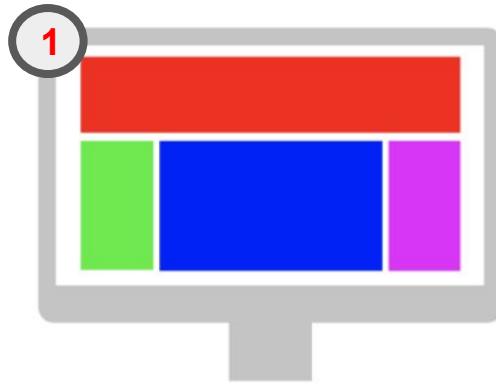
Zwei Methoden für Responsive Design

1. Desktop First
2. Mobile First



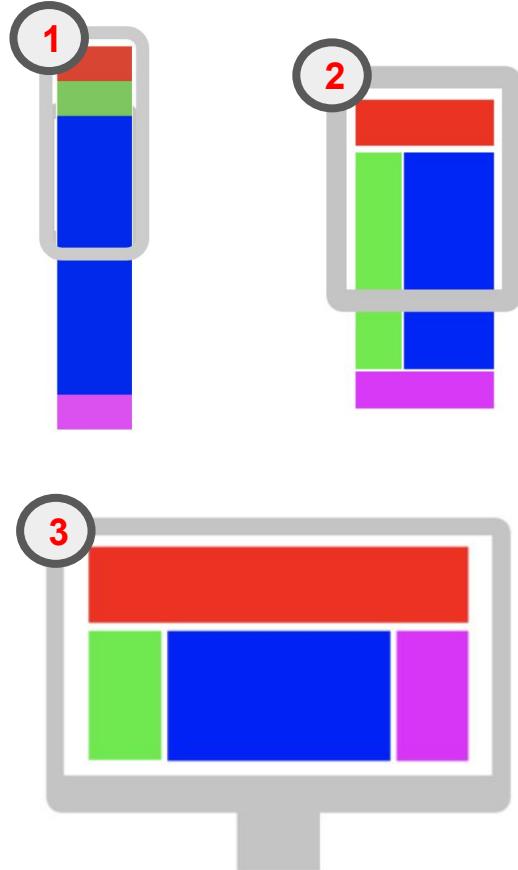
Desktop First

- 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

1. <meta>-Tag viewport

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



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

1. <meta>-Tag viewport

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

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://medium.com/@flik185/understanding-device-resolution-for-web-design-and-development-3bb4a5183478>

2. Media Queries (in HTML, CSS und JavaScript)

Mit **Medienabfragen** können Autoren Werte oder Funktionen des Benutzeragenten oder des Anzeigegeräts unabhängig vom gerenderten Dokument testen und abfragen. Sie werden in der CSS @ media-Regel verwendet, um Stile bedingt auf ein Dokument und in verschiedenen anderen Kontexten und Sprachen wie HTML und JavaScript anzuwenden.

HTML

```
<link rel="stylesheet" href="screen.css" media="screen" />
<link rel="stylesheet" href="print.css" media="print" />
```

CSS

@media screen (min-width: 320px) and (max-width: 768px)

AT-RULE

MEDIA TYPE

MEDIA FEATURE

OPERATOR

MEDIA FEATURE

JavaScript

```
<!DOCTYPE html>
<script>
    // Create a condition that targets viewports at least 768px wide
    const mediaQuery = window.matchMedia('(min-width: 768px)');
    mediaQuery.addEventListener( "change", console.log );
</script>
```

<https://css-tricks.com/a-complete-guide-to-css-media-queries/>

Media Queries in CSS

@media screen (min-width: 320px) and (max-width: 768px)

AT-RULE	MEDIA TYPE	MEDIA FEATURE	OPERATOR	MEDIA FEATURE
	screen	min-width	and	
	print	max-width	or	
	speech	min-height	not	
	all	max-height		
		aspect-ratio		
		orientation		

Beispiele Media Queries

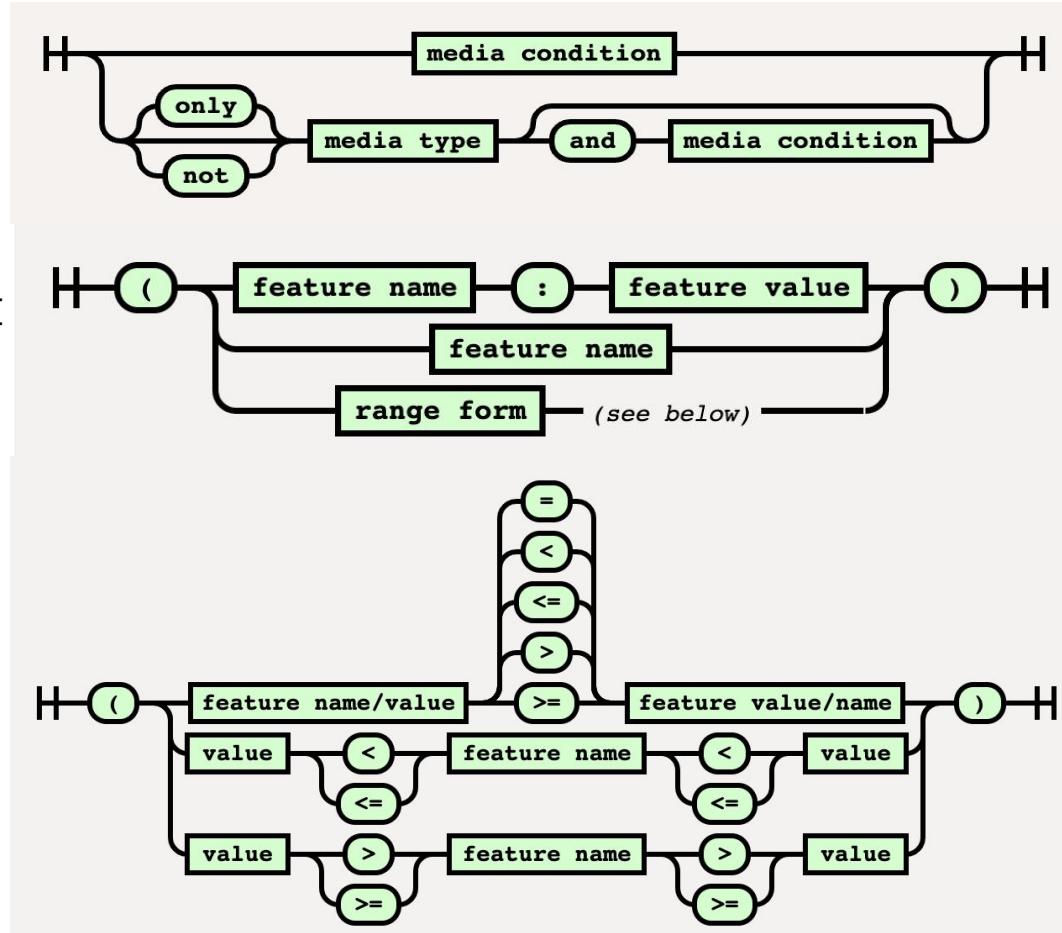
```
@media print {  
    * {  
        background-color: white;  
        font-family: "times new roman", times, serif;  
        text-align: justify;  
    }  
}  
  
@media screen and (max-width: 220px){ }  
@media (min-width: 30em) and (orientation: landscape) { }  
@media screen and (min-width: 30em) and (orientation: landscape) { }  
@media (min-height: 680px), screen and (orientation: portrait) { }  
@media (min-width: 30em) and (max-width: 50em) { }
```

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

Weitere Medienmerkmale

```
@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){ }
```

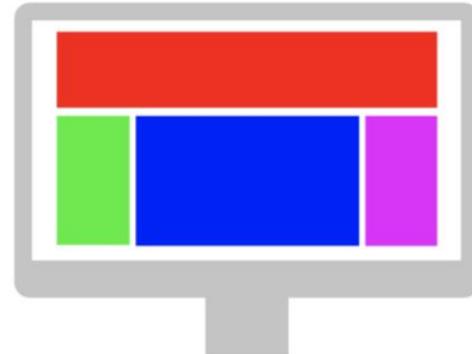


Desktop First mit max

- 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;  
  }  
}
```

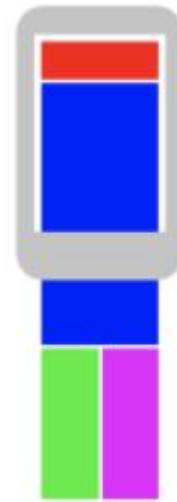


Mobile First mit min

- 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

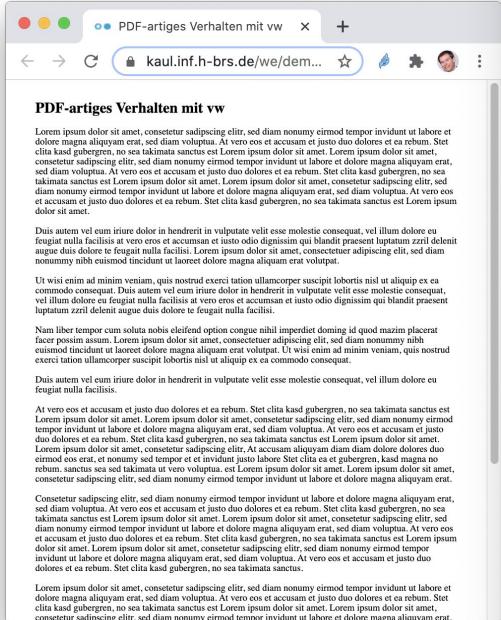
em, rem, ex,
ch,
vw, vh,
vmin, vmax, relativ zum Viewport
%

Absolute
CSS-Maßeinheiten

cm, mm, in,
px, pt, pc

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

Variable Font Size mit vw



PDF-artiges Verhalten mit vw

Et dolor 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. Et dolor 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. Et dolor 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.

Duis autem vel eum iriato 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. Lorem ipsum dolor sit amet, consecetuer adipiscing elit, sed diam nonumquam nibh euismod tincidunt ut laoreet 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 facilisi at vero eros et accumsan et iusto odio dignissim qui blandit praesent lurtatibus zzril delenit ague duis dolore te, feugiat nulla facilisi.

Nam liber tempor cum soluta nobis eleifend option congue nihil imperdiet doming id quod mazim placerat facer possim assum. Lorem ipsum dolor sit amet, consecetur adipiscing elit, sed diam nonummy nibh euismod tincidunt ut laoreet dolore magna aliquam erat volutpat. Ut wisi enim ad minim veniam, quis nostrud exercitation 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 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, At accusam aliquyam diam dolore dolores duo eirmod eos erat, et nonumy sed tempor et et invidunt justo labore Stet clita ea et gubergren, kasd magna no rebum, sanctus sea sea takimata ut vero voluptua, 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 setetut adipsicing elit, 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 kas ubergren, 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 paena aliquyam erat, sed diam voluptua. At vero eos

```
<style>
  * {
    margin: 2vw;
    padding: 0;
    font-size: 1.92vw;
    /** ( 100 - 4 ) / 50 ***/
  }
  h1 {
    font-size: 3vw;
  }
</style>
```

4. calc() Function Expressions



CSS Demo: calc()

```
width: calc(10px + 100px);
```

```
width: calc(100% - 30px);
```

```
width: calc(2em * 5);
```

```
width: calc(var(--variable-width) + 20px);
```

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

This ensures that text size will scale if the page is zoomed.

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 | }
```

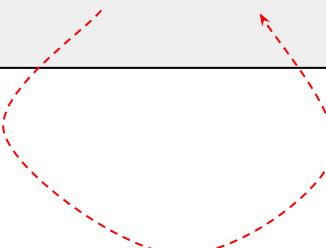
Deklaration mit `--name`

Verwendung mit `var(--name)`

CSS 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>
```

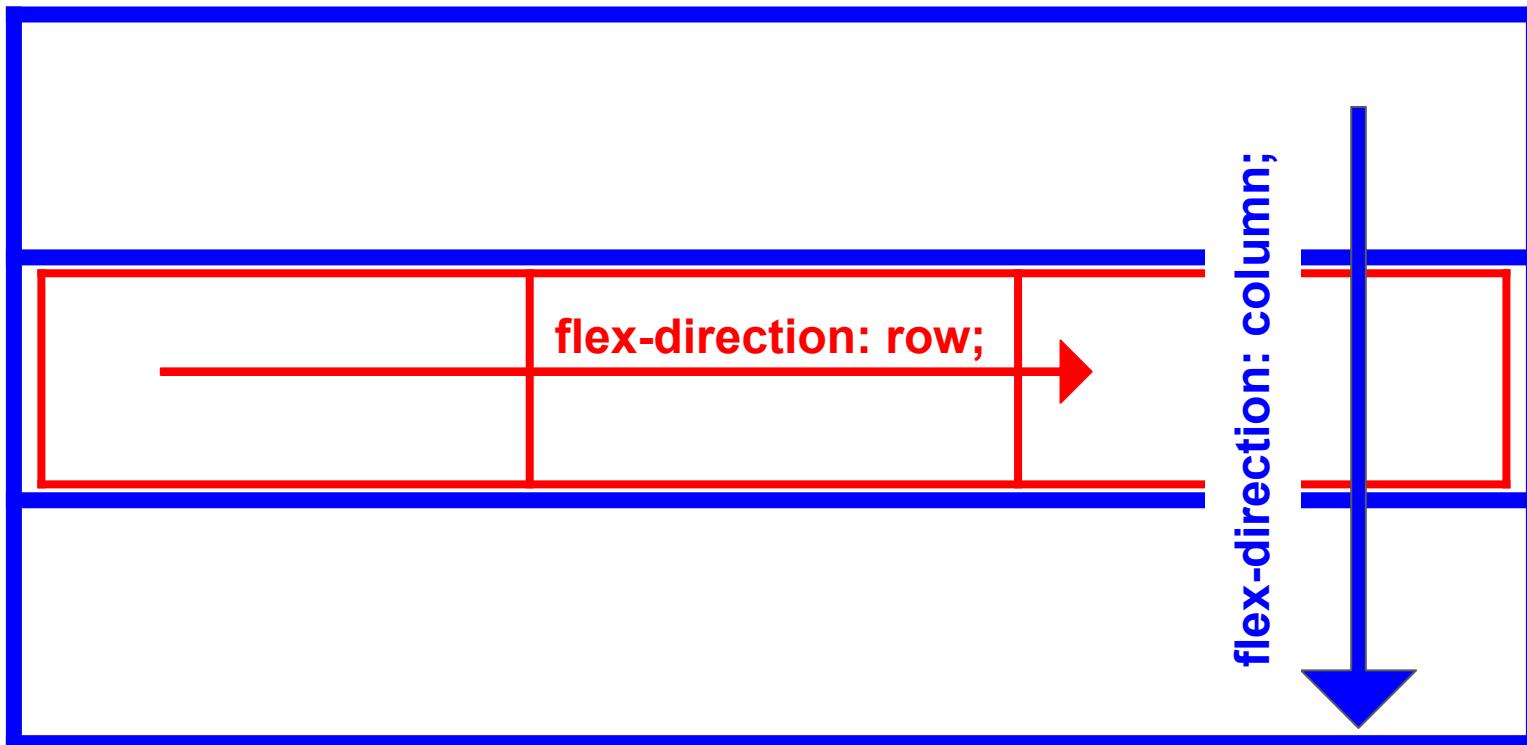


6. Flexible Box Layout und 7. CSS Grid Layout

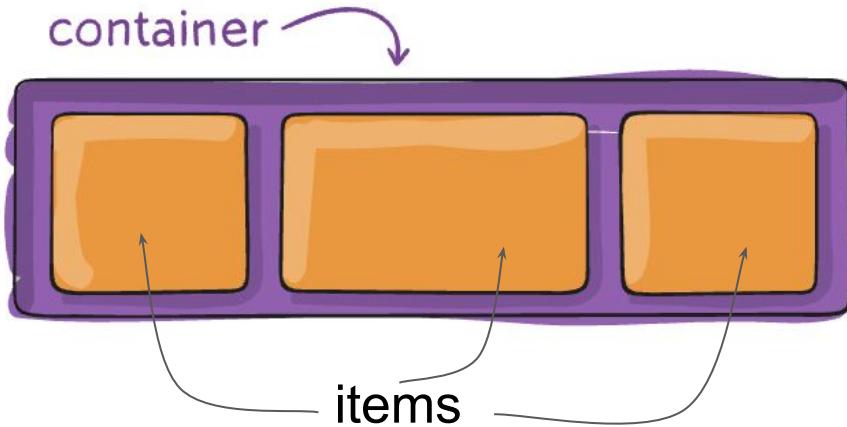
	Flexbox	Grid
CSS-Regel	<code>display: flex;</code>	<code>display: grid;</code>
Strategie Orientierung	content-out : erst Content, dann mit Regeln Verteilung festlegen	layout-in : erst Grid-Layout, dann Content den Zellen zuordnen (Mapping)
Dimensionalität	eher 1-dimensional	eher 2-dimensional

<https://medium.com/js-imaginea/designing-a-layout-using-css-grid-flexbox-10df75a7f590>

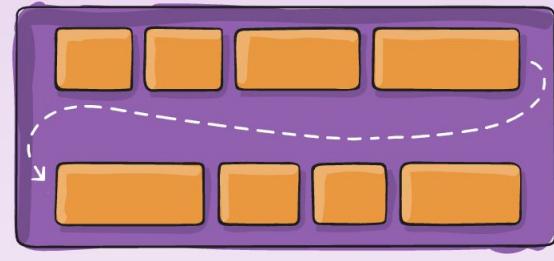
Mehrdimensional ⇒ Nested Flexbox



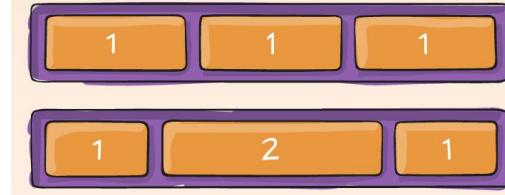
Flexbox Grundbegriffe



flex-wrap

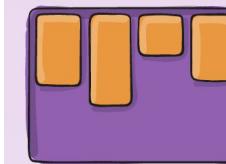


flex-grow

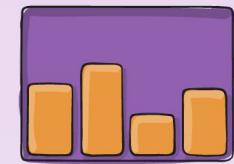


align-items

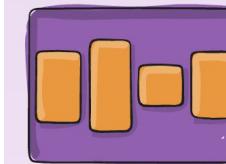
flex-start



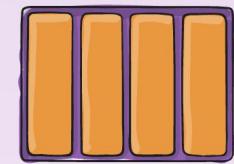
flex-end



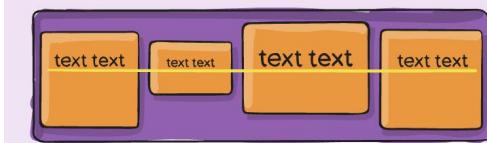
center



stretch

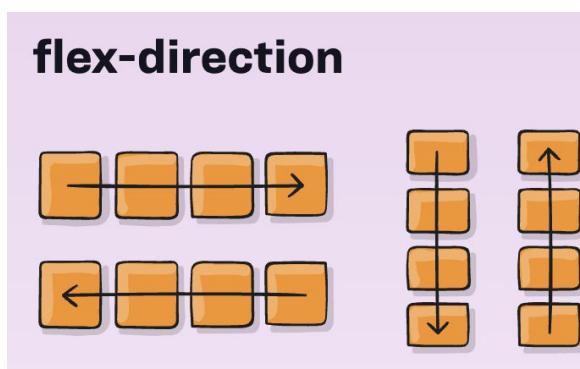


baseline



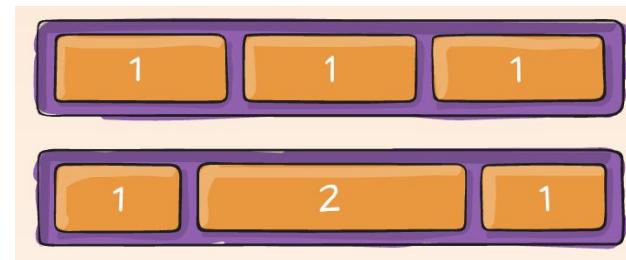
Flex Container Properties

- flex-direction
- flex-wrap
- flex-flow
- justify-content
- align-items
- align-content



Flex Item Properties

- order
- flex-grow
- flex-shrink
- flex-basis
- flex
- align-self

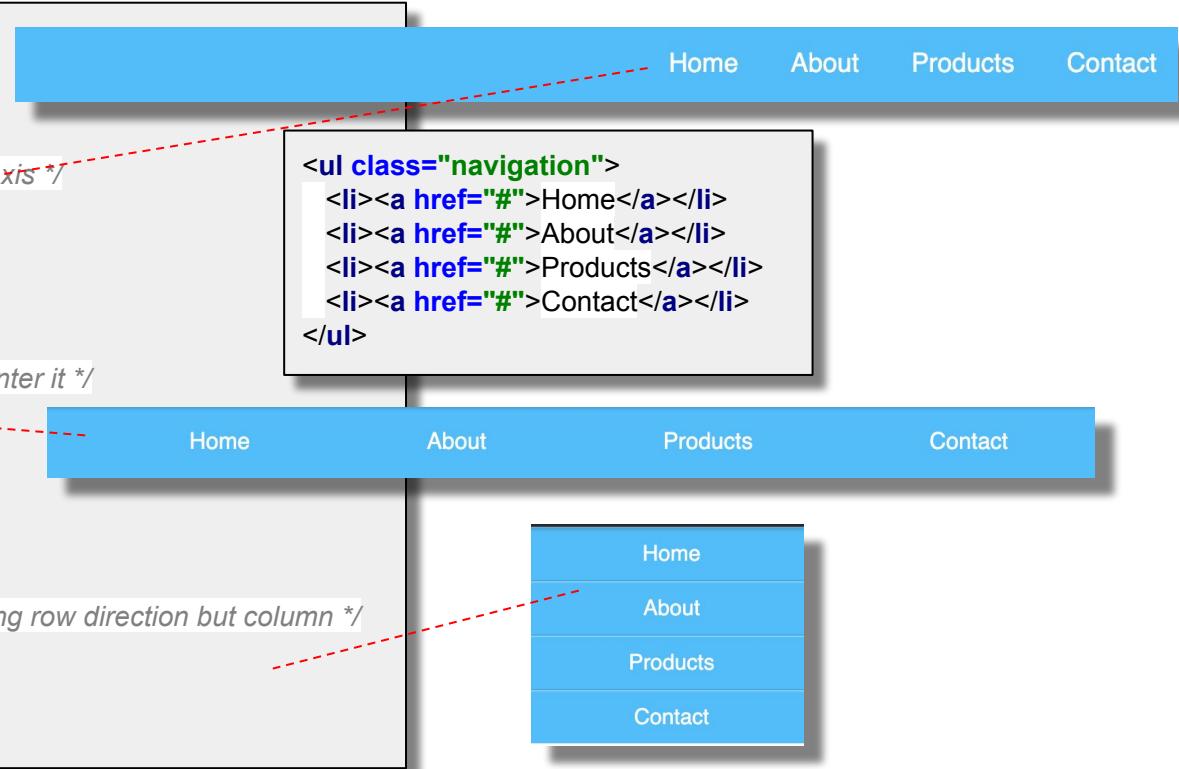


Beispiel: RWD Menu with Flexbox

```
/* 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;
  }
}
```

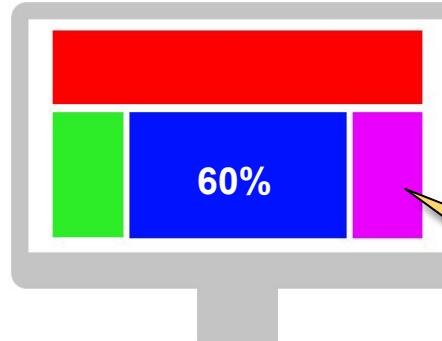


Mit Flexbox die Reihenfolge tauschen

```
* {  
  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; }  
}
```

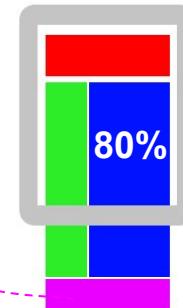
order

Dann kann man mit ganzen
Prozent-Zahlen arbeiten



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

rot - grün - blau - pink



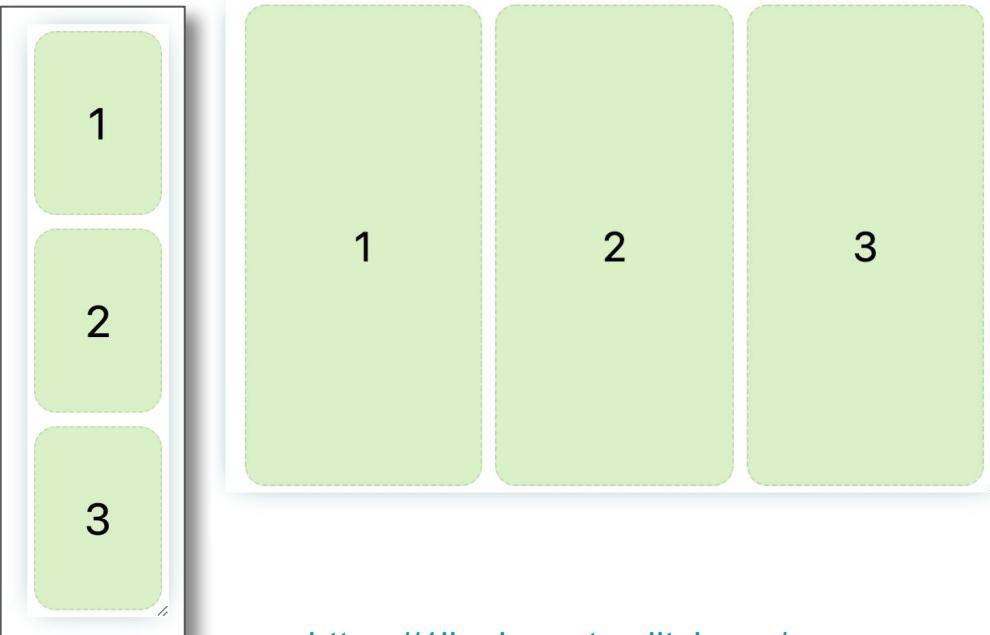
rot - blau - grün - pink

flex: <flex-grow> <flex-shrink> <flex-basis>

flex: 1 1 150px

02. The Deconstructed Pancake

flex: 0 1 <baseWidth>



Current Browser Support:

Edge Firefox Chrome Safari

HTML

```
<div class="parent">
  <div class="box">1</div>
  <div class="box">2</div>
  <div class="box">3</div>
</div>
```

CSS

```
.ex2 .parent {
  display: flex;
  flex-wrap: wrap;
  justify-content: center;
}

.ex2 .box {
  flex: 1 1 150px; /* Stretching: */
  flex: 0 1 150px; /* No stretching: */
  margin: 5px;
}
```

[Explore on CodePen](#)

<https://1linelayouts.glitch.me/>

<https://youtu.be/qm0lfG1GyZU?t=257>

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 */

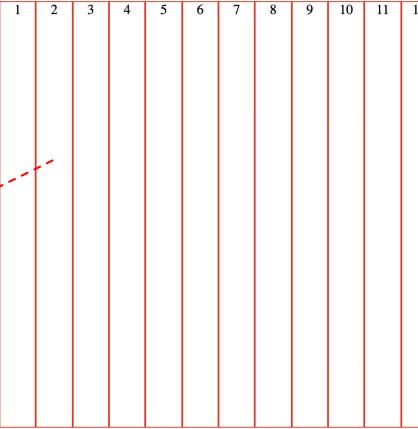
  flex-basis: calc(var(--width) / var(--columns) * 100% - 2px);
}

.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



```
<body>


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


```

ab 360px



ab 520px



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>

Firefox Flexbox Inspector

The screenshot shows the Firefox Developer Tools interface with the 'Page Inspector' tab selected. The main content area displays a web page with a red header, a green sidebar, a blue main content area, and a pink footer. The 'Layout' tab is active in the inspector panel, which contains several sections:

- Stile filteren**: Shows styles for ':hov .cls'.
- Element**: Shows styles for 'body' and '*'. The 'body' style includes 'display: flex; flex-wrap: wrap; margin: 0;'. The '*' style includes 'box-sizing: border-box;'.
- Flex-Container**: Shows the 'body' element as a flex container with 'flex-direction: row; flex-wrap: wrap; align-items: flex-start;'. It also lists child elements: 'nav' (rank 1), 'aside.left' (rank 2), 'article' (rank 3), and 'aside.right' (rank 4).
- Raster**: A note stating 'Es wird kein CSS-Raster auf dieser Seite verwendet.'
- Box-Modell**: A note stating 'Die Box-Modell-Eigenschaften sind hier nicht dargestellt.'

To the right of the main content area, there is a separate panel titled 'Flex Item of ul.card-cont' with the following details:

final	basis
Content Size	794.65px
Flexibility (flex-shrink: 1)	Item was set to shrink. -664.17px
Minimum Size	145.08px The item was clamped to its minimum size.
Final Size	145.08px

w3schools Tutorial zum Flexbox-Layout

The screenshot shows a web browser window displaying the w3schools CSS Flexbox tutorial. The top navigation bar includes links for Home, HTML, CSS (which is highlighted in green), JavaScript, SQL, PHP, Bootstrap, How To, jQuery, W3.CSS, and Angular. A sidebar on the left lists various CSS topics, with 'CSS Flexbox' also highlighted in green. The main content area has a title 'The flex-wrap Property' and a brief description of what it does. It includes a visual example showing twelve numbered boxes (1-12) arranged in two rows of six within a blue-bordered container. Below this is an 'Example' section with code for a '.flex-container' class and a 'Try it Yourself' button.

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



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.

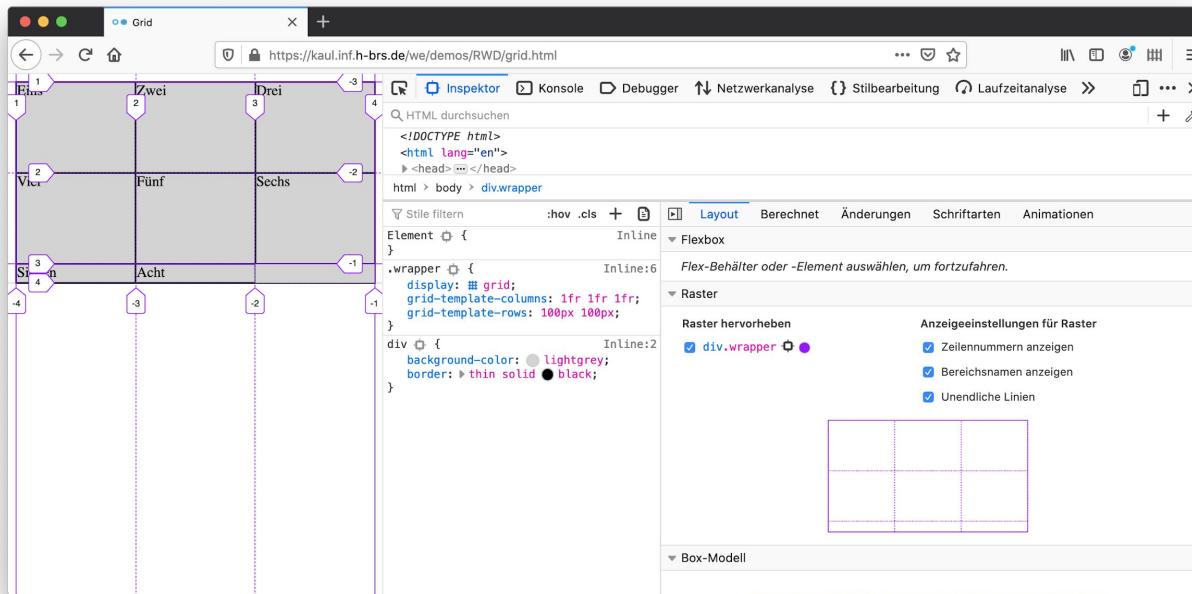
Example

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';  
}
```

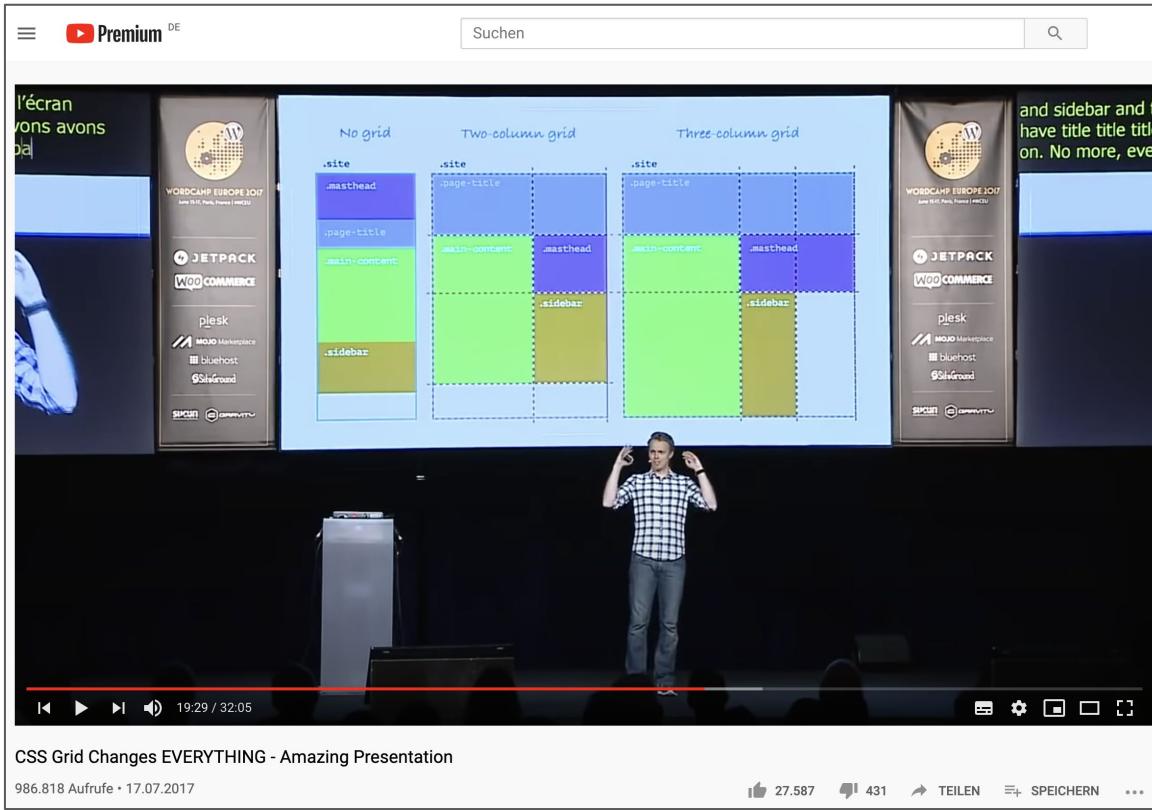
[Try it Yourself »](#)

Firefox Grid Inspector

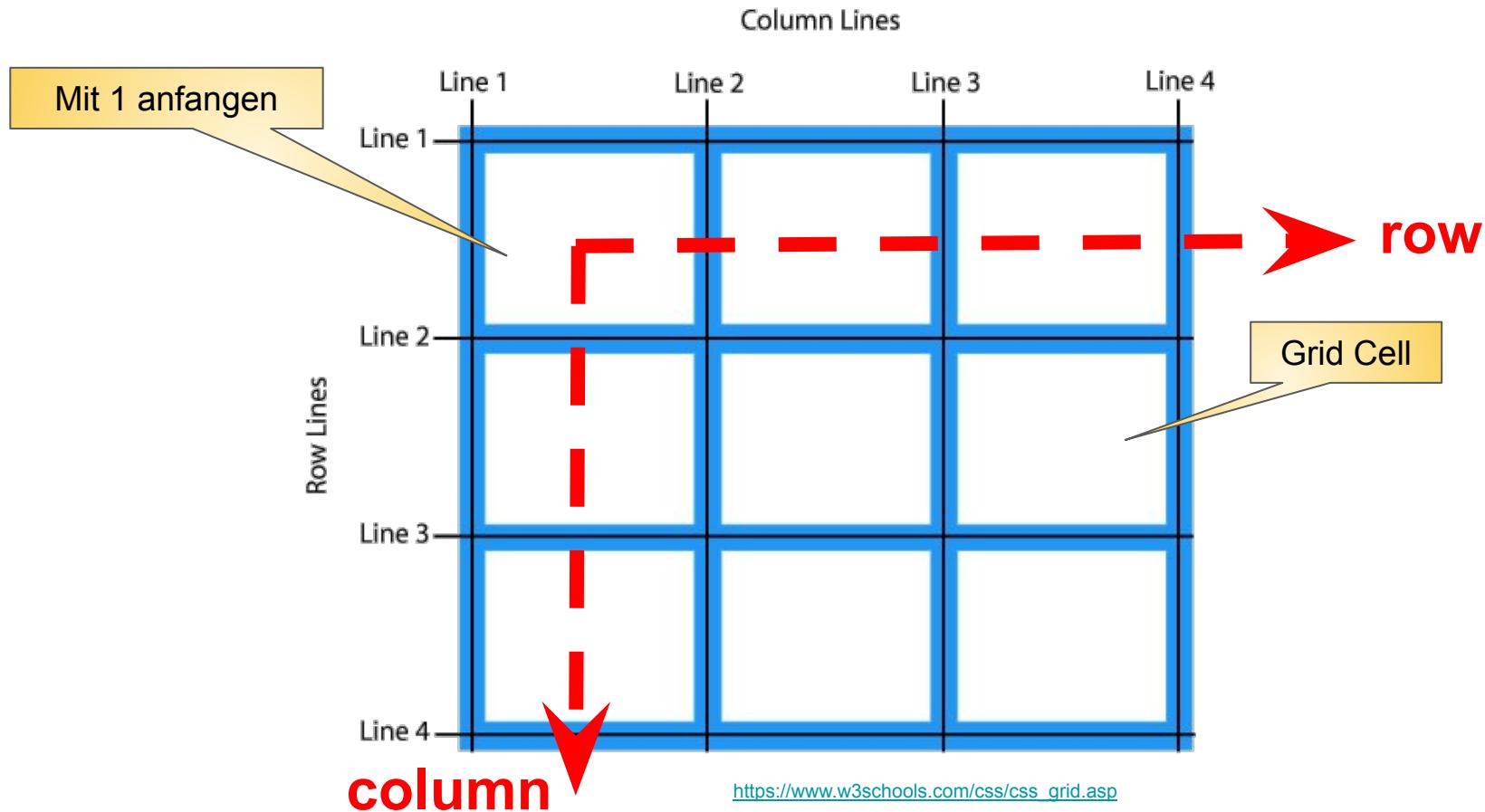


https://developer.mozilla.org/de/docs/Tools/Page_Inspector/How_to/Raster_LayoutUntersuchen

CSS Grid changes EVERYTHING



Grid-Terminologie



Grid Container Properties

- display: grid
- grid-template-columns
- grid-template-rows
- grid-template-areas
- grid-template
- grid-column-gap
- grid-row-gap
- grid-gap
- justify-items
- align-items
- place-items
- justify-content
- align-content
- place-content
- grid-auto-columns
- grid-auto-rows
- grid-auto-flow
- grid

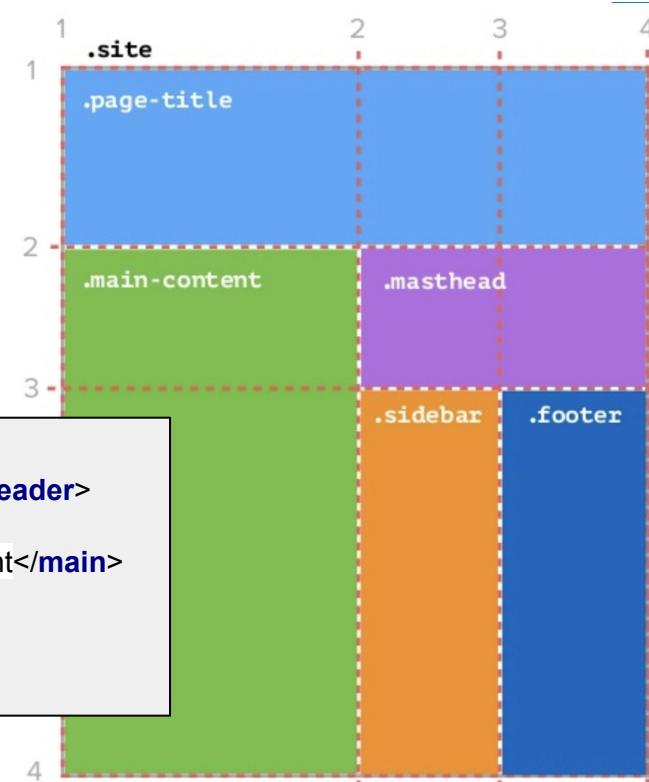
Grid Item Properties

- grid-column-start
- grid-column-end
- grid-row-start
- grid-row-end
- grid-column
- grid-row
- grid-area
- justify-self
- align-self
- place-self

Grid mit "Anfang / Ende-Notation"

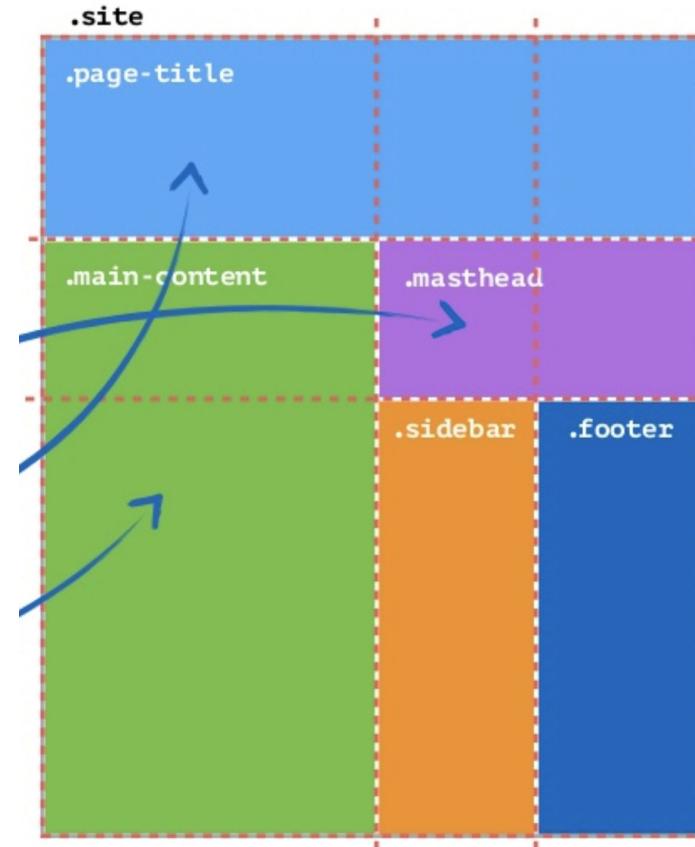
```
<style>
div * {
  border: thin dashed red;
  min-height: 10vh;
}
.site {
  display: grid;
  grid-template-columns: 2fr 1fr 1fr;
  grid-template-rows: auto 1fr 3fr;
}
.masthead {
  grid-column: 2/4;
  grid-row: 2/3;
}
.page-title {
  grid-column: 1/4;
  grid-row: 1/2;
}
.main-content {
  grid-column: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



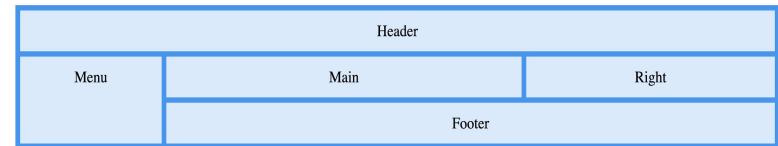
Positionierung eines Items im Grid

```
.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.

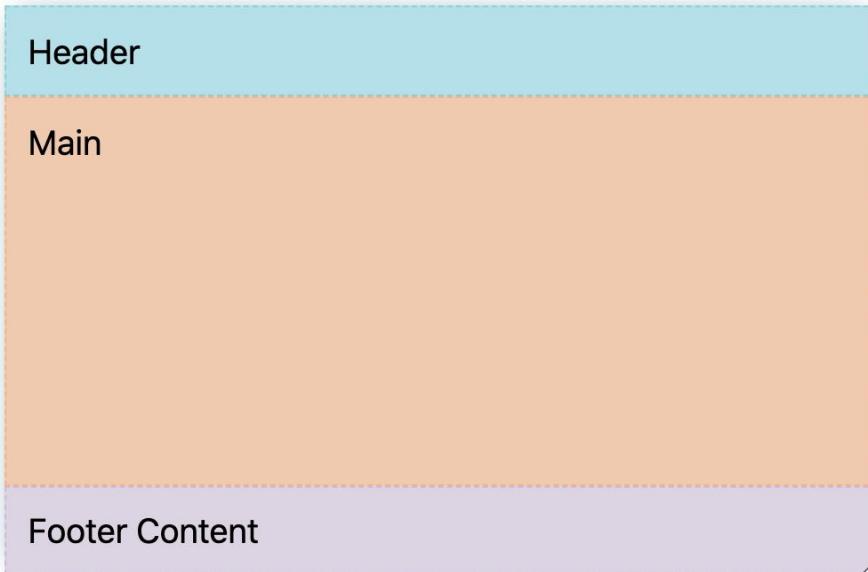


```
.item1 { grid-area: header; }  
.item2 { grid-area: menu; }  
.item3 { grid-area: main; }  
.item4 { grid-area: right; }  
.item5 { grid-area: footer; }  
  
.grid-container {  
    grid-template-areas:  
        'header header header header header header'  
        'menu main main main right right'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

Pancake Stack Layout with Grid "auto 1fr auto"

04. Pancake Stack

`grid-template-rows: auto 1fr auto`



```
.ex4 .parent {  
  display: grid;  
  grid-template-rows: auto 1fr auto;  
}
```

HTML

CSS

Current Browser Support:

Edge

Firefox

Chrome

Safari

[Explore on CodePen](#)

<https://1linelayouts.glitch.me/>

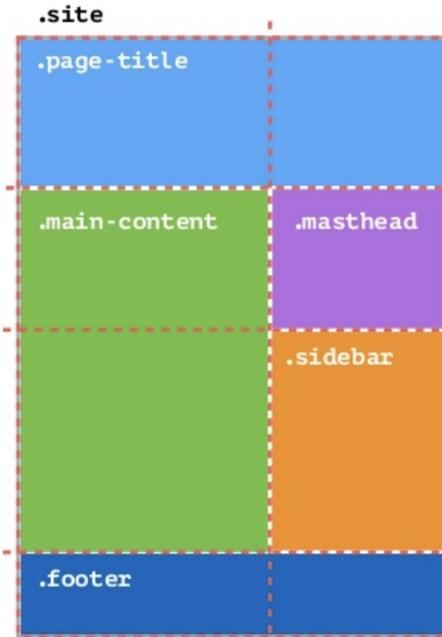
<https://youtu.be/qm0IfG1GyZU?t=414>

RWD mit Grid

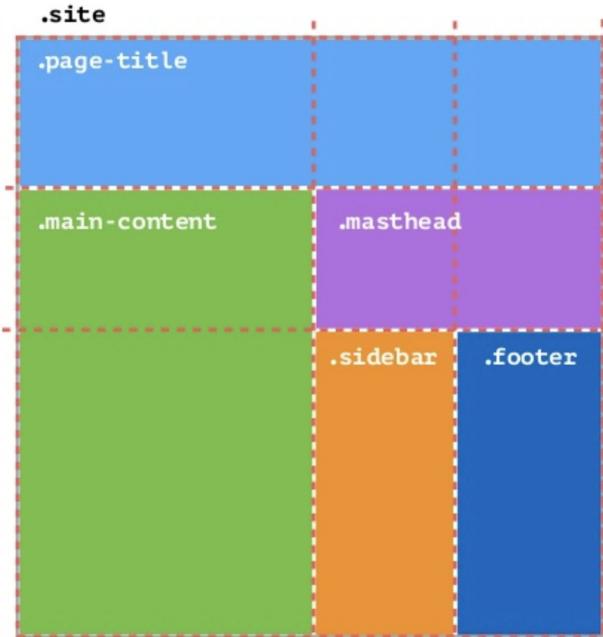
```
.site {  
    display: grid;  
    grid-template-columns: 1fr 1fr;  
    grid-template-rows: auto 1fr 3fr 1fr;  
    grid-template-areas:  
        "title title"  
        "main header"  
        "main sidebar"  
        "footer footer";  
}  
  
.masthead { grid-area: header; }  
.page-title { grid-area: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

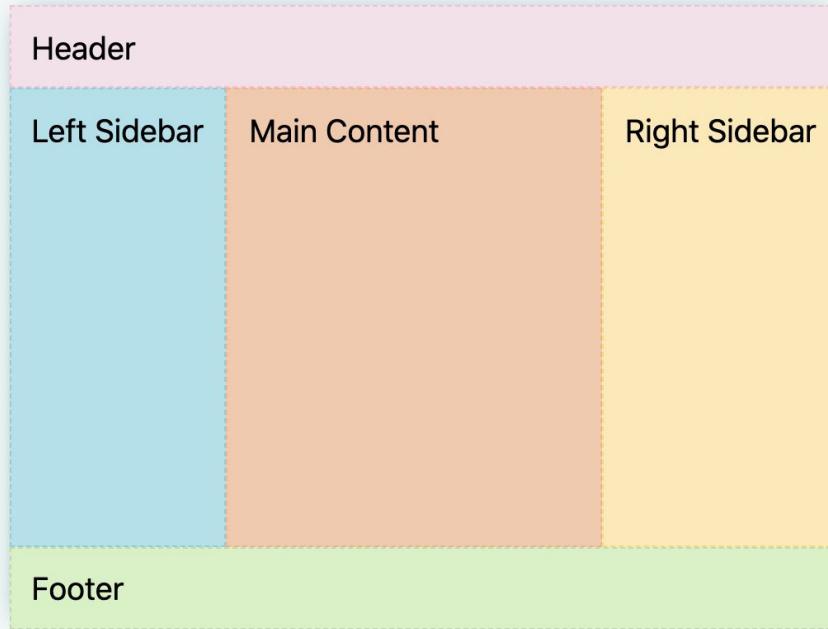


<https://www.slideshare.net/mor10/css-grid-changes-everything-about-web-layouts-wordcamp-europe-2017/42>

RWD Holy Grail Layout with Grid Template

05. Classic Holy Grail Layout

```
grid-template: auto 1fr auto / auto 1fr auto
```



Current Browser Support:

Edge Firefox Chrome Safari

HTML

CSS

```
.ex5 .parent {  
  display: grid;  
  grid-template: auto 1fr auto / auto 1fr  
  auto;  
}  
  
.ex5 header {  
  padding: 2rem;  
  grid-column: 1 / 4;  
}  
  
.ex5 left-side {
```

[Explore on CodePen](#)

<https://1linelayouts.glitch.me/>

<https://youtu.be/qm0IfG1GyZU?t=494>

```

.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";
  }
}

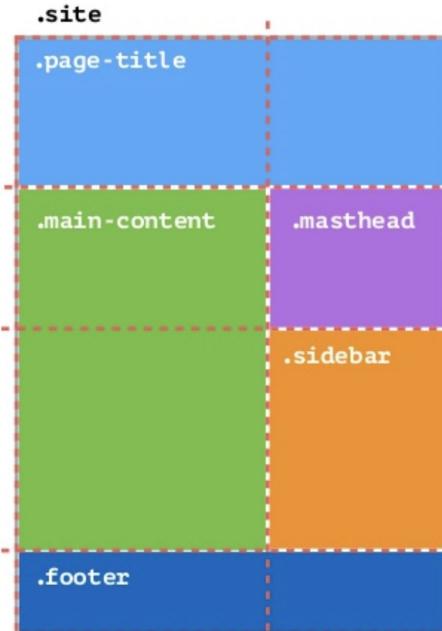
```

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>

Erzeugen von Grids mit **repeat(anzahl, wert)**

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

```
/* <track-repeat> values */  
repeat(4, 1fr)  
repeat(4, [col-start] 250px [col-end])  
repeat(4, [col-start] 60% [col-end])  
repeat(4, [col-start] 1fr [col-end])  
repeat(4, [col-start] min-content [col-end])  
repeat(4, [col-start] max-content [col-end])  
repeat(4, [col-start] auto [col-end])  
repeat(4, [col-start] minmax(100px, 1fr) [col-end])  
repeat(4, [col-start] fit-content(200px) [col-end])  
repeat(4, 10px [col-start] 30% [col-middle] auto [col-end])  
repeat(4, [col-start] min-content [col-middle] max-content [col-end])
```

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, Code Snippets, CSS, Grid, 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

CSS Grid playground
A PEN BY Morten Rand-Hendriksen

Fork Change View Log In Sign Up

CSS Grid Playground

Get to know the CSS Grid Layout Module

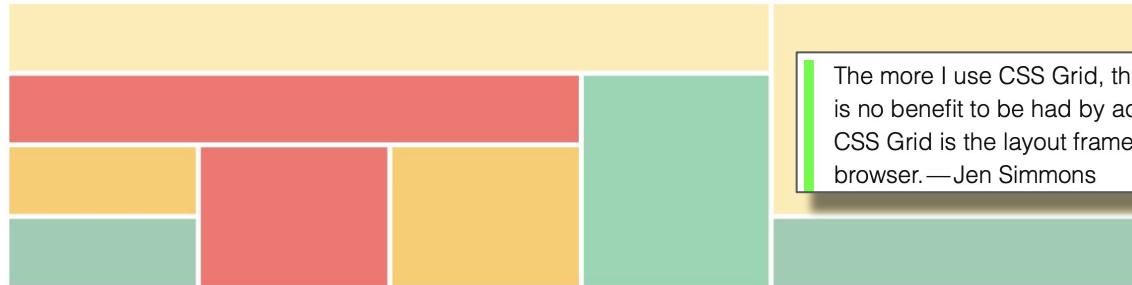
Box 1	Box 2	Box 3
Box 4	Box 5	Box 7

Vergleich von Grid mit Bootstrap

Why CSS Grid is better than Bootstrap for creating layouts

October 14th 2019

 TWEET THIS



The more I use CSS Grid, the more convinced I am that there is no benefit to be had by adding a layer of abstraction over it. CSS Grid is the layout framework. Baked right into the browser.—Jen Simmons

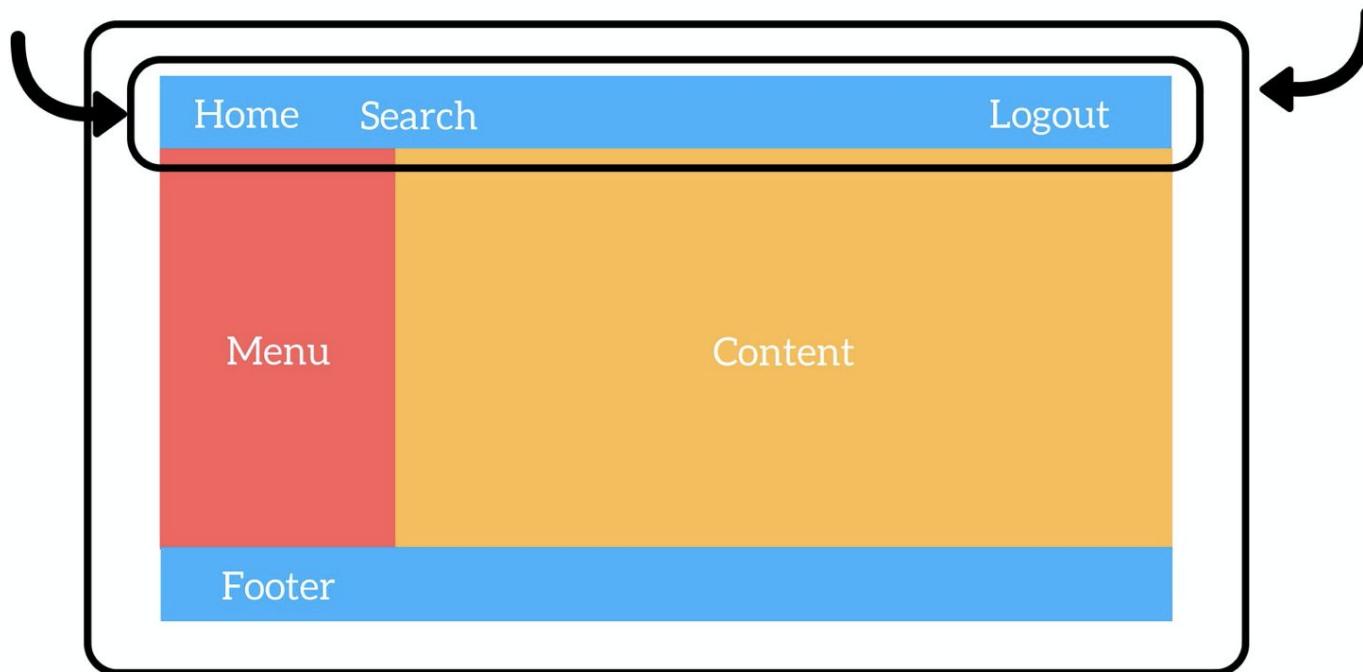
CSS Grid is a new way of creating layouts on the web. For the first time ever we have a proper layout system available natively in the browser, which gives us a ton of benefits.

<https://hackernoon.com/how-css-grid-beats-bootstrap-85d5881cf163>

Kombinationen aus Flexbox und Grid

Flexbox Container

Grid Container



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

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>
```

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

Image lazy loading

```

```



A screenshot of a browser window showing two images of a kitten. The top image is fully loaded and displayed. The bottom image is partially visible and has a 'loading' placeholder.

QuickTime Player File Edit View Window Help

mathiessyns.be/demo/img-loading-lazy

Network Performance

Filter XHR JS CSS Img Media Font Doc WS Manifest Other

2000 ms 4000 ms 6000 ms 8000 ms 10000 ms 12000 ms 14000 ms 16000 ms 18000 ms

Name	Status	Size	Waterfall
437	200	13.0 KB	
438	200	17.7 KB	
439	200	14.4 KB	
440	200	25.8 KB	
441	200	16.6 KB	
442	200	17.9 KB	
443	200	25.5 KB	
444	200	19.9 KB	
445	200	14.6 KB	
446	200	13.2 KB	
447	200	16.6 KB	
448	200	16.6 KB	
449	200	14.4 KB	
450	200	24.5 KB	
451	200	13.7 KB	
452	200	17.2 KB	
453	200	24.6 KB	
454	200	26.7 KB	
455	200	24.9 KB	
456	200	24.7 KB	
457	200	14.0 KB	
458	200	15.9 KB	
459	200	23.1 KB	

46 requests 833 KB transferred 829 KB resources

<https://web.dev/browser-level-image-lazy-loading/>

Lazy loading via attribute for images & iframes

LS

The `loading` attributing on images & iframes gives authors control over when the browser should start loading the resource.



<https://caniuse.com/loading-lazy-attr>

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.ogg" 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

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 Web Design?



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

Responsive web design basics

How to create sites which respond to the needs and capabilities of the device they are viewed on.

Feb 12, 2019 • Updated May 14, 2020



Pete LePage

[Twitter](#) · [GitHub](#) · [Glitch](#) · [Blog](#)



Rachel Andrew

[Twitter](#) · [GitHub](#) · [Glitch](#) · [Blog](#)

1. [Set the viewport](#)
2. [Size content to the viewport](#)
3. [Use CSS media queries for responsiveness](#)
4. [How to choose breakpoints](#)
5. [View media query breakpoints in Chrome DevTools](#)

<https://web.dev/responsive-web-design-basics/>

Zusammenfassung und Ausblick

But the core **technical** implementation isn't particularly complex:

1. Stay **fluid**;
2. use some **@media queries** to restyle things as needed.

The bigger deal in the last decade was the **impact on businesses**. Adjusting workflows to accommodate this style of thinking. Combining teams of developers who used to work on entirely different codebases now working on a single codebase. The impact at organizations wasn't nearly as straightforward as the technology of it all.

<https://css-tricks.com/responsive-web-design-turns-ten/>