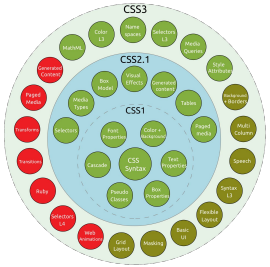
Responsive Web Design (RWD)

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?

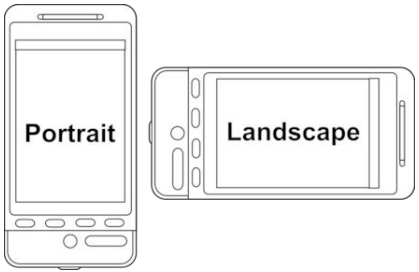
http://entnemdept.ufl.edu/walker/pdfvhtml.htm

Definition Responsive Webdesign (RWD)

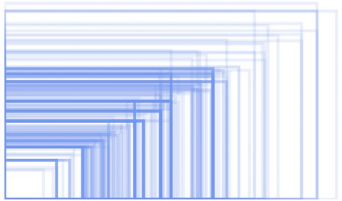
- Unter einer responsiven Gestaltung einer Webseite versteht man, dass sich das Layout der Seite an das Ausgabemedium selbstständig (möglichst nahtlos, fluide) anpasst.
- Der grafische Aufbau einer „responsiven“ Webseite folgt den Anforderungen des jeweiligen Gerätes dynamisch.

SmartWatch

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

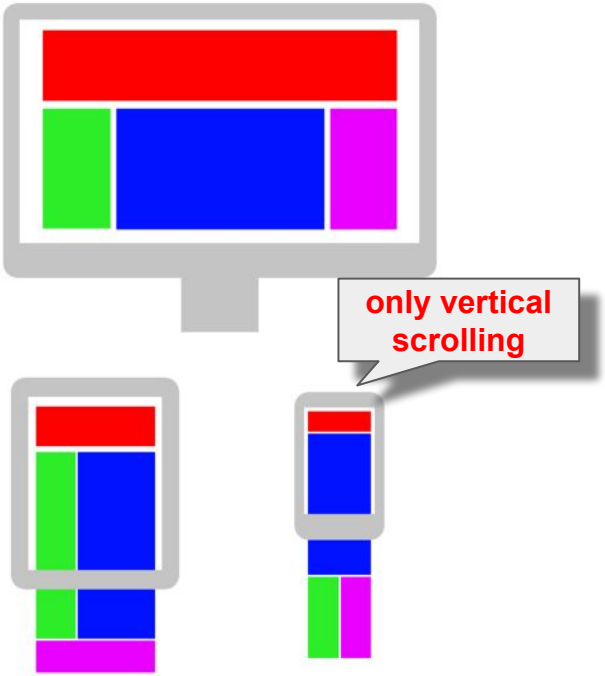


https://en.wikipedia.org/wiki/Page_orientation



Viewportauflösungen heutiger Geräte

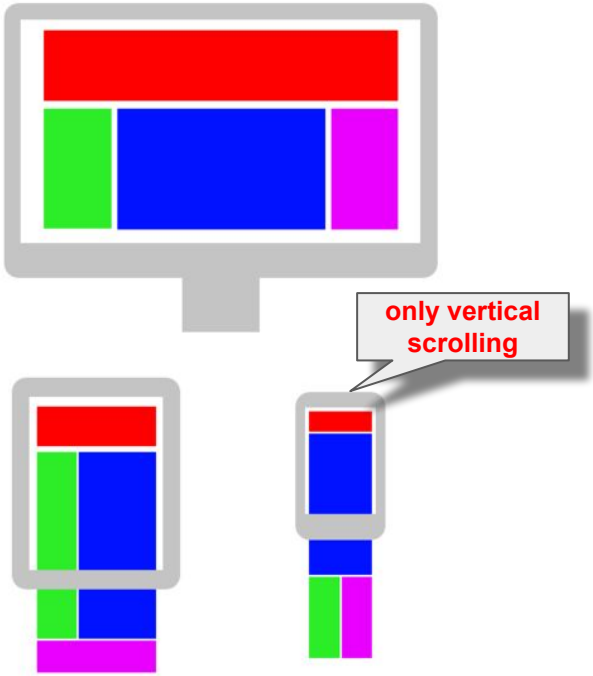
https://opensignal.com/reports/fragmentation.php



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

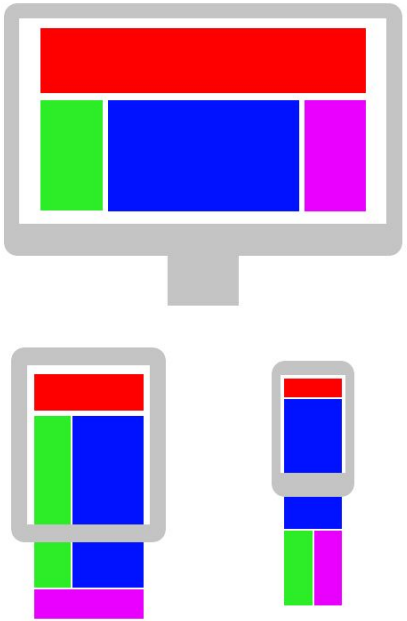


https://de.wikipedia.org/wiki/Responsive_Webdesign

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

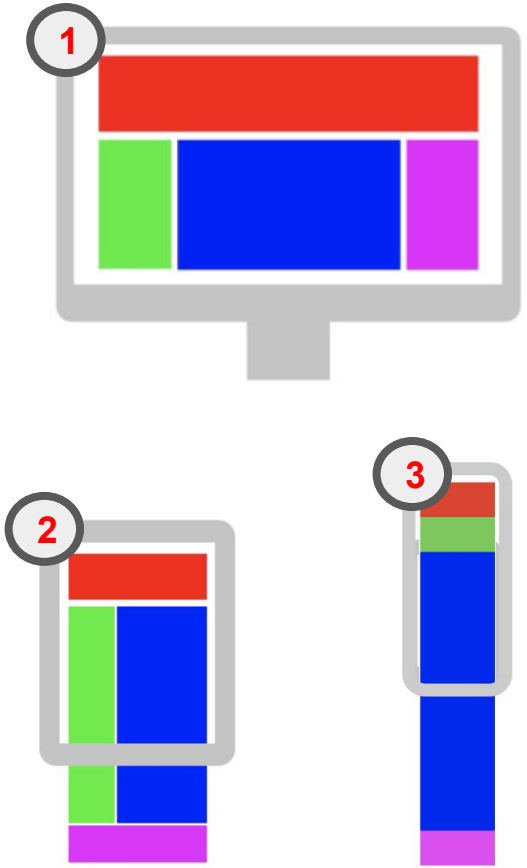
Zwei Methoden für Responsive Design

1. Desktop First
2. Mobile First



Desktop First

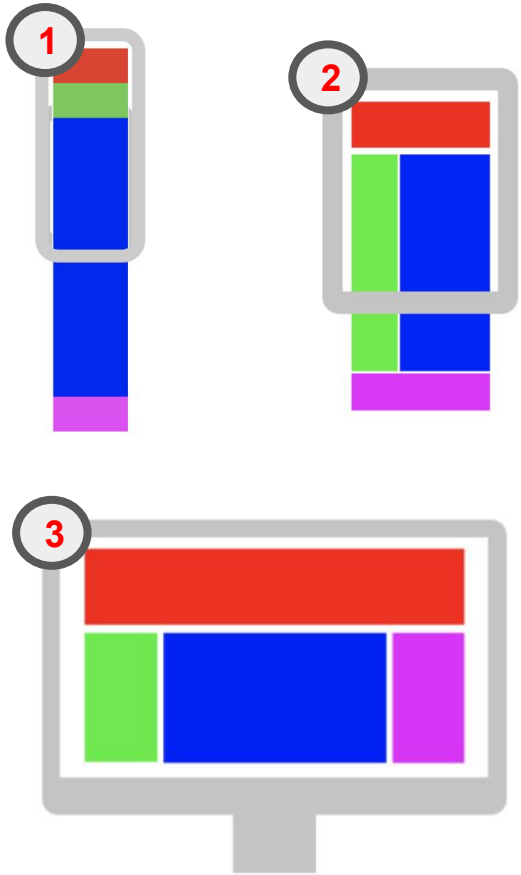
- Erstellen Sie die Seite,
- schieben Sie Ihr Browserfenster zusammen
- Fügen Sie Layout-Regeln für kleinere Viewports hinzu



https://wiki.selfhtml.org/wiki/CSS/Tutorials/Einstieg/Media_Queries#Mobile_first

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



https://wiki.selfhtml.org/wiki/CSS/Tutorials/Einstieg/Media_Queries#Mobile_first

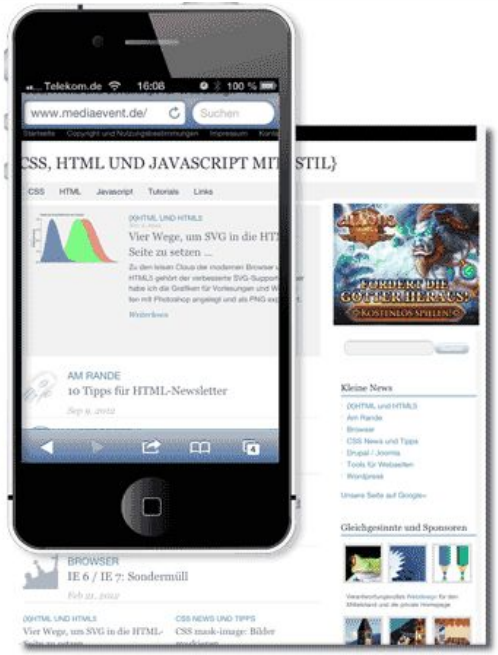
Technische Realisierung von RWD mit CSS

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 comes 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

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

CSS

JavaScript

```html
<!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

screen
print
speech
all

min-width
max-width
min-height
max-height
aspect-ratio
orientation

and
or
not

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

Beispiele Media Queries

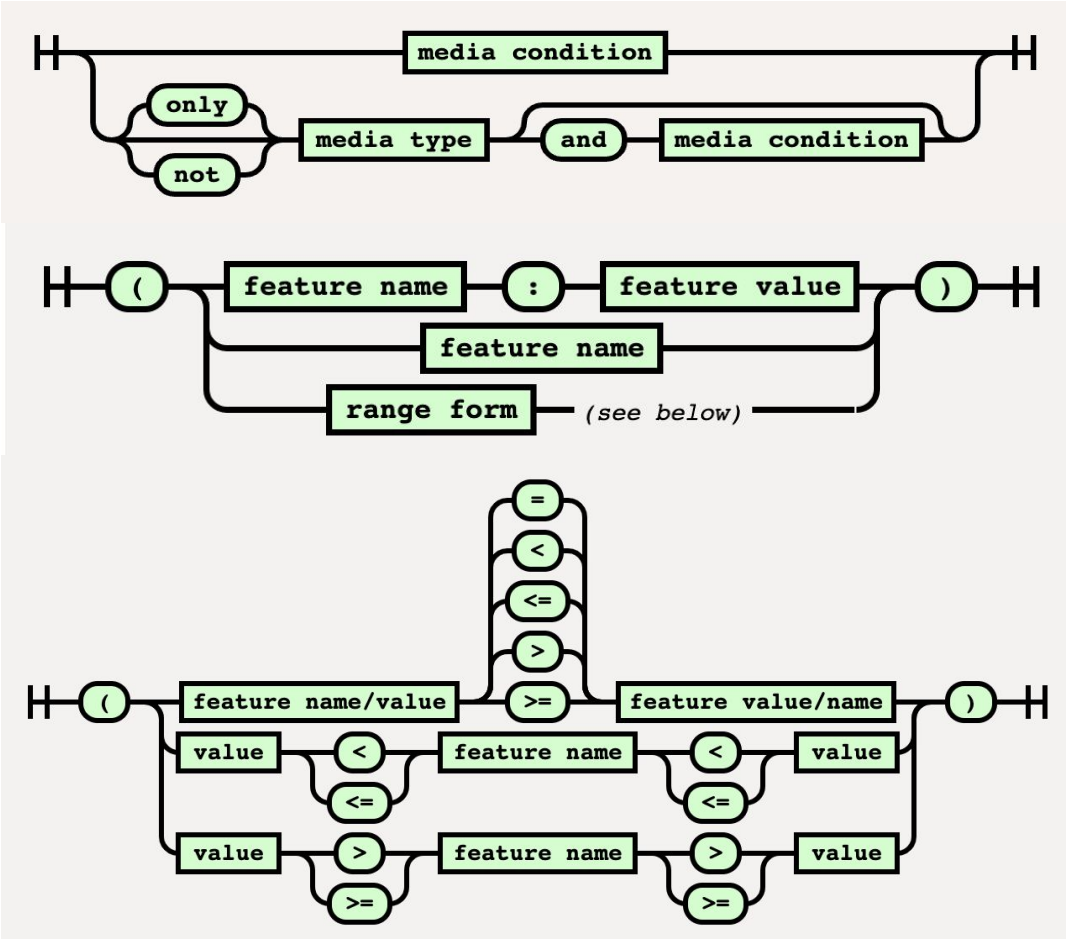
```
@media print {
  * {
    background-color: white;
    font-family: "times new roman", times, serif;
    text-align: justify;
  }
}
@media screen and (max-width: 220px){ }
@media (min-width: 30em) and (orientation: landscape) { }
@media screen and (min-width: 30em) and (orientation: landscape) { }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

Druck mit Serifen, Bildschirm ohne

https://wiki.selfhtml.org/wiki/CSS/Media_Queries

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) { }
```

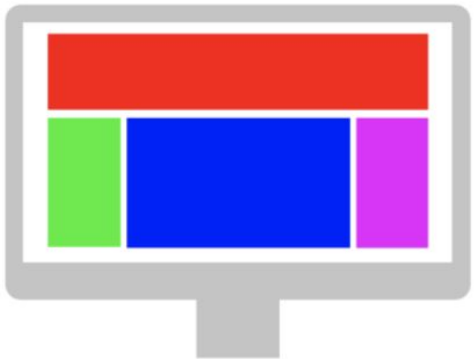


https://drafts.csswg.org/mediaqueries-4/#mq-features

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;
  }
}
```

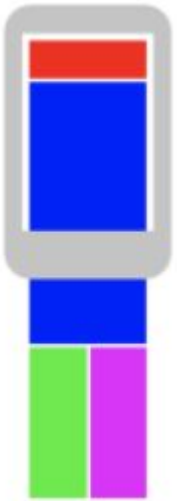


https://wiki.selfhtml.org/wiki/CSS/Tutorials/Einstieg/Media_Queries#Desktop_first

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%;
  }
}
```



https://wiki.selfhtml.org/wiki/CSS/Tutorials/Einstieg/Media_Queries#Mobile_first

3. Viewport Units for RWD

CSS-Maßeinheiten (*CSS Units*)

Relative CSS-Maßeinheiten

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

relativ zum Viewport

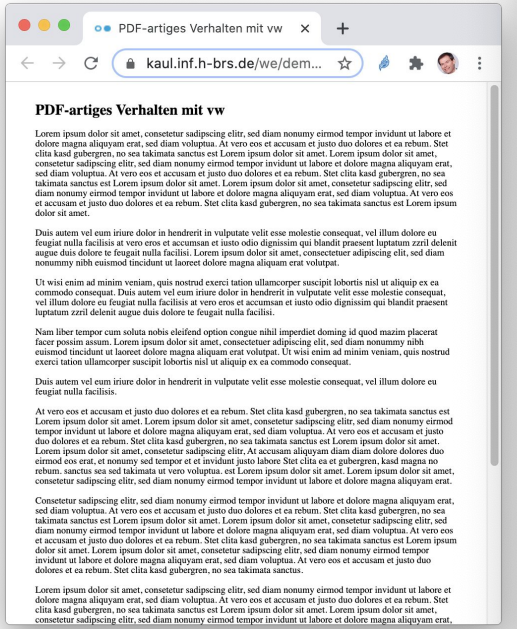
Absolute CSS-Maßeinheiten

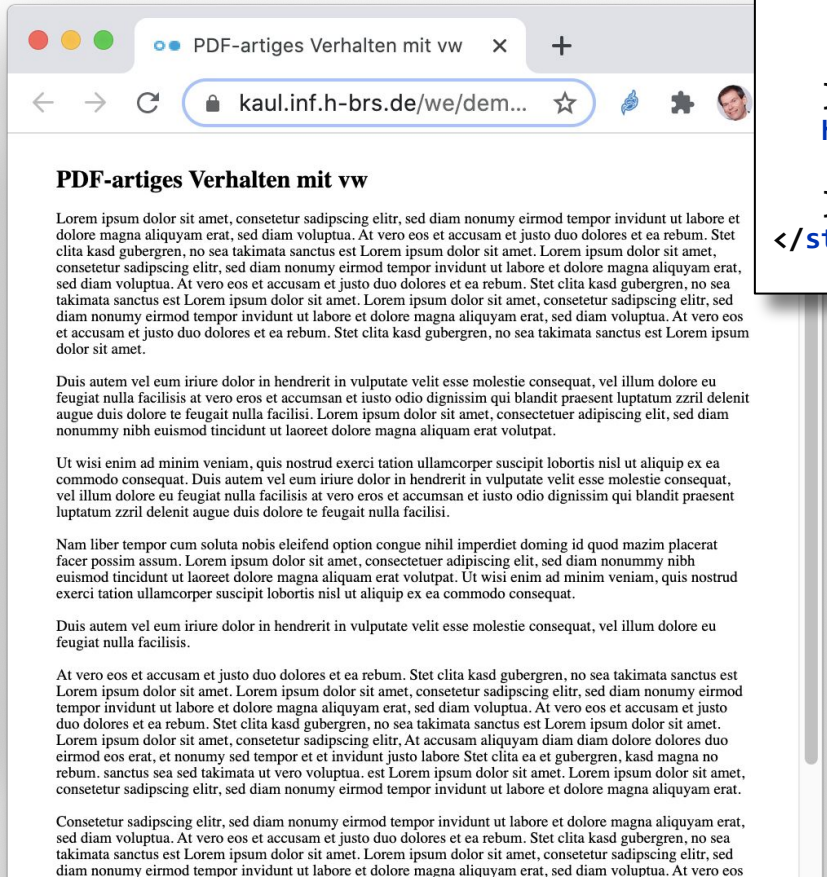
cm, mm, in,
px, pt, pc

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

Variable Font Size mit vw

```
<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  }
```

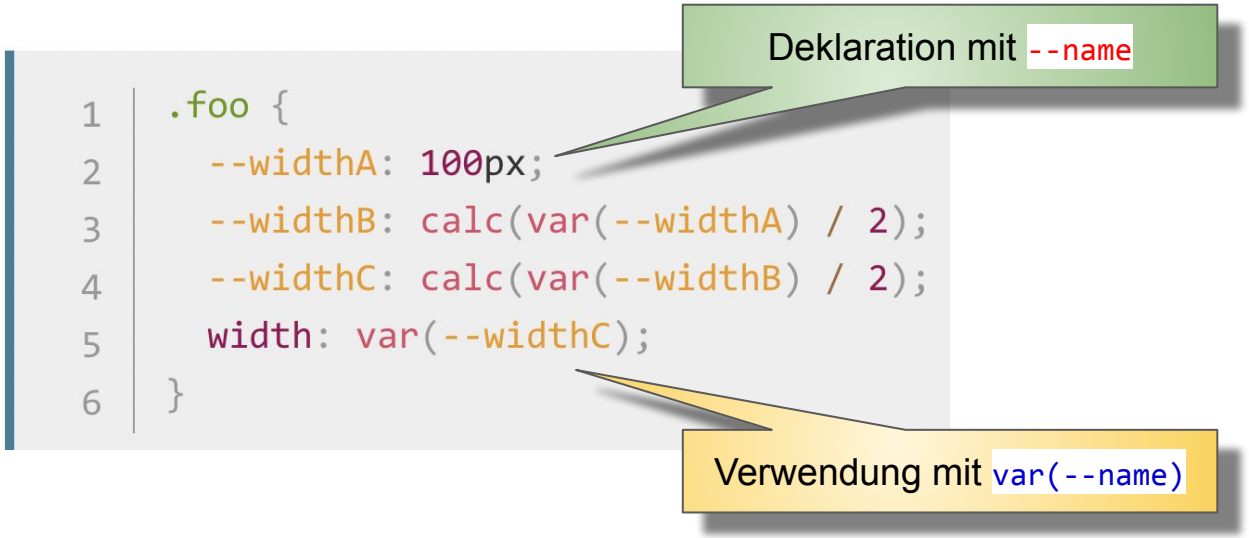
This ensures that text size will scale if the page is zoomed.

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

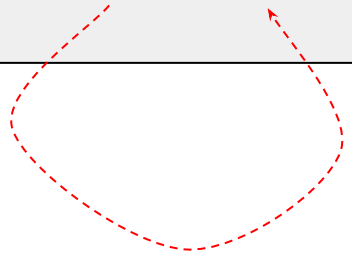
5. CSS Custom Properties (*aka* CSS Variables)



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 */
}
```

https://www.smashingmagazine.com/2017/04/start-using-css-custom-properties/

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>
```

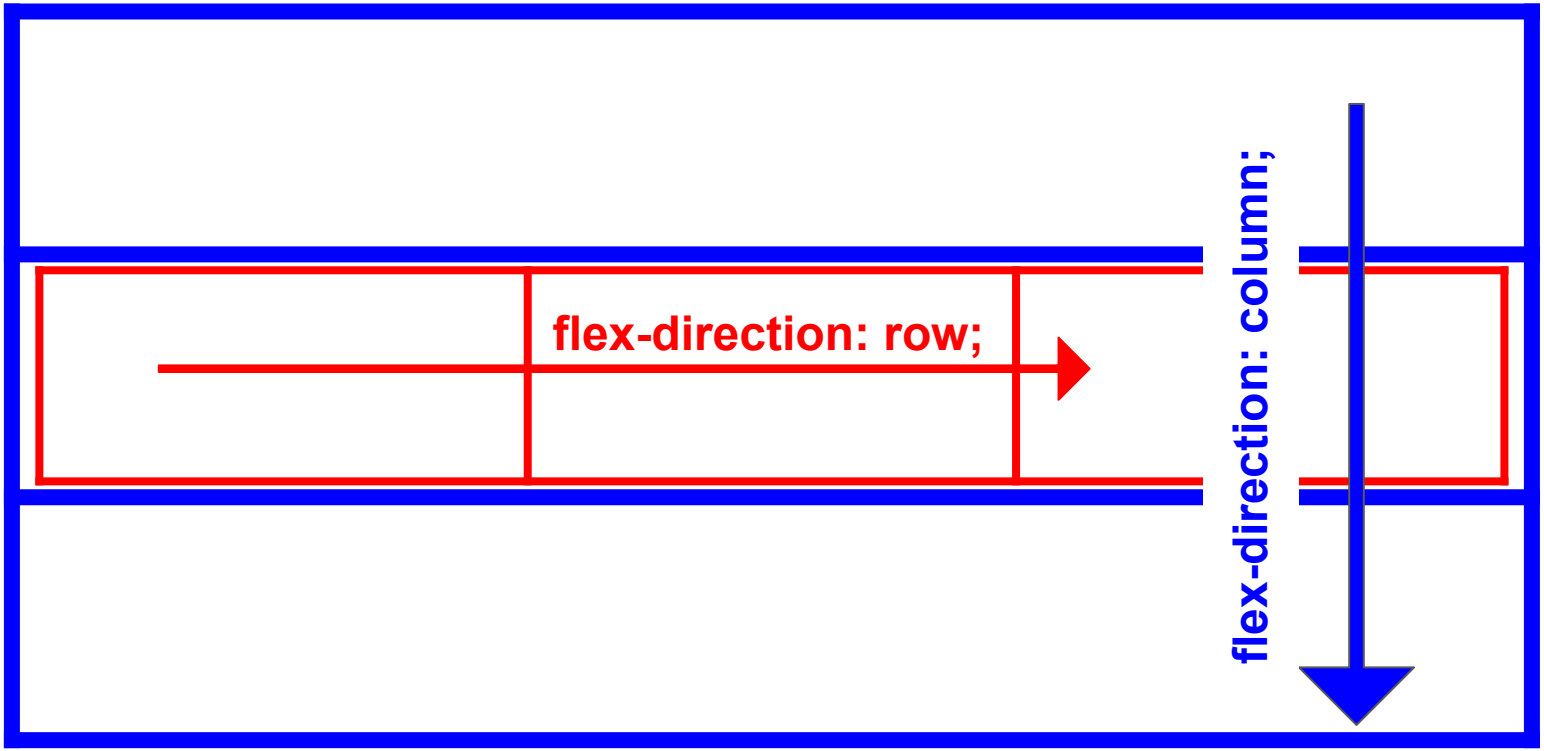
https://css-tricks.com/responsive-designs-and-css-custom-properties-defining-variables-and-breakpoints/

6. Flexible Box Layout und 7. CSS Grid Layout

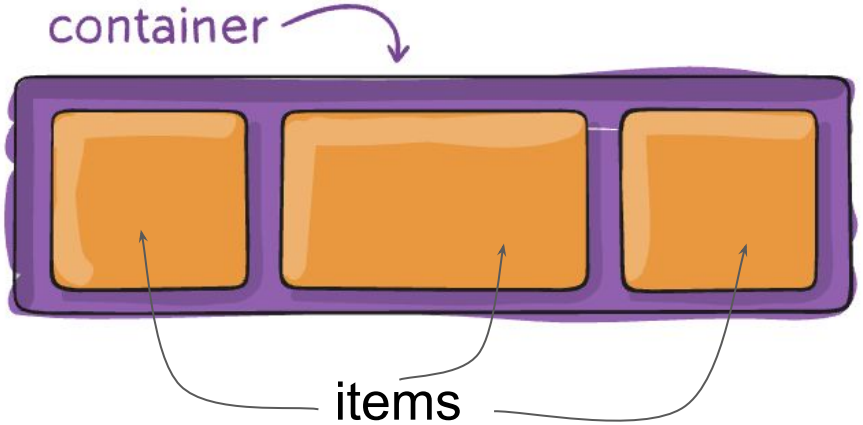
	Flexbox	**Grid**
CSS-Regel	**display: flex;**	**display: grid;**
Strategie Orientierung	**content-out**: erst Content, dann mit Regeln Verteilung festlegen	**layout-in**: erst Grid-Layout, dann Content den Zellen zuordnen (Mapping)
Dimensionalität	eher 1-dimensional	eher 2-dimensional

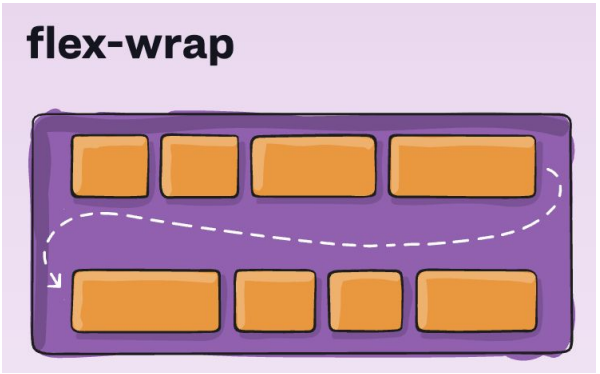
https://medium.com/js-imaginea/designing-a-layout-using-css-grid-flexbox-10df75a7f590

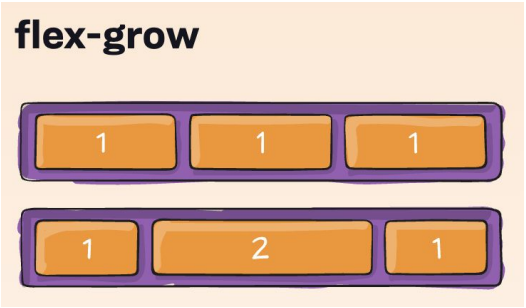
Mehrdimensional ⇒ Nested Flexbox

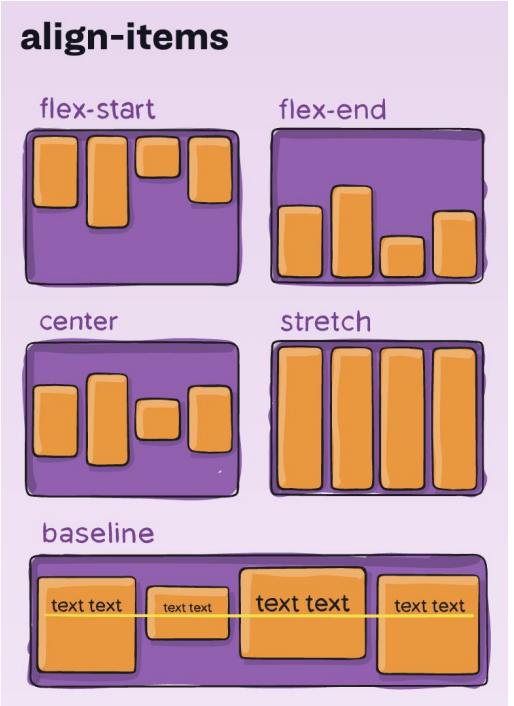


Flexbox Grundbegriffe





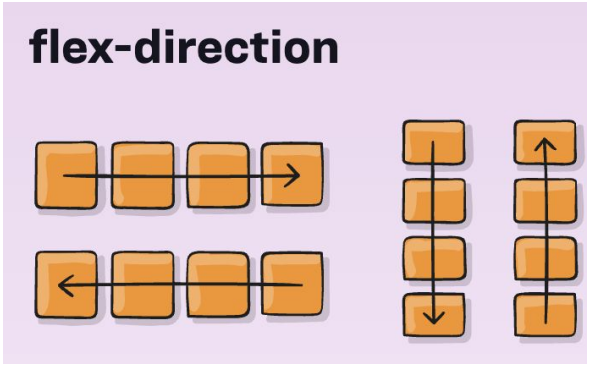




https://css-tricks.com/snippets/css/a-guide-to-flexbox/

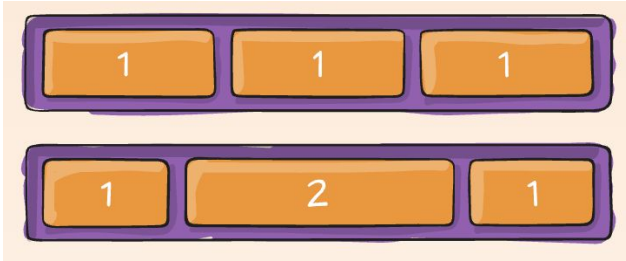
Flex Container Properties

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



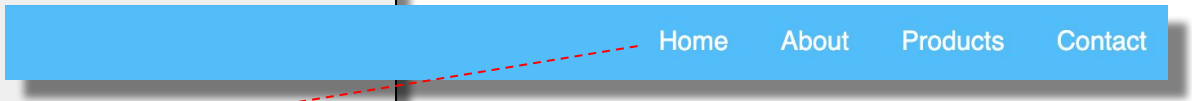
Flex Item Properties

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

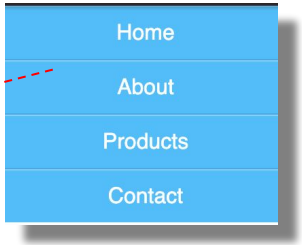


Beispiel: RWD Menu with Flexbox

```css
/* 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;
  }
}
```




```html
<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>
```



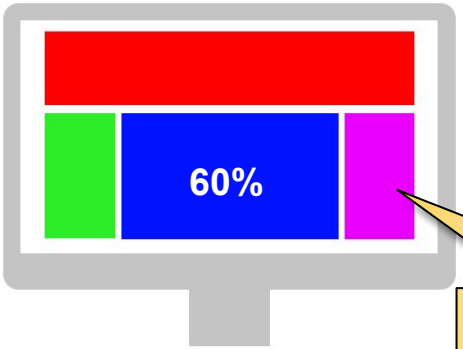
https://css-tricks.com/snippets/css/a-guide-to-flexbox/

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; }
}
```

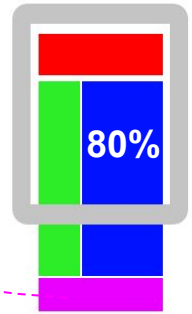
Dann kann man mit ganzen Prozent-Zahlen arbeiten

order




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

rot - grün - blau - pink

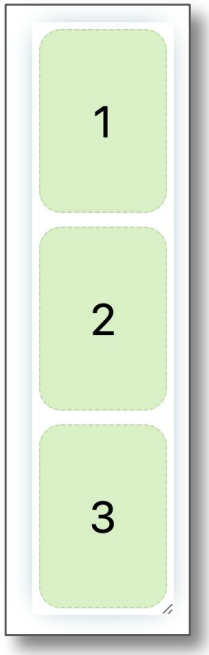


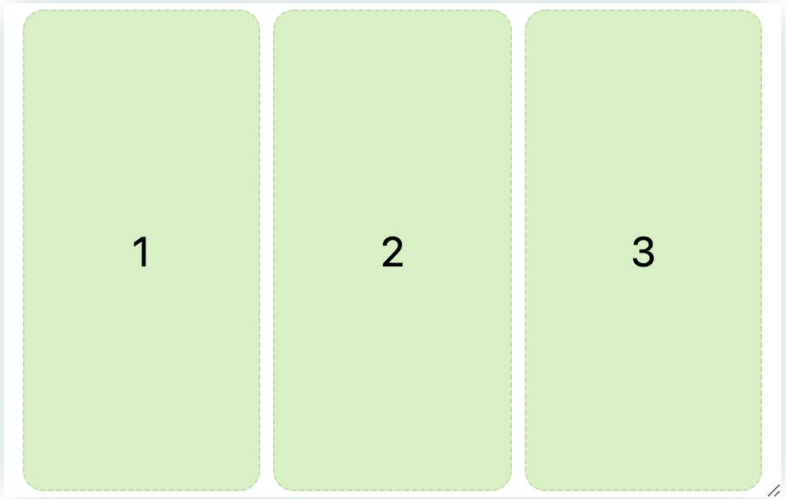
rot - blau - grün - pink

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 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/

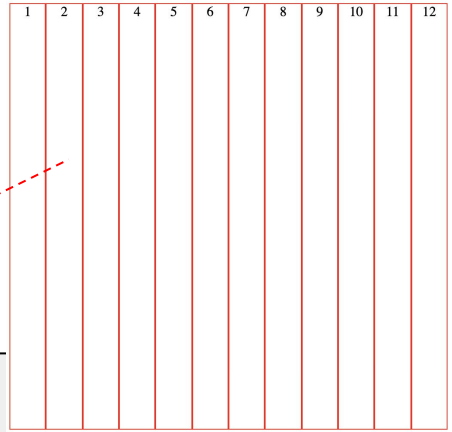
https://youtu.be/qm0IfG1GyZU?t=257

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>
```

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

in 12 Teile zerlegt



.header
.content
.sidebar

ab 360px

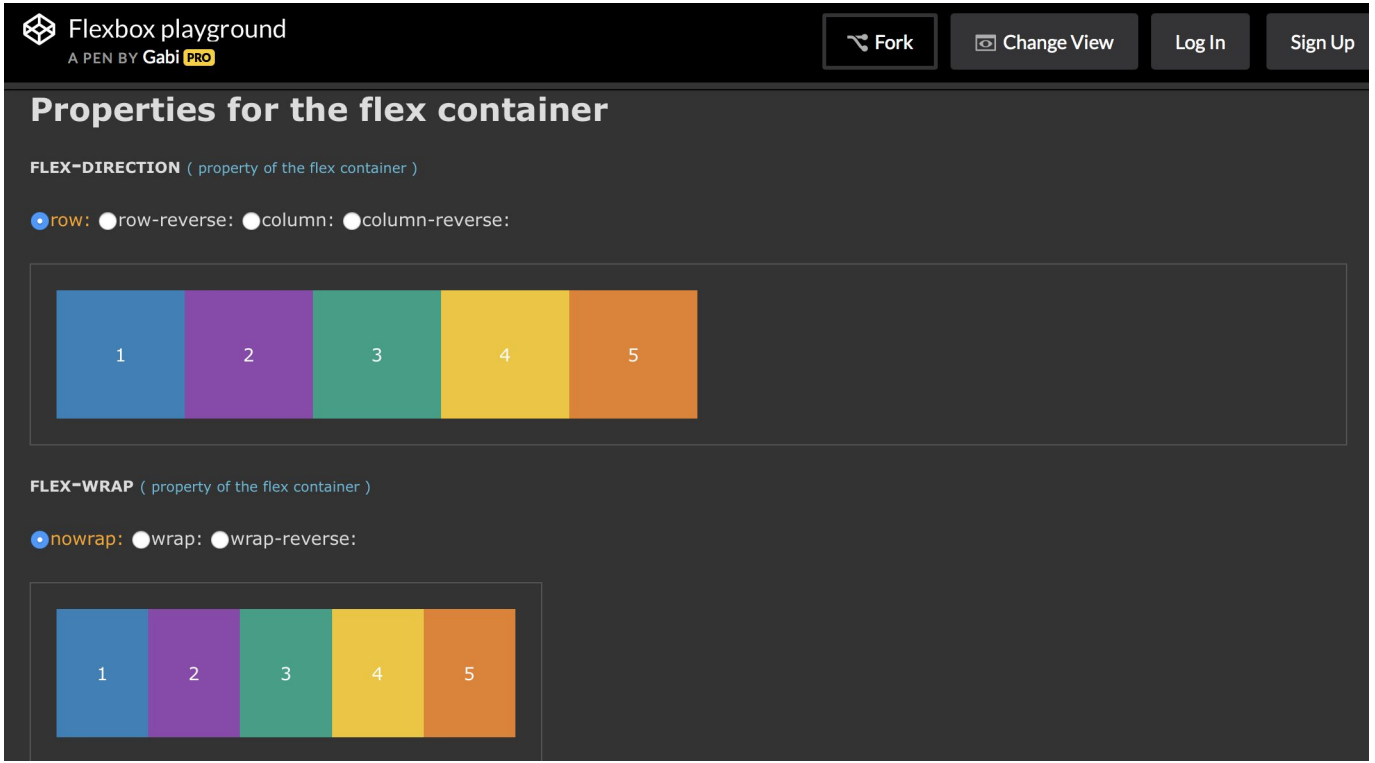
.header	
.content	.sidebar

ab 520px

.header	
.content	.sidebar

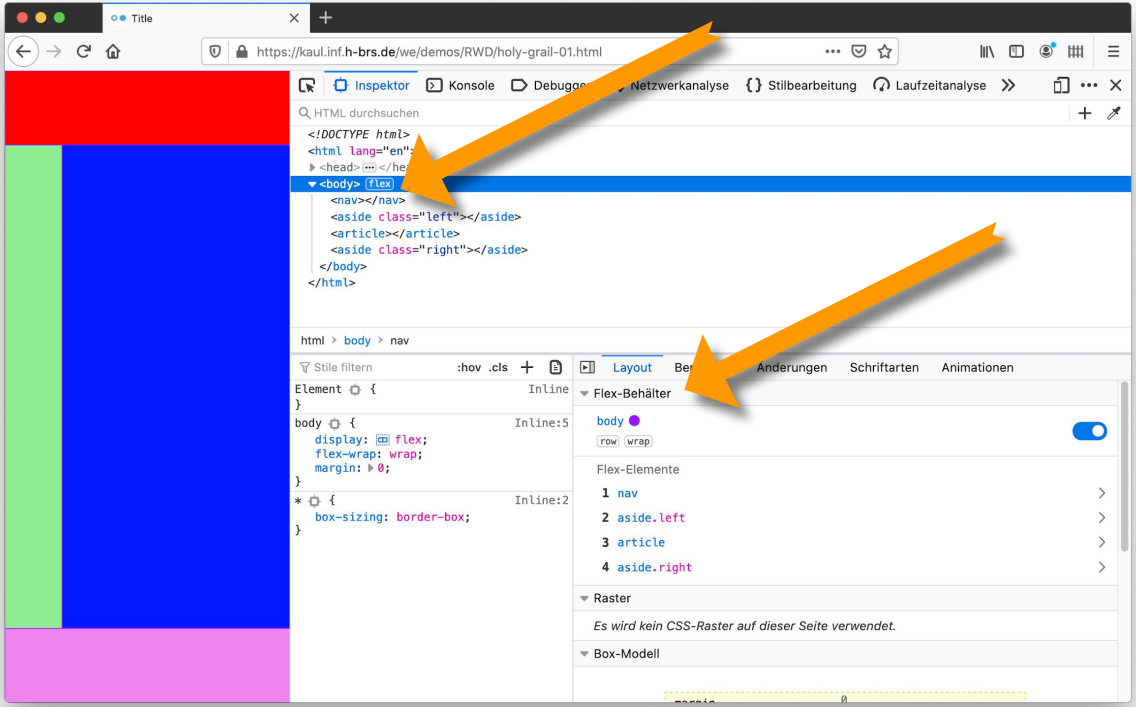
https://css-tricks.com/responsive-designs-and-css-custom-properties-building-a-flexible-grid-system/

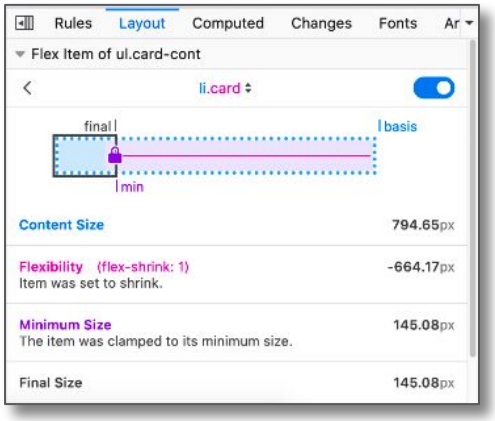
Codepen Flexbox Playground



https://codepen.io/enxaneta/full/adLPwv

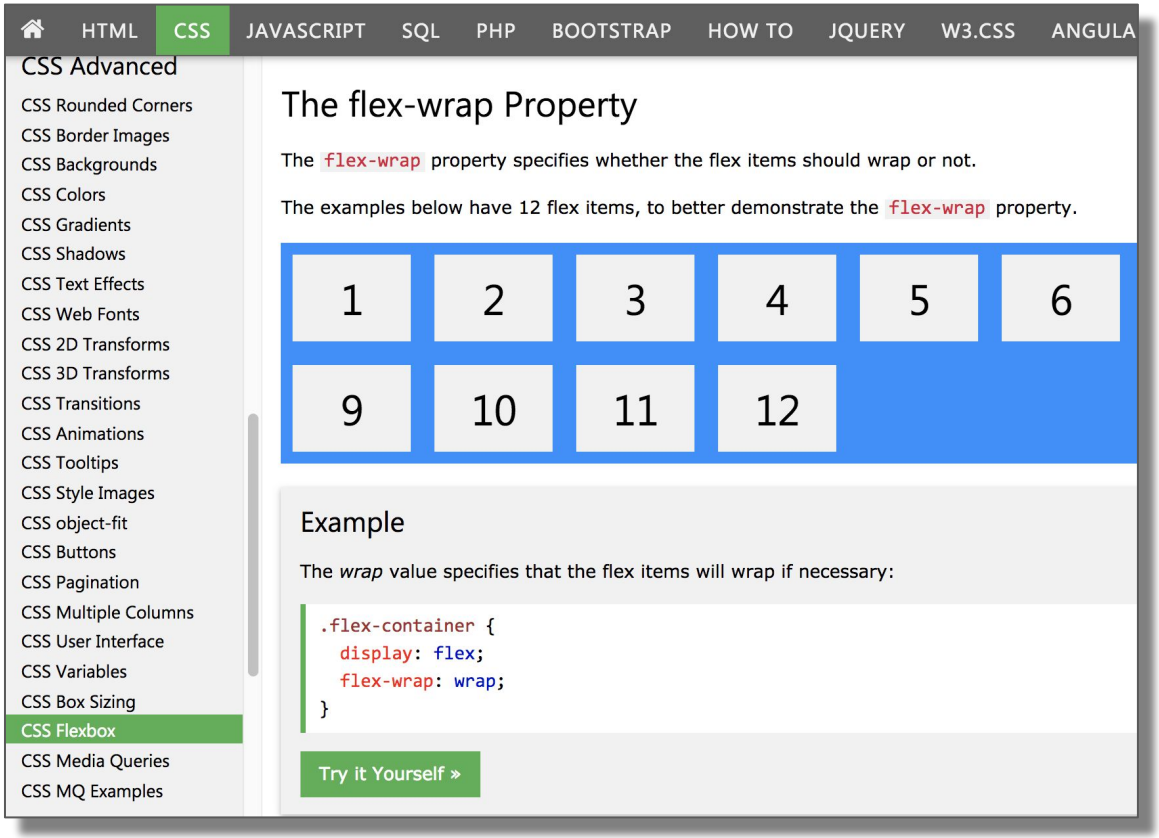
Firefox Flexbox Inspector





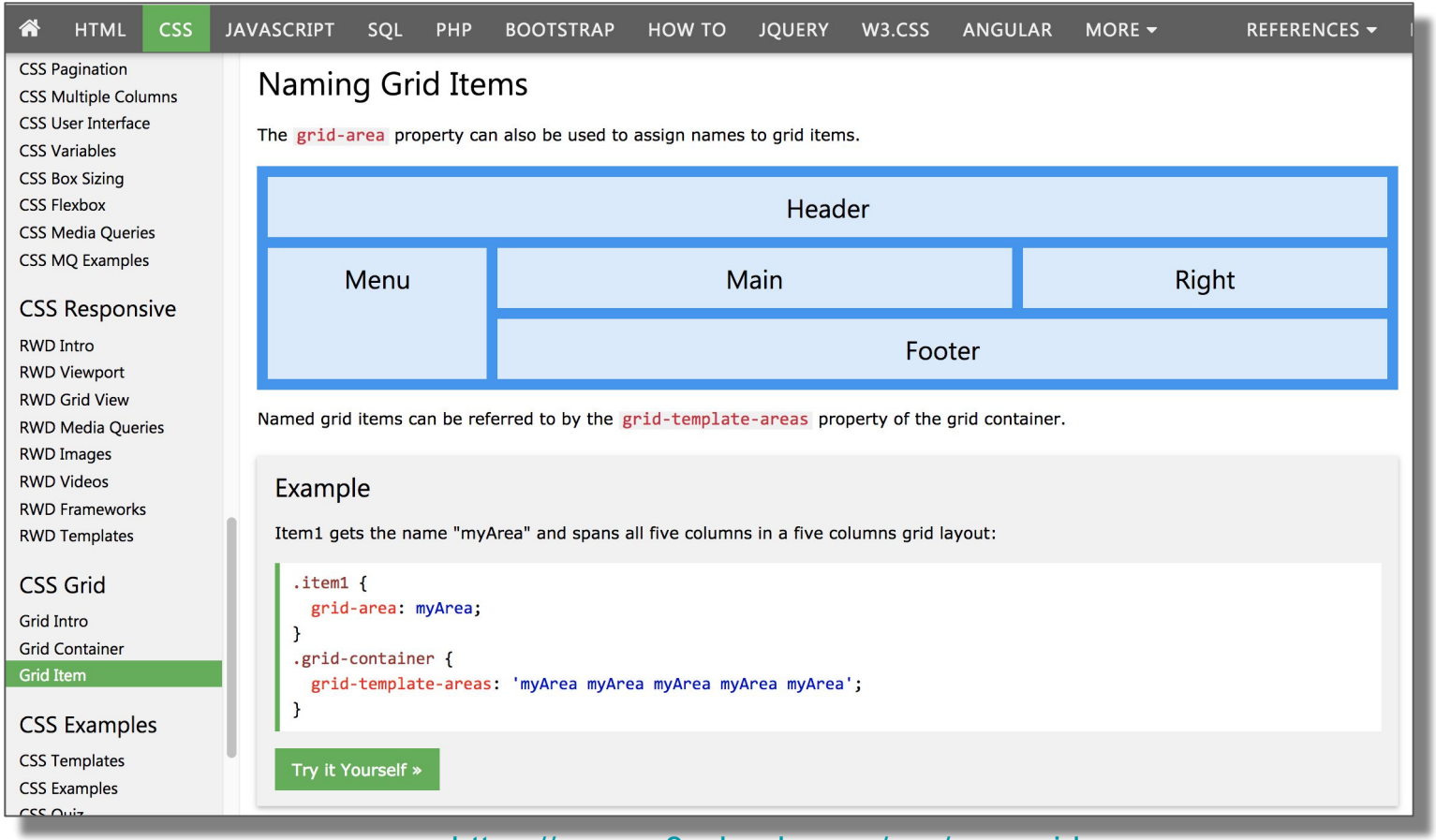
https://developer.mozilla.org/en-US/docs/Tools/Page_Inspector/How_to/Examine_Flexbox_layouts

w3schools Tutorial zum Flexbox-Layout



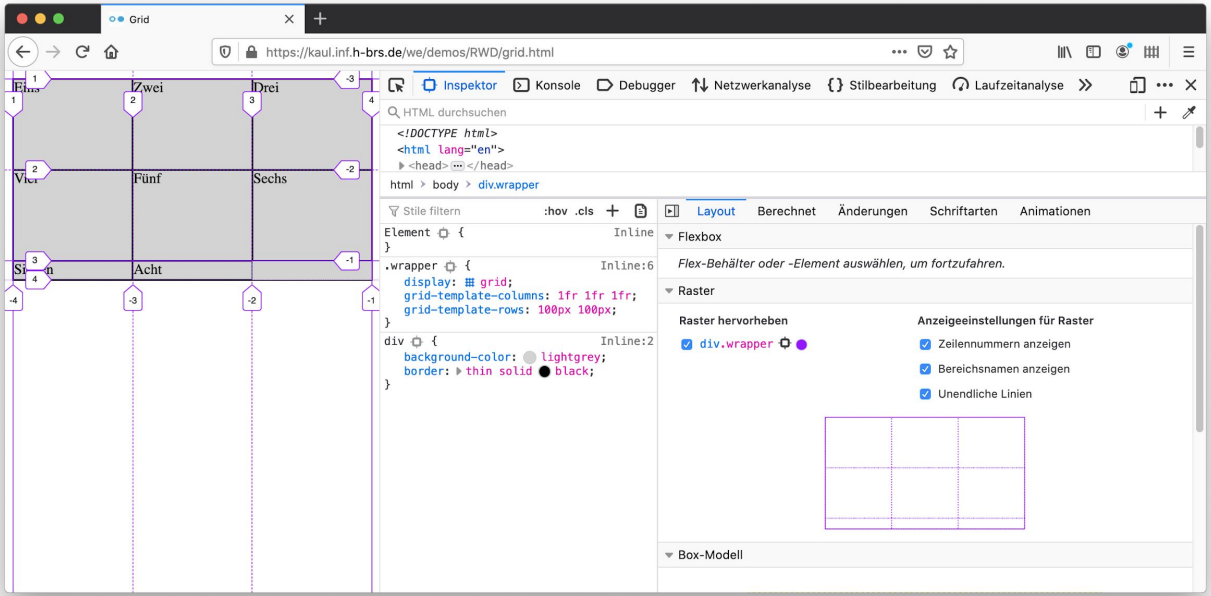
https://www.w3schools.com/css/css3_flexbox.asp

7. Grid-Layout



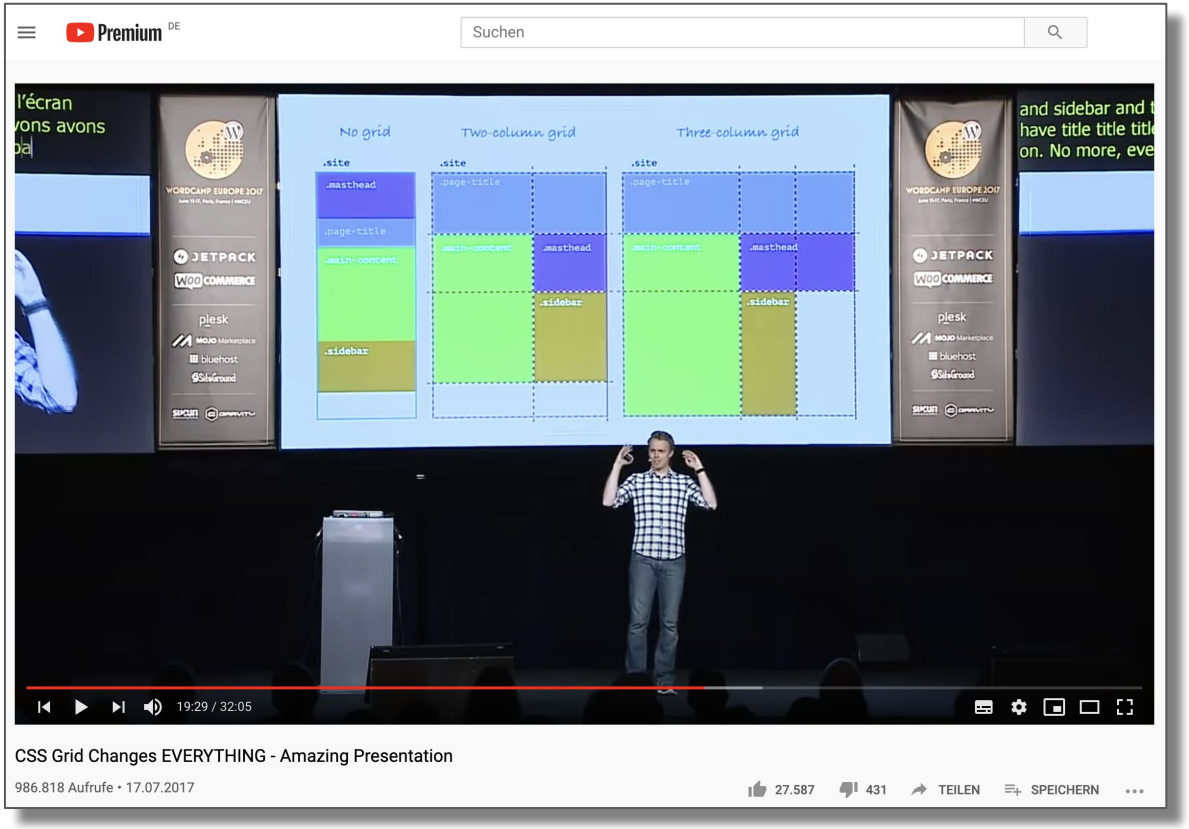
https://www.w3schools.com/css/css_grid.asp

Firefox Grid Inspector



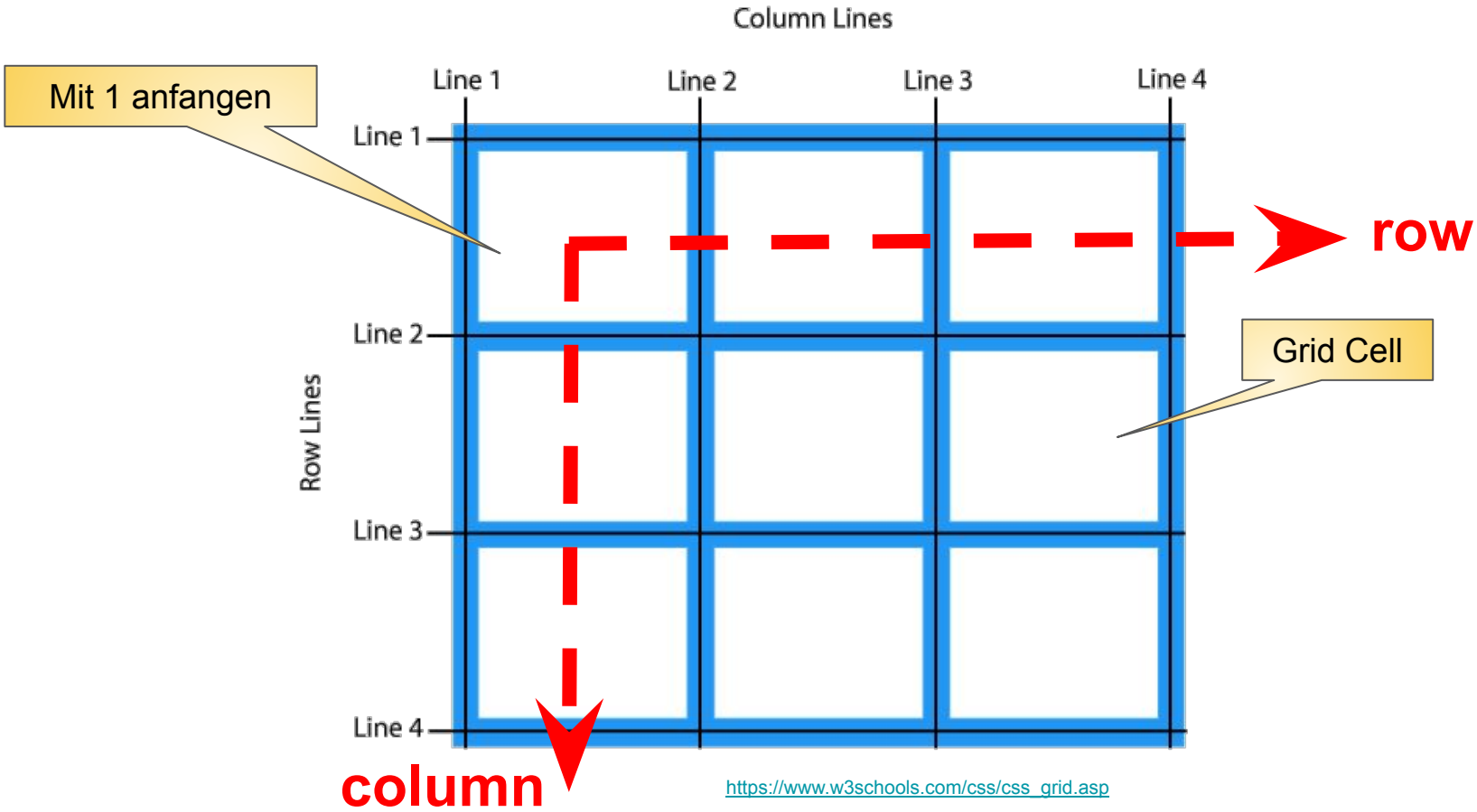
https://developer.mozilla.org/de/docs/Tools/Page_Inspector/How_to/Raster_Layout_untersuchen

CSS Grid changes EVERYTHING



https://www.youtube.com/watch?v=7kVeCqQCxlk

Grid-Terminologie



https://www.w3schools.com/css/css_grid.asp

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

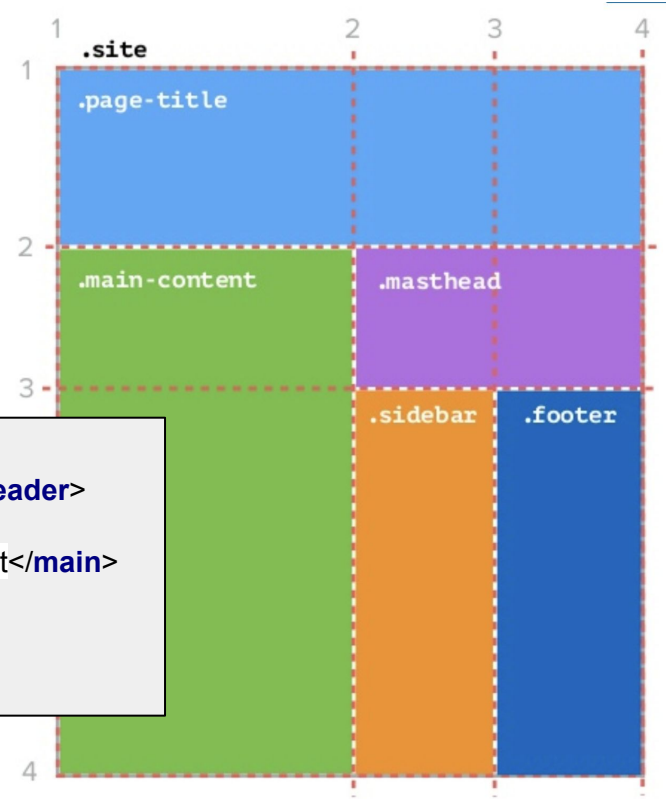
Grid Item Properties

- 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>
```

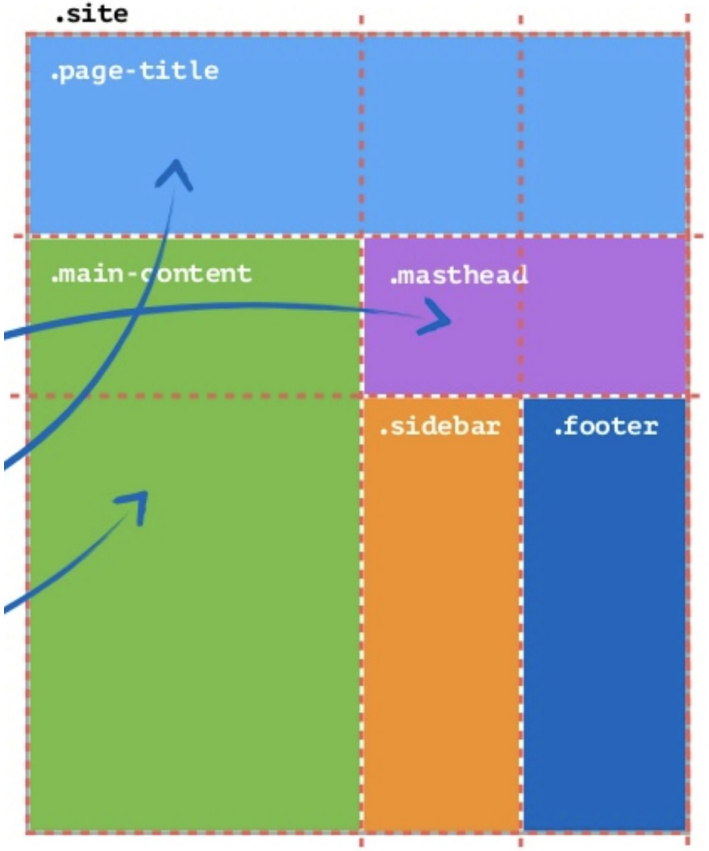
```
<div class="site">
  <header class="masthead">masthead</header>
  <nav class="page-title">page-title</nav>
  <main class="main-content">main-content</main>
  <aside class="sidebar">sidebar</aside>
  <footer class="footer">footer</footer>
</div>
```



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

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>
```



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

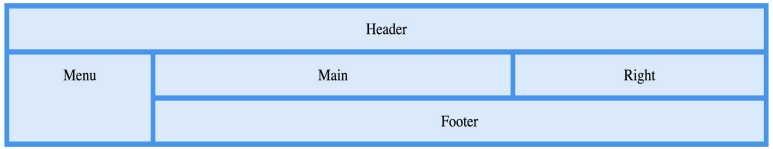
Positionierung eines Items im Grid

```
.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.



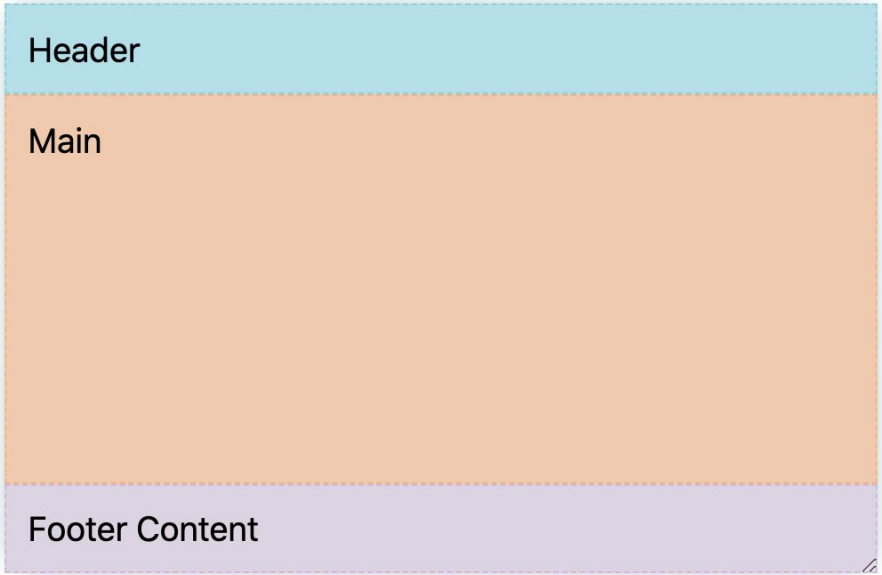

```
.item1 { grid-area: header; }
.item2 { grid-area: menu; }
.item3 { grid-area: main; }
.item4 { grid-area: right; }
.item5 { grid-area: footer; }

.grid-container {
  grid-template-areas:
      'header header header header header header'
      'menu main main main right right'
      'menu footer footer footer footer footer';
}
```

https://www.w3schools.com/css/css_grid_item.asp

Pancake Stack Layout with Grid "auto 1fr auto"

04. Pancake Stack

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



Current Browser Support:

Edge Firefox Chrome Safari

HTML CSS

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

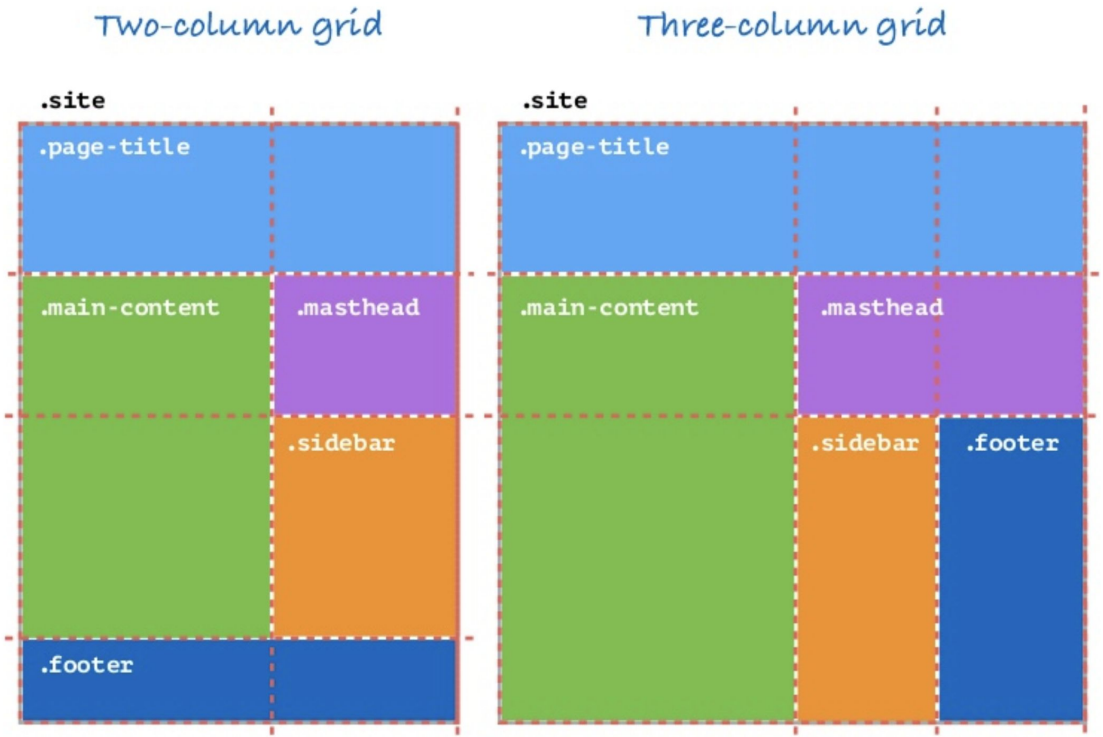
Explore on CodePen

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";
  }
}
```

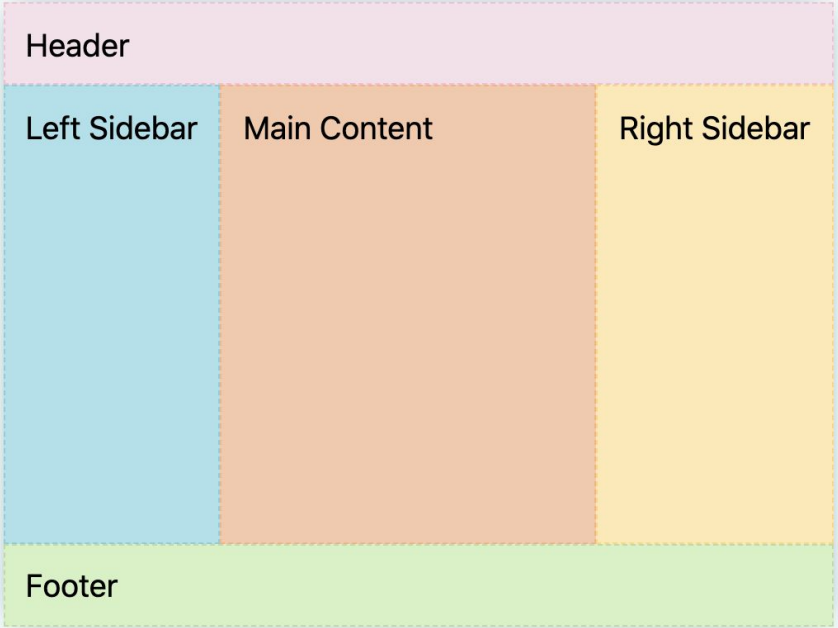


https://www.slideshare.net/mor10/css-grid-changes-everything-about-web-layouts-wordcamp-europe-2017/42

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 {
```

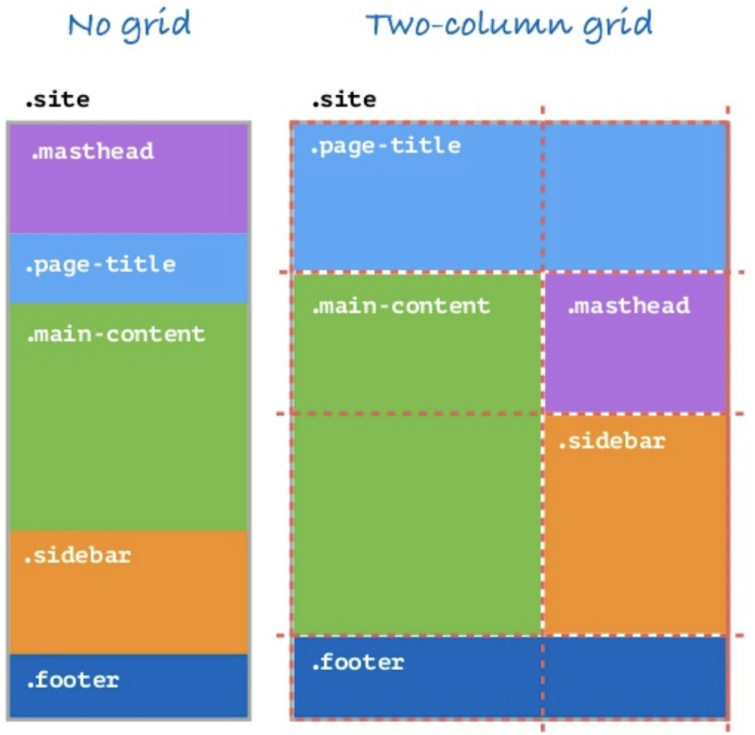
Explore on CodePen

https://1linelayouts.glitch.me/

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

```css
.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



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;

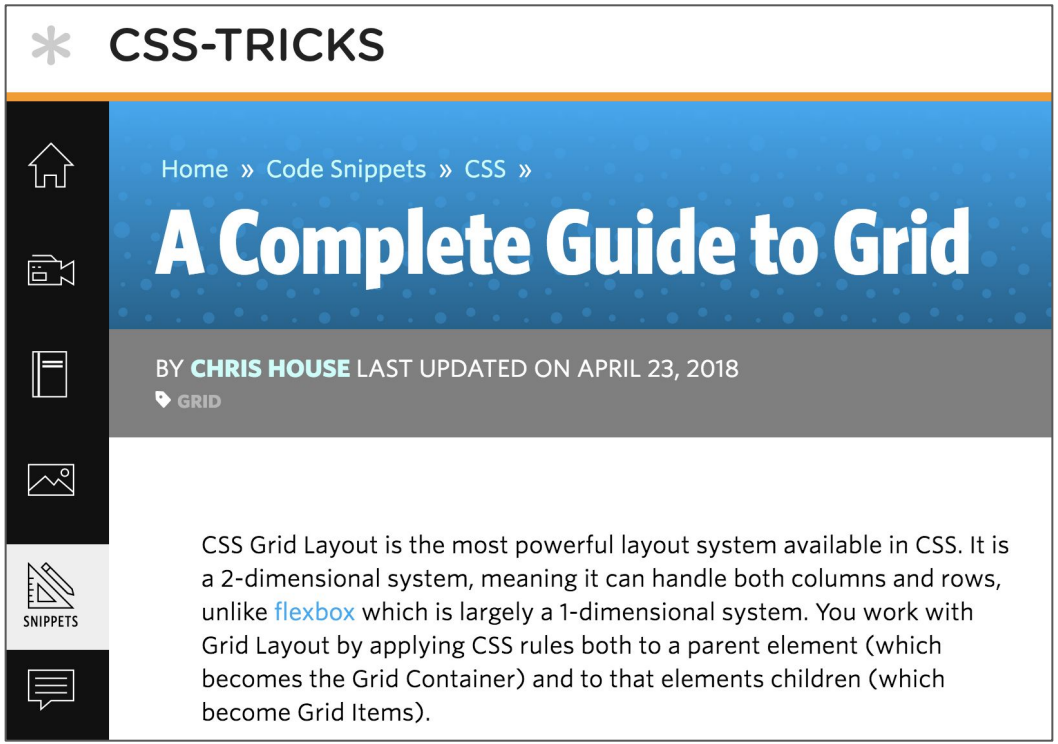
```
@media (min-width: 400px) {
  body {
    grid-template-columns: repeat(4, 1fr);
  }
}
```

⇔ 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])
```

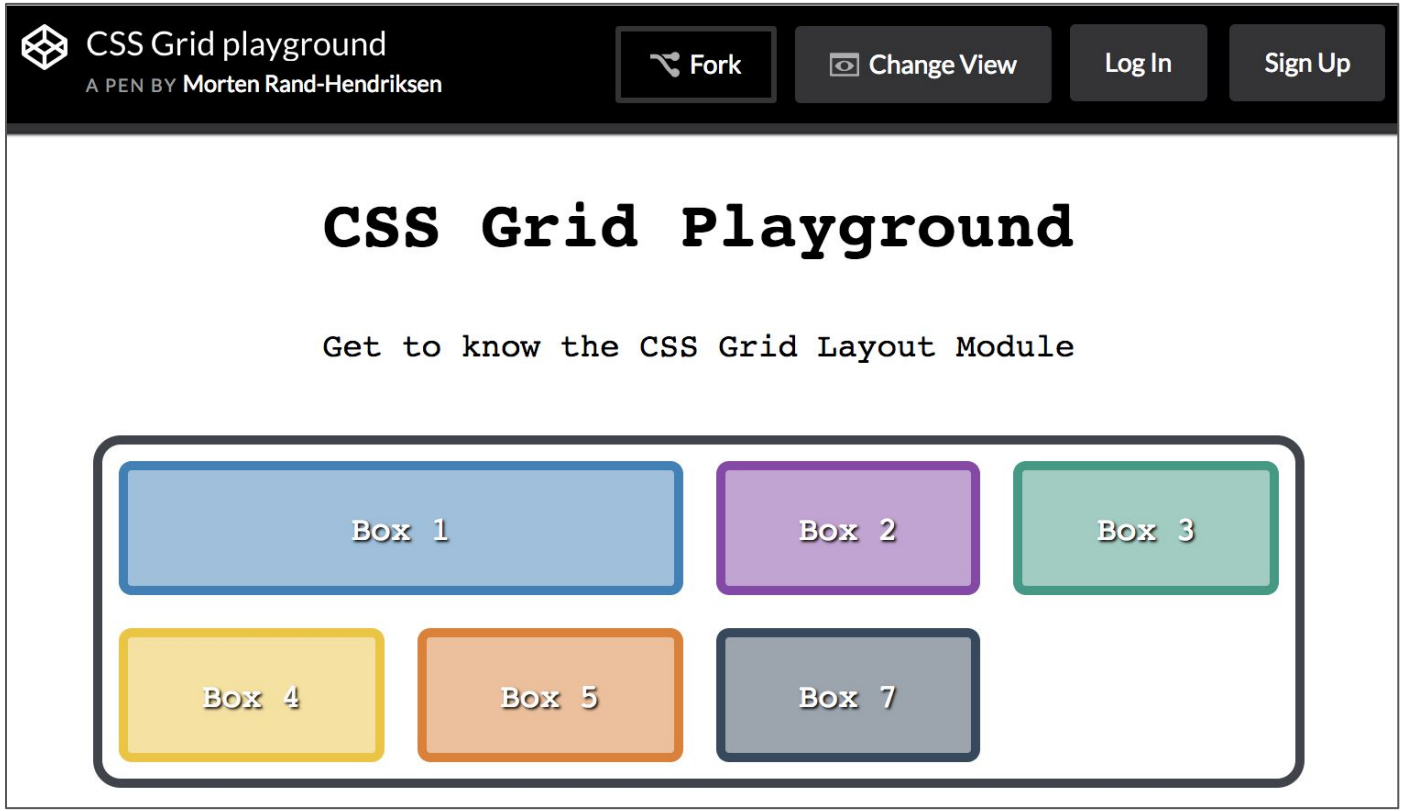
https://developer.mozilla.org/en-US/docs/Web/CSS/repeat

CSS Tricks Complete Guide to Grid



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

CodePen Grid Playground



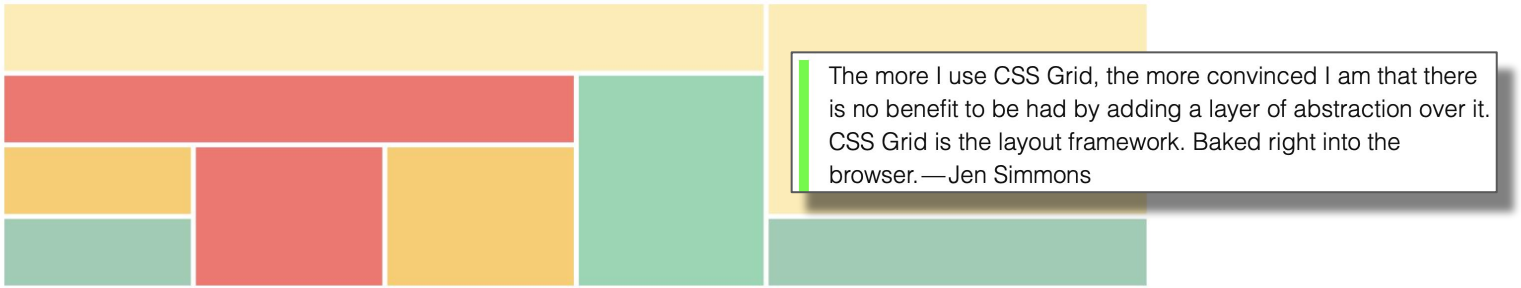
https://codepen.io/mor10/full/MEQJbM

Vergleich von Grid mit Bootstrap

Why CSS Grid is better than Bootstrap for creating layouts

October 14th 2019

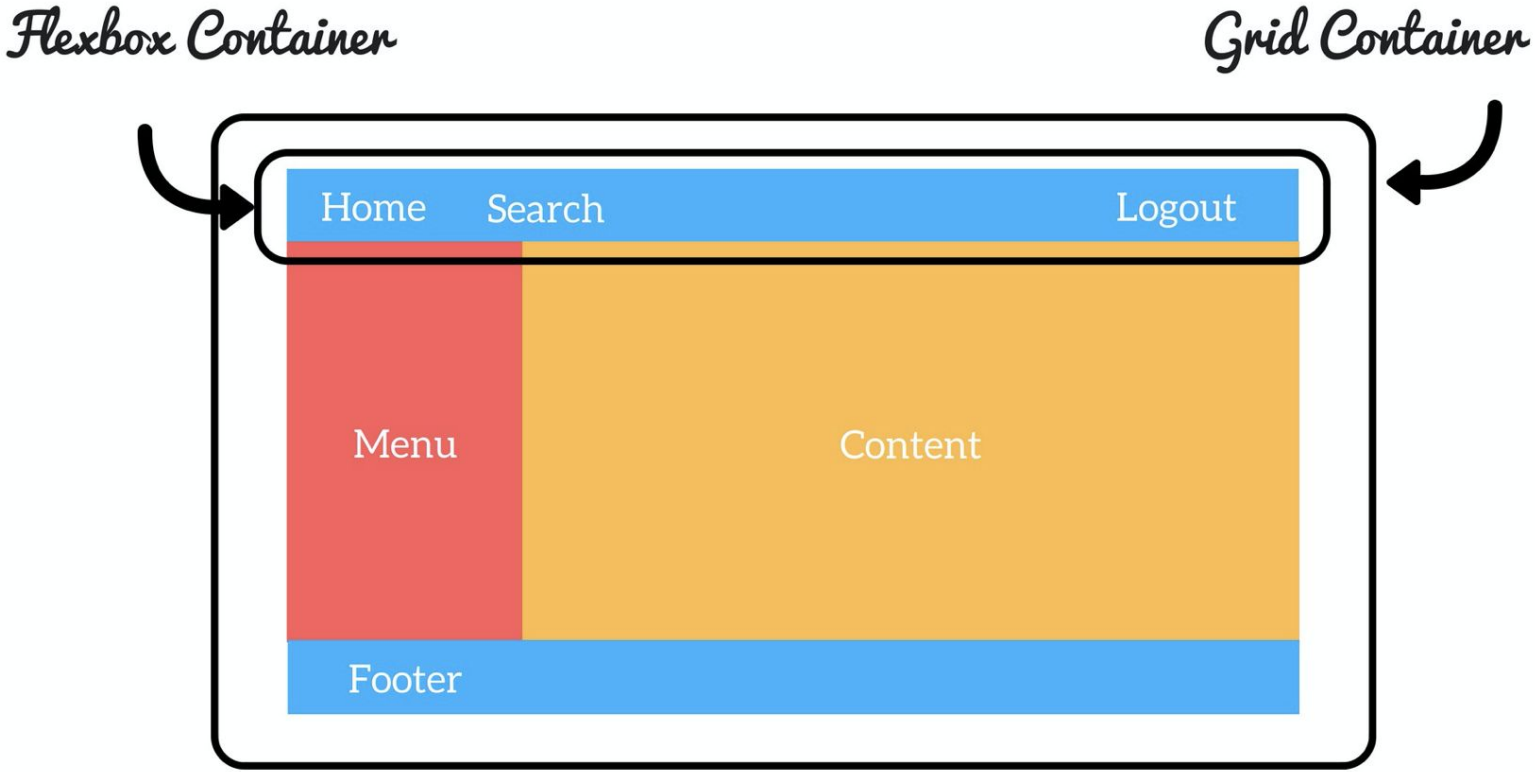
TWEET THIS



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



https://medium.com/youstart-labs/beginners-guide-to-choose-between-css-grid-and-flexbox-783005dd2412

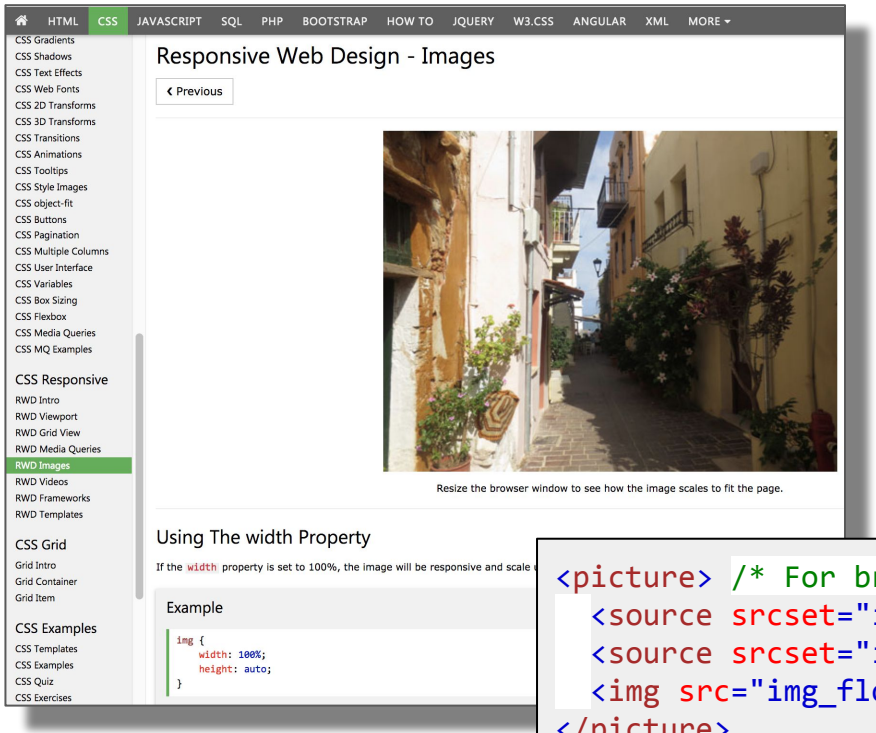
8. Flexible Images

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



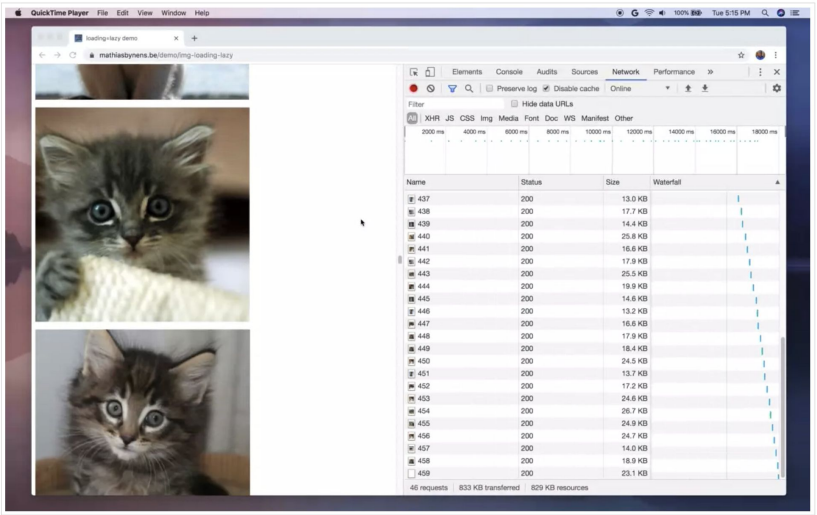

```
<figure>
  <img src="images/filename.jpg"
       alt="some text about the image"
       width="60vw"
       height="auto">
  <figcaption>Bildunterschrift</figcaption>
</figure>
```

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

https://www.w3schools.com/css/css_rwd_images.asp

Image lazy loading

```
<img src="cat1.jpg" loading="lazy">
```



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.

Usage: Global — % of all users: 71.31% + 3.47% = 74.77%

Current aligned | Usage relative | Date relative | Filtered | All

IE	Edge *	Firefox	Chrome	Safari	Opera	iOS Safari *	Opera Mini *	Android Browser *	Opera Mobile *	Chrome for Android	Firefox for Android	UC Browser for Android
			4-74		10-60							
	12-18	2-74	75-76	3.1-13	62-63	3.2-13.3						
6-10	79-85	1 75-80	77-85	2 13.1	64-70	2 13.7		2.1-4.4.4	12-12.1			
11	86	1 81	86	2 14	71	2 14	all	81	59	85	1 79	12.12
		1 82-83	87-89	2 TP								

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

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

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

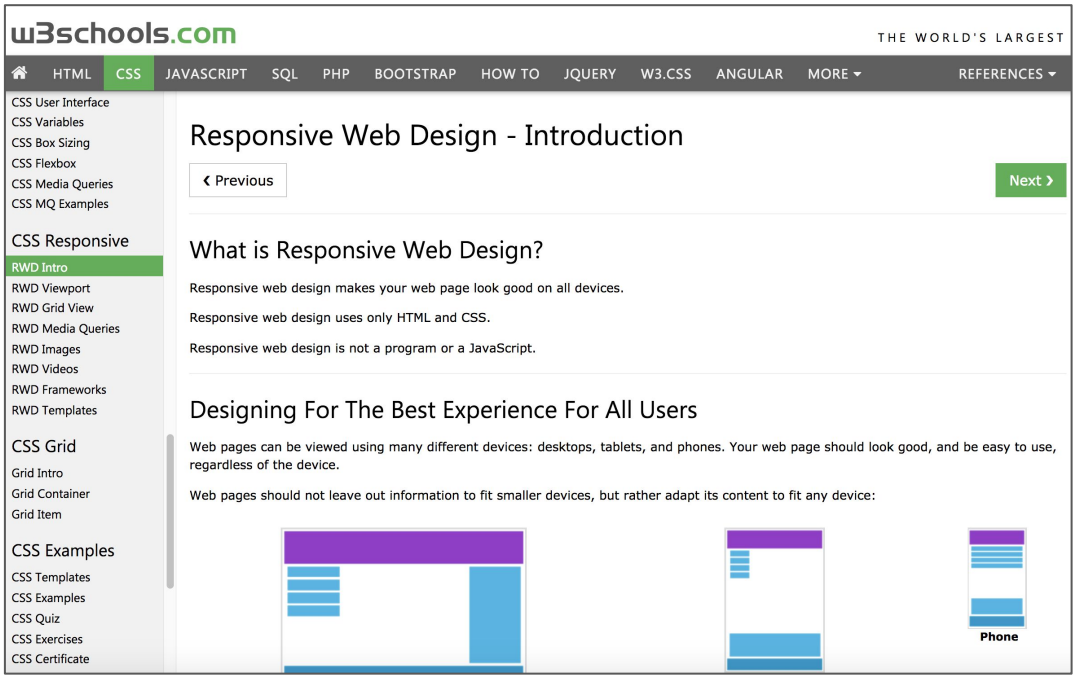
```
<style>
  p {
    /*** Fallback ***/
    font-size: 18px;

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

selfhtml wiki
Die Energie des Verstehens

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



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/