CSS Advanced

CSS Position Property

Video Quelle:

CSS Positions in unter 3 Minuten | Tutorial Deutsch

https://www.youtube.com/watch?v=9u11eALvhhU

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

Position Intro

Die Eigenschaft **position** gibt die Art der für ein Element verwendeten Positionierungsmethode an

- static
- relative
- fixed
- absolute
- sticky

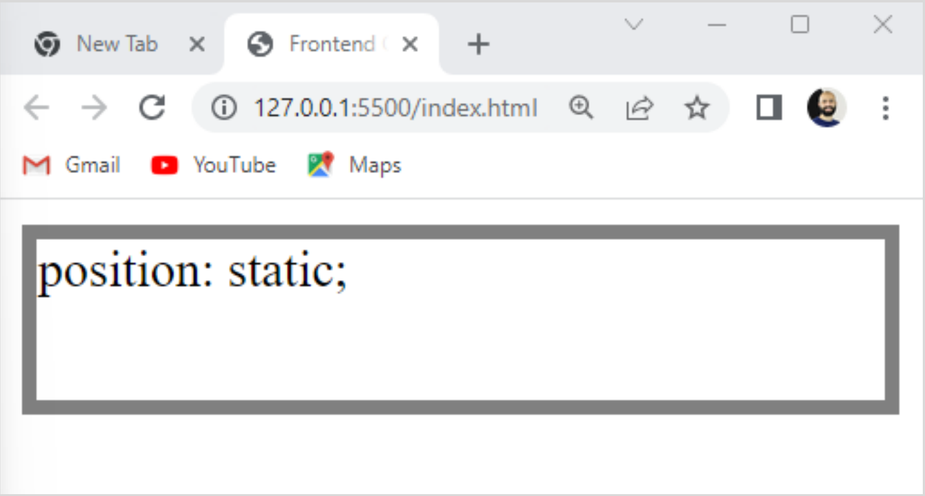
NOTE:
Die Elemente werden dann mit den Eigenschaften oben, unten, links und rechts positioniert. Diese Eigenschaften funktionieren jedoch nur, wenn zuvor die Eigenschaft **position** festgelegt wurde. Außerdem funktionieren sie je nach dem Positionswert unterschiedlich.

Position: static

```
position: static;
```

HTML Elemente werden standardmäßig statisch positioniert.

- Statisch positionierte Elemente sind von den Eigenschaften oben, unten, links und rechts nicht betroffen.
- Statisch immer entsprechend dem normalen Fluss der Seite positioniert

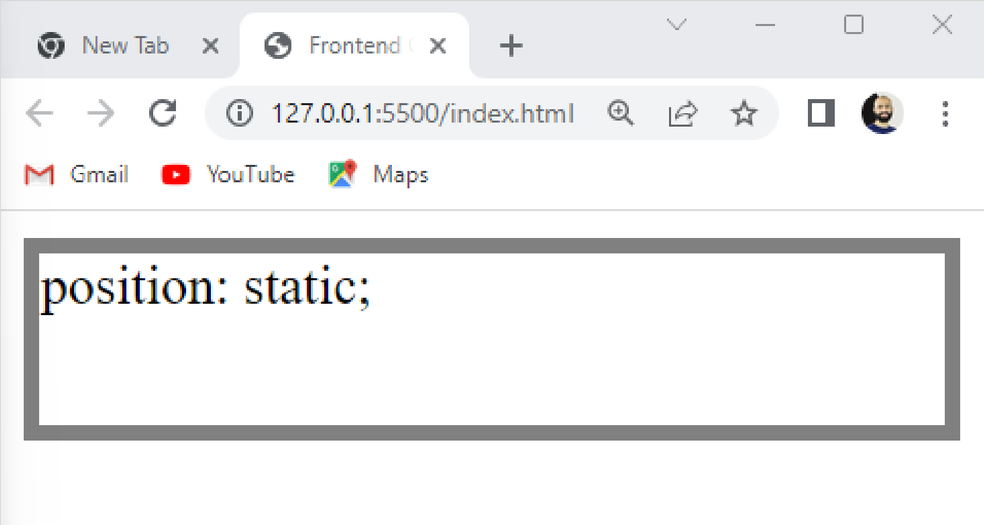


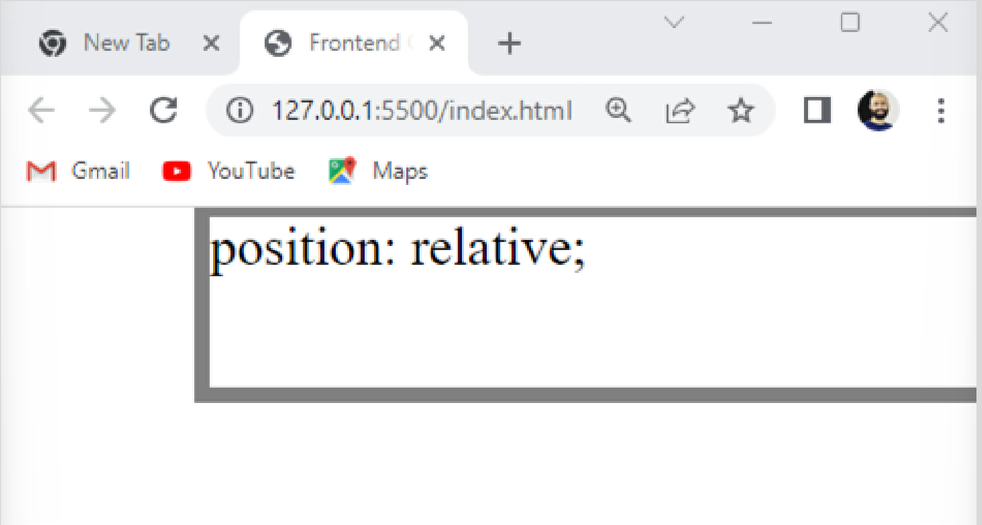
Position: relative

Ein Element mit **position**: relativ; wird relativ zu seiner normalen Position positioniert.

Setzen der top, right, bottom, and left Eigenschaften eines relativ positionierten Elements bewirkt, dass es von seiner normalen Position abweicht.

```
position: relative;
top: -10px;
left: 50px;
```





Position: fixed

```
position: fixed;
bottom: 0;
right: 0;
```

Ein Element mit Position: fixed; wird relativ zum Ansichtsfenster positioniert, d. h. es bleibt immer an der gleichen Stelle, auch wenn die Seite gescrollt wird. Die Eigenschaften oben, rechts, unten und links werden zur Positionierung des Elements verwendet.

- Ein feststehendes Element hinterlässt keine Lücke auf der Seite, wo es sich normalerweise befunden hätte.

Position: absolute

Ein Element mit Position : absolute; wird relativ zum nächstgelegenen positionierten Vorgänger positioniert (statt relativ zum Ansichtsfenster (Viewport), wie bei fixed).

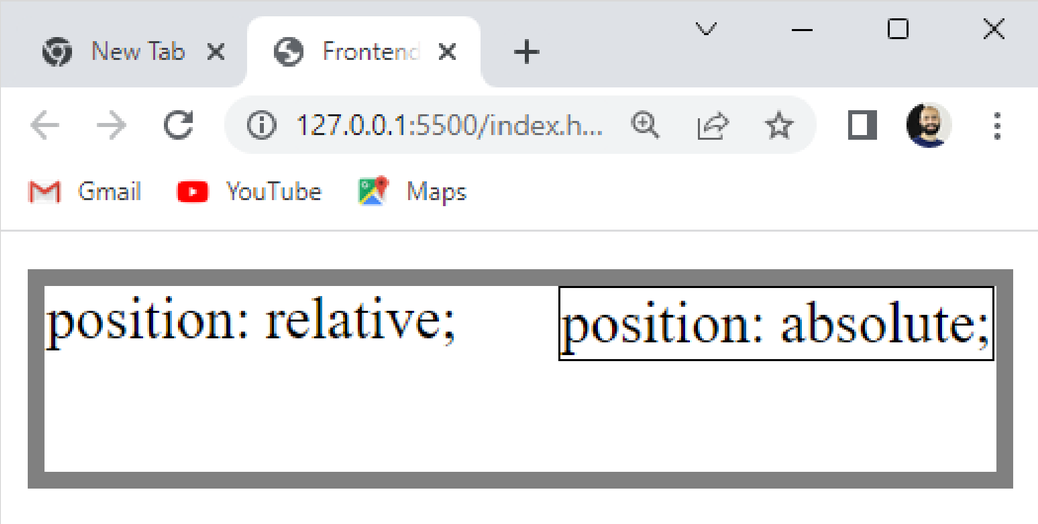
- Wenn ein absolut positioniertes Element keine positionierten Vorgänger hat, verwendet es den Dokumentenkörper (document body) und bewegt sich mit dem Bildlauf der Seite mit.

Sehen Sie sich die Beispiele auf der nächsten Folie an

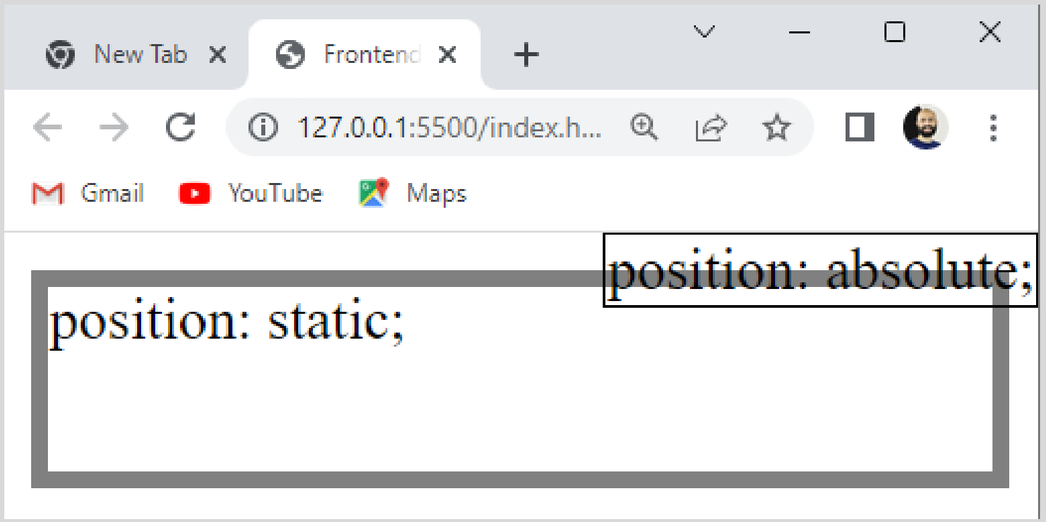
Position: absolute

```
position: absolute;
top: 0;
right: 0;
```

Parent div ist relative



Parent div ist static



Position: sticky

Ein Element mit Position : sticky; wird auf der Grundlage der Bildlaufposition des Benutzers positioniert.

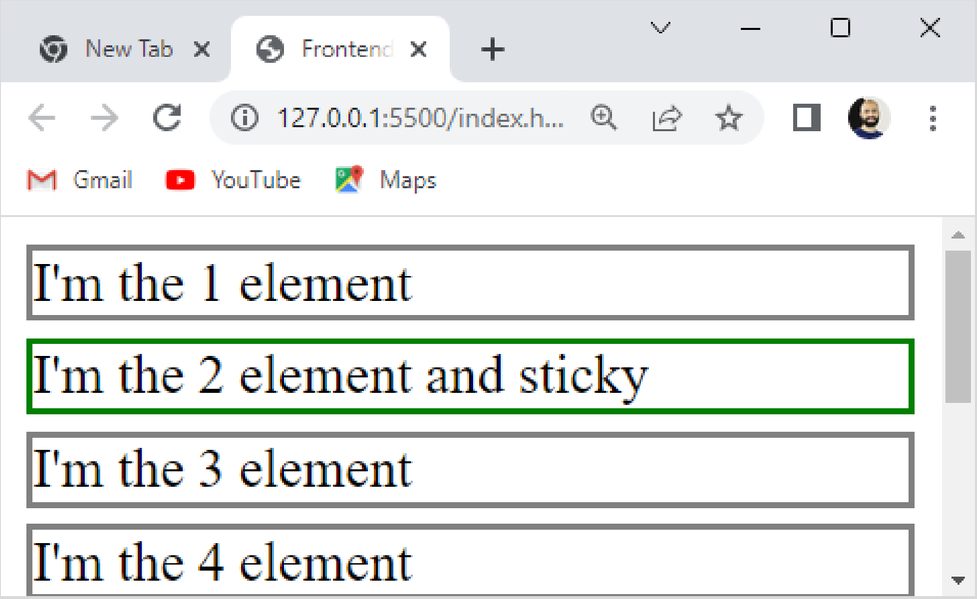
- Ein Sticky-Element schaltet je nach Bildlaufposition zwischen relative und fixed um. Es wird relativ positioniert, bis eine bestimmte Offset-Position im Ansichtsfenster (Viewport) erreicht wird - dann "klebt" es an seinem Platz (wie position:fixed).

Sehen Sie sich die Beispiele auf der nächsten Folie an

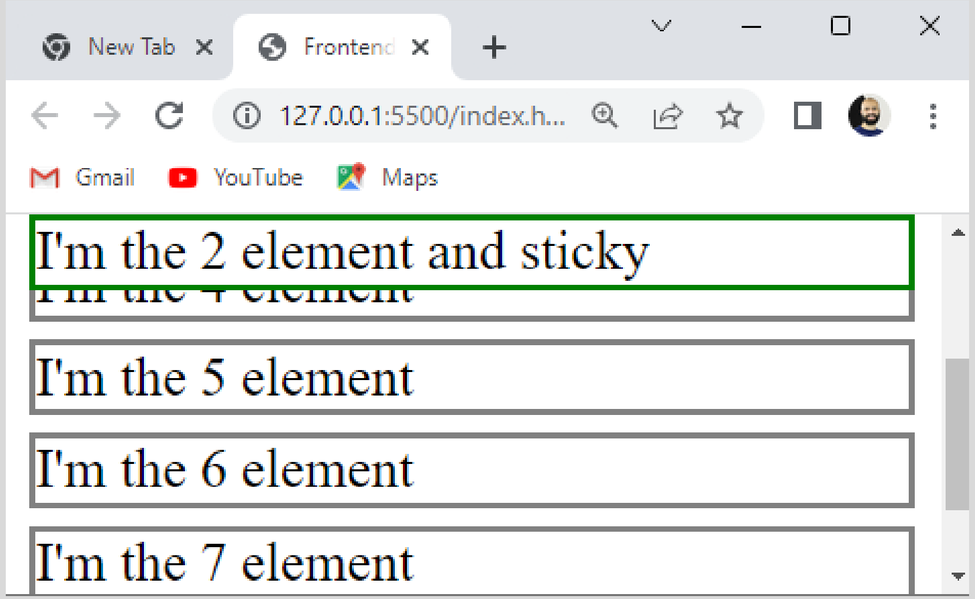
Position: sticky

```
position: sticky;
top: 0;
```

Vor scrolling



Während/nach scrolling



Position: Video

Video Quelle:

Learn CSS Position In 9 Minutes

https://youtu.be/jx5jmI0UlXU

https://codepen.io/WebDevSimplified/pen/VNRMNP

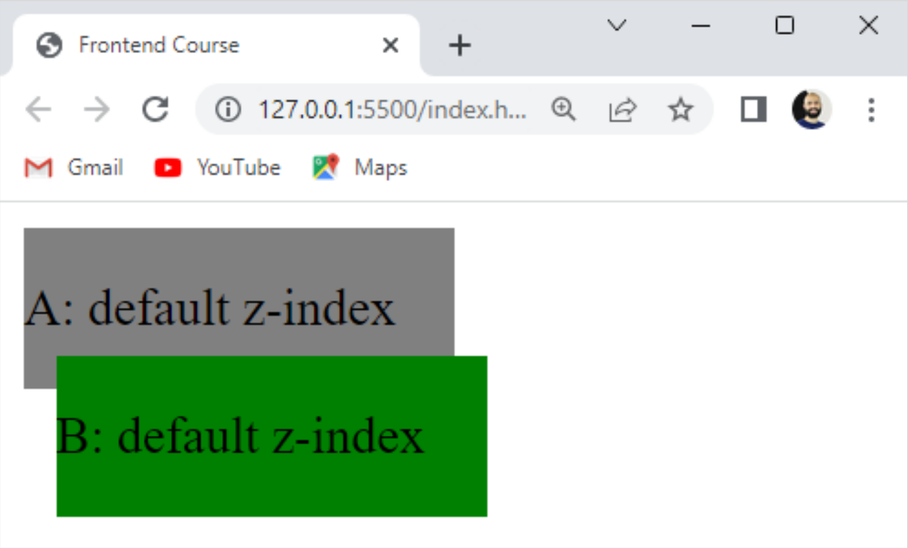
CSS Z-index

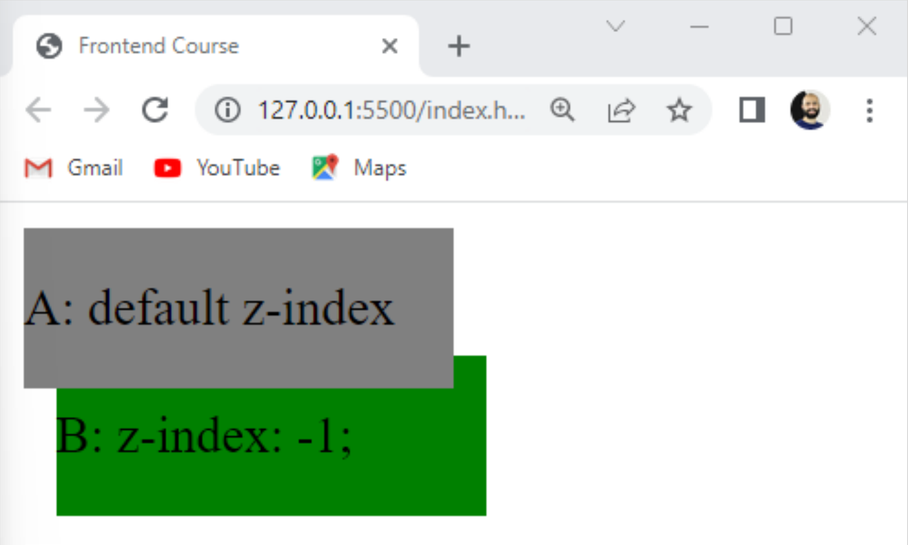
Z-index

Die Eigenschaft **z-index** gibt die Stapelreihenfolge eines Elements an (welches Element vor oder hinter den anderen platziert werden soll).

- Auto ist der Standardwert für alle HTML-Elemente.
- Die zuletzt in HTML definierten Elemente werden standardmäßig ganz oben angezeigt.

```
z-index: -1;
```





https://www.w3schools.com/css/css_z-index.asp

CSS Overflow

Overflow: Intro

Die Eigenschaft **overflow** gibt an, ob der Inhalt abgeschnitten oder Bildlaufleisten hinzugefügt werden sollen, wenn der Inhalt eines Elements zu groß ist, um in den angegebenen Bereich zu passen.

Werte:

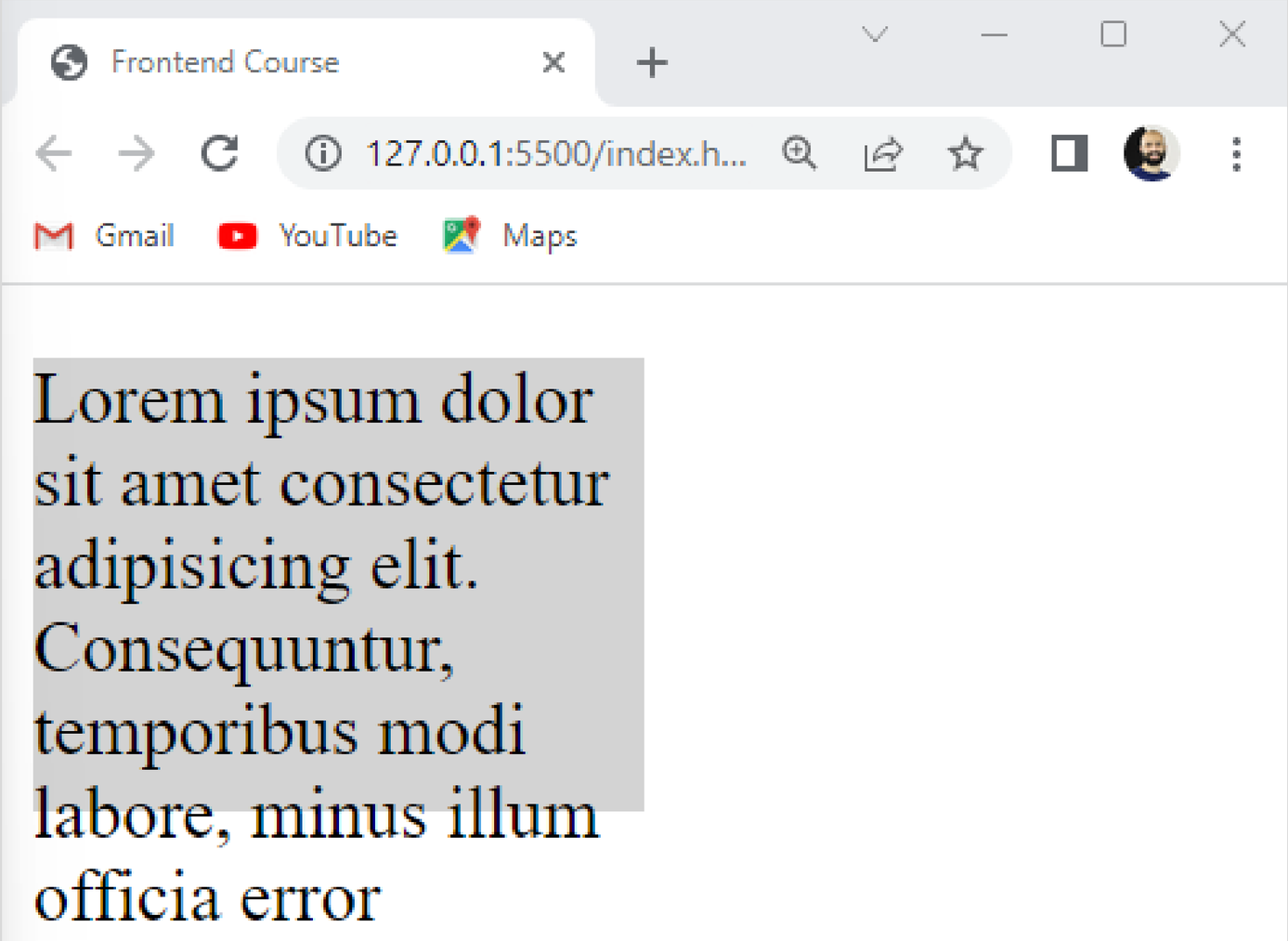
- Visible: Standard. Der Überlauf wird nicht beschnitten. Der Inhalt wird außerhalb des Rahmens des Elements wiedergegeben.
- Hidden: Der Überlauf wird abgeschnitten, und der Rest des Inhalts wird ausgeblendet
- Scroll: Der Überlauf wird abgeschnitten und eine Bildlaufleiste wird hinzugefügt, um den Rest des Inhalts zu sehen
- Auto: Ähnlich wie scroll, aber es fügt Bildlaufleisten nur bei Bedarf hinzu

Note:

Die Eigenschaft overflow funktioniert nur bei Blockelementen mit einer bestimmten Höhe.

Overflow: visible "default"

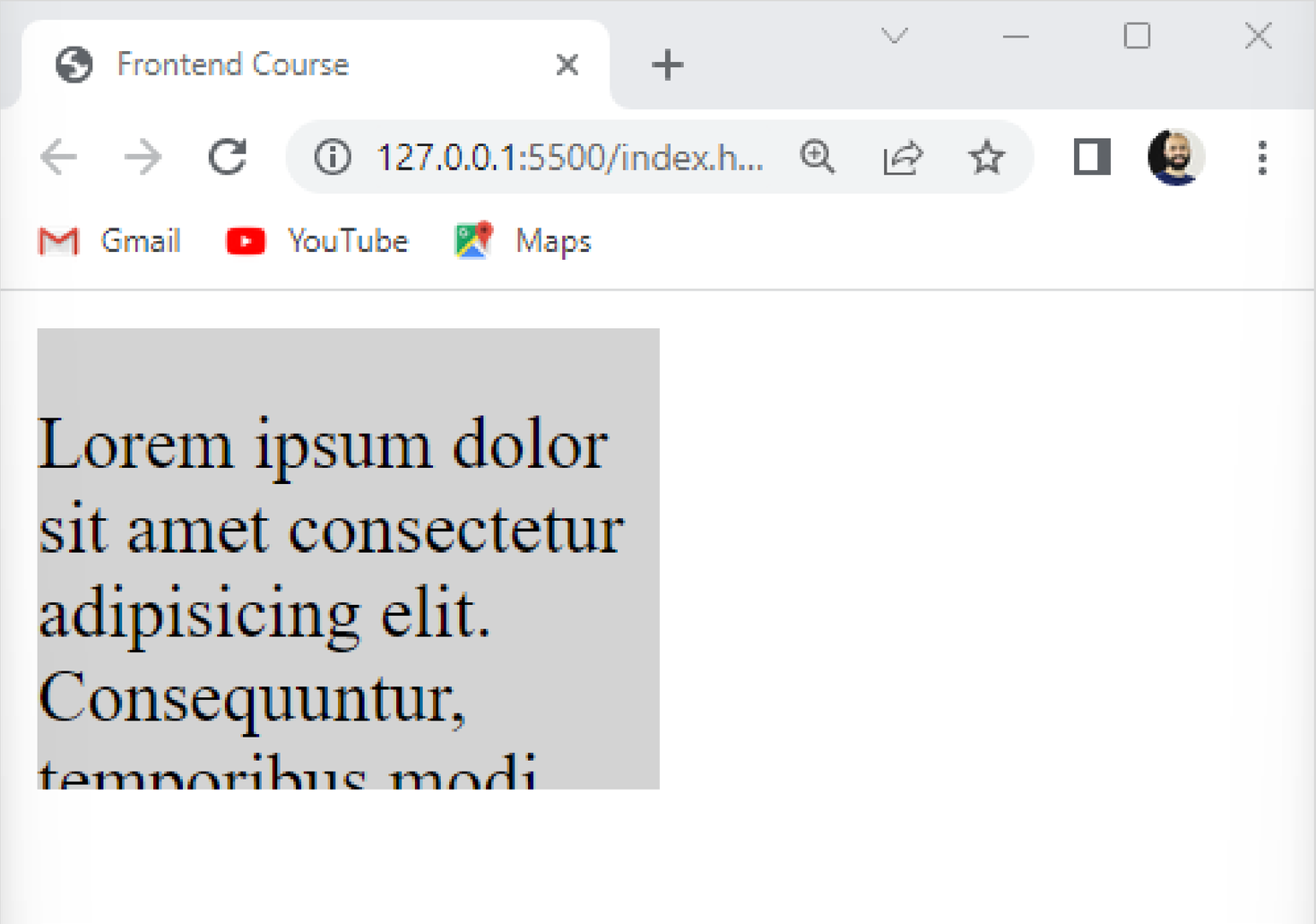
- Visible: Standard. Der Überlauf wird nicht beschnitten. Der Inhalt wird außerhalb des Rahmens des Elements wiedergegeben.



Overflow: hidden

- Hidden: Der Überlauf wird abgeschnitten, und der Rest des Inhalts wird ausgeblendet.

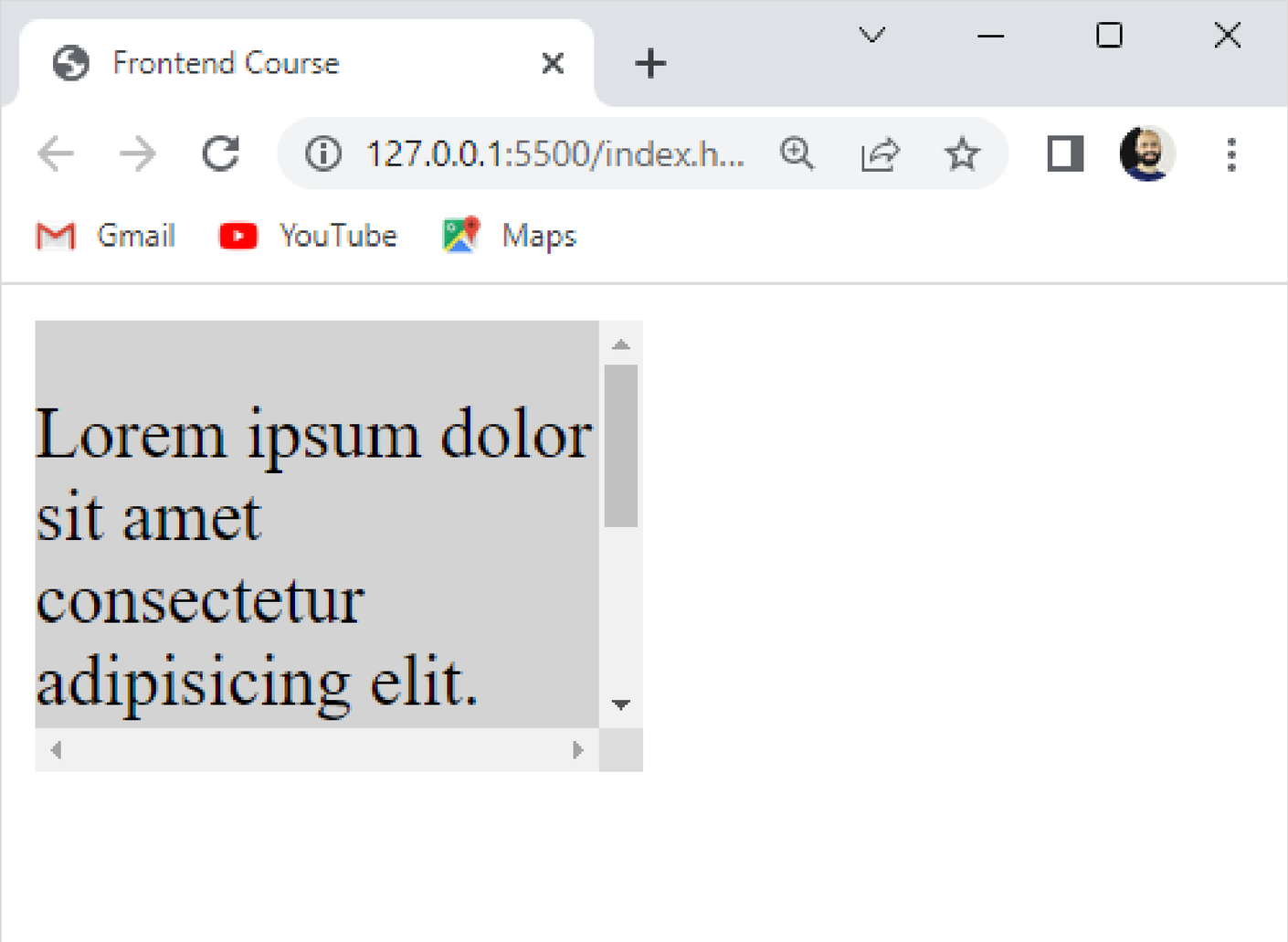
```
overflow: hidden;
```



Overflow: scroll

- Scroll: Der Überlauf wird abgeschnitten und eine Bildlaufleiste wird hinzugefügt, um den Rest des Inhalts zu sehen

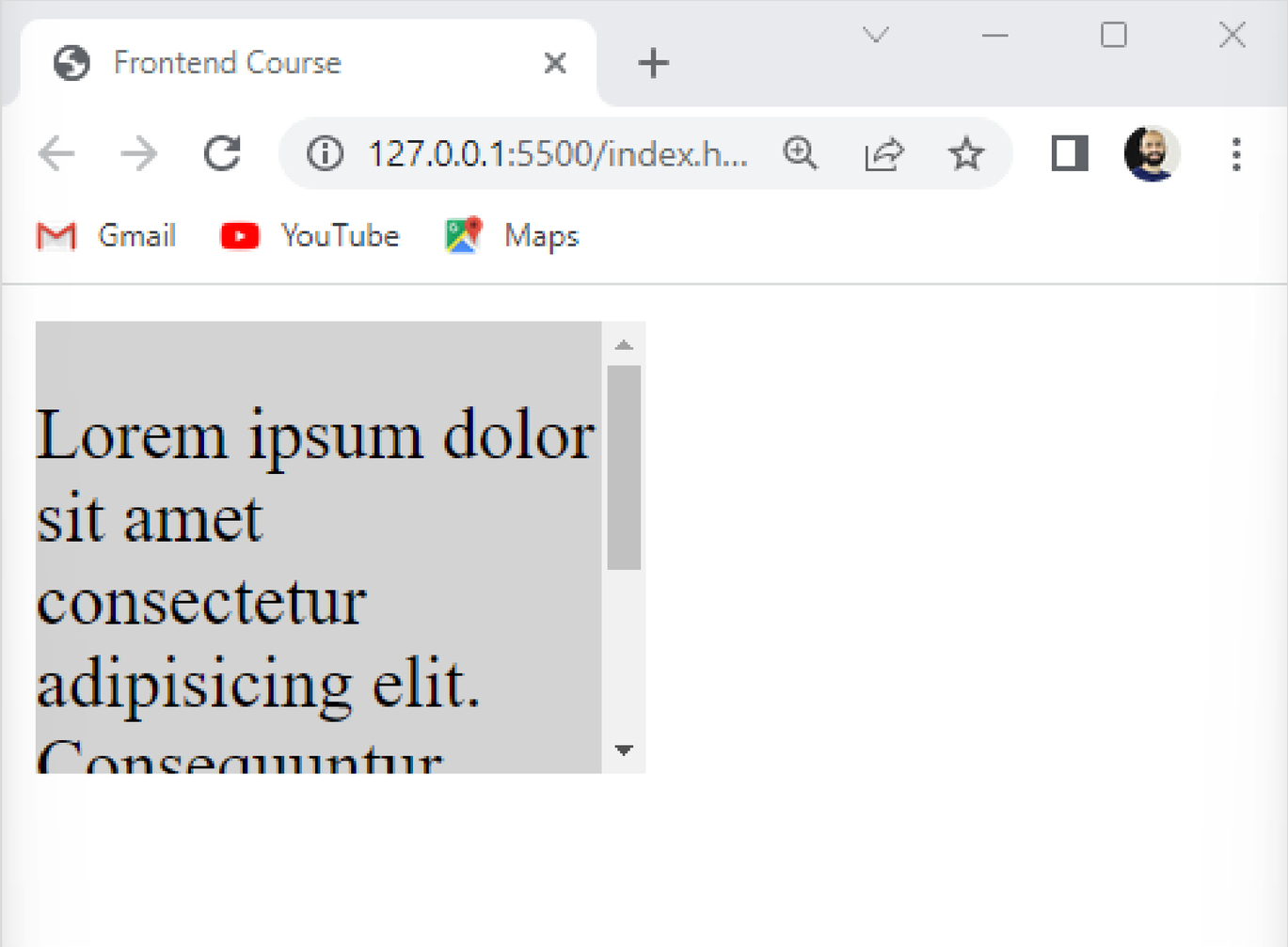
```
overflow: scroll;
```



Overflow: auto

- Auto: Ähnlich wie **scroll**, aber es fügt Bildlaufleisten nur bei Bedarf hinzu

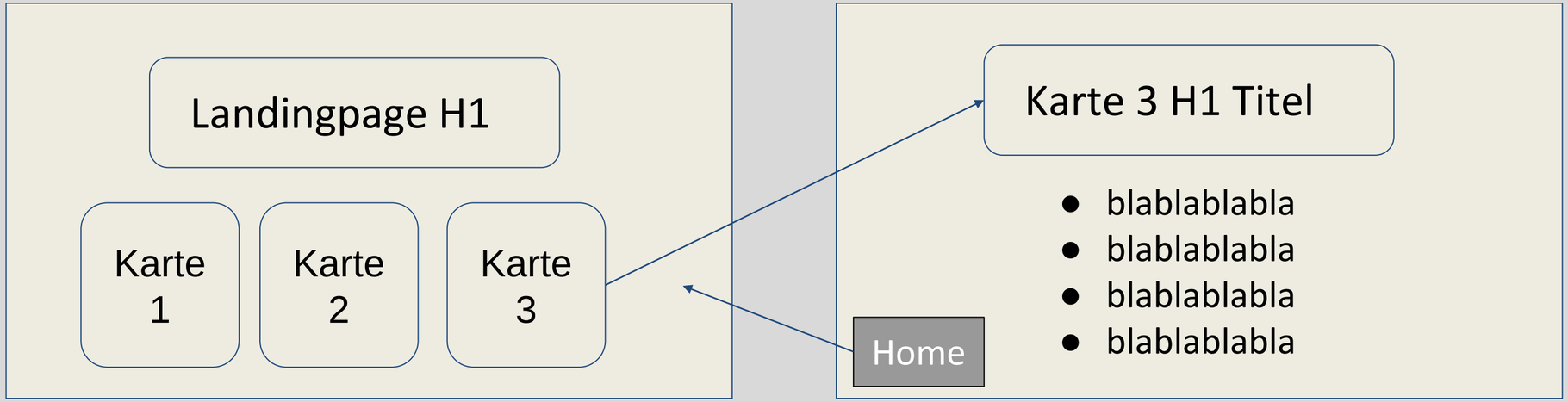
```
overflow: auto;
```



Übung: Places als Multipage

Erstelle eine Multipage Webseite :

1. Vorlage https://www.figma.com/file/SauSOcwUPSyqo7njUUrbf1/Places?node-id=0%3A1
2. Erstelle eine Projektstruktur mit Unterordner für CSS Dateien und Bilddateien
3. Auf der Landingpage müssen mind. 3 Karten sein.
4. Jeder Klick auf eine Karte führt zu einer neuen Detailseite, von ihr kann man auch wieder zurück zur Landingpage.



CSS Float

Float: Intro

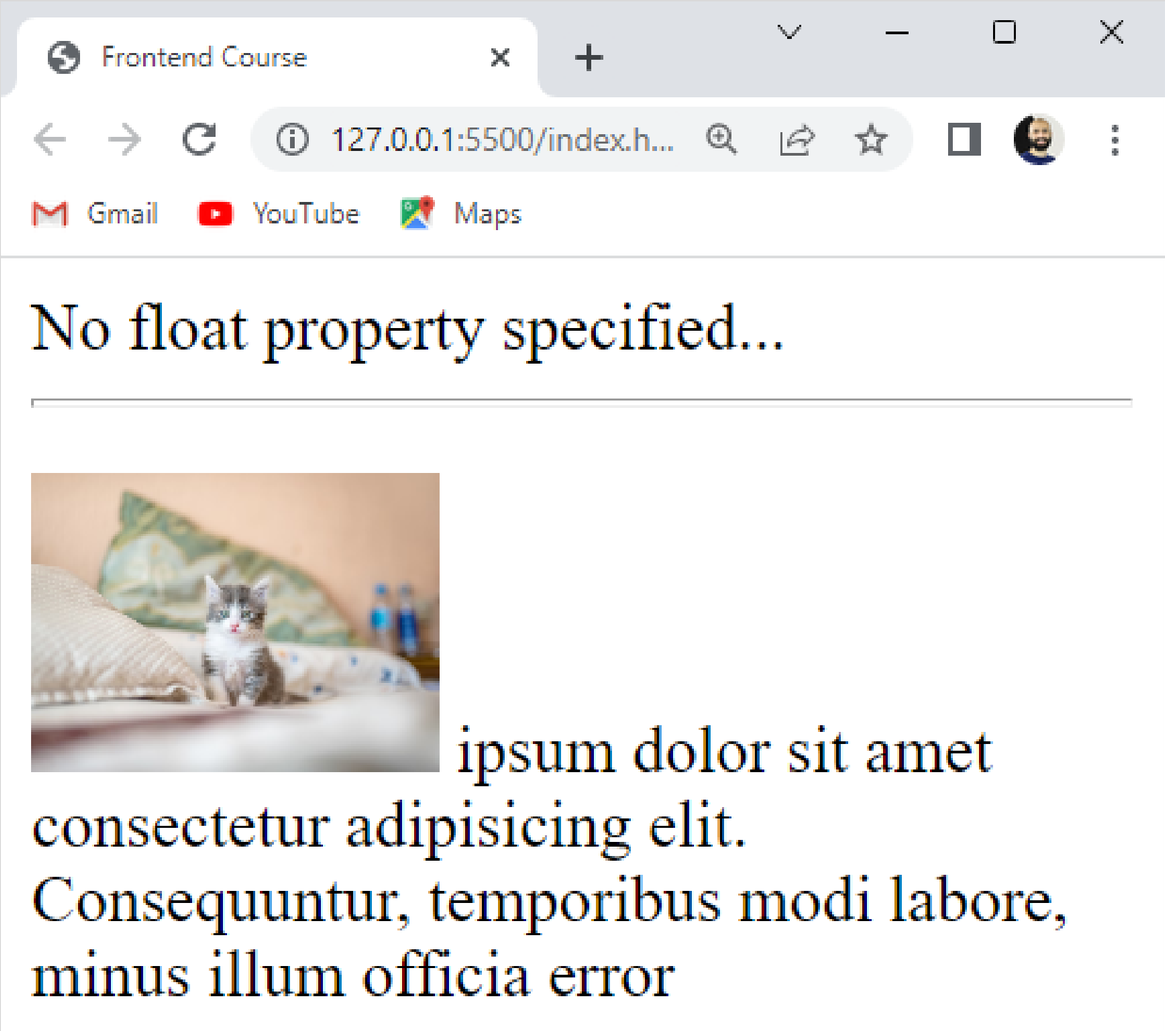
Die **float**-Eigenschaft wird für die Positionierung und Formatierung von Inhalten verwendet, z. B. um ein Bild links vom Text in einem Container schweben zu lassen.

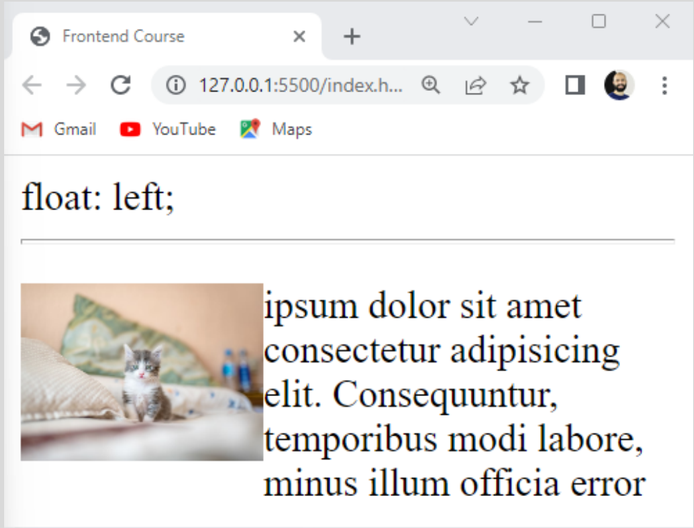
Werte:

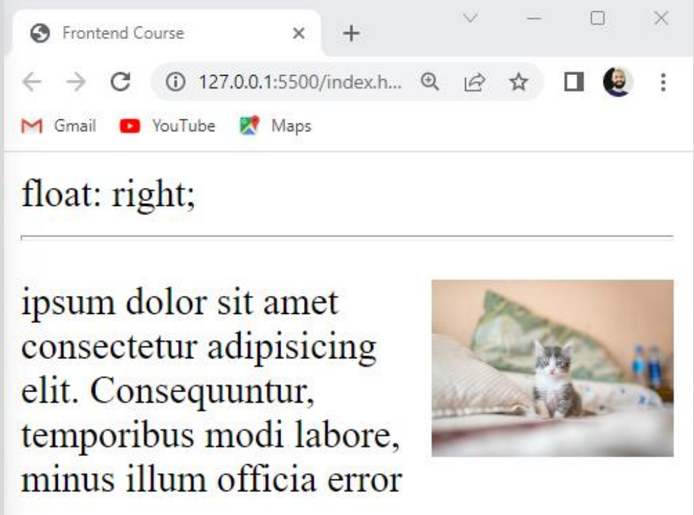
- left: Das Element schwebt links von seinem Container
- right: Das Element schwebt rechts von seinem Container
- none: Das Element ist nicht frei beweglich (es wird nur an der Stelle im Text angezeigt). Dies ist der Standard
- inherit: Das Element erbt den Float-Wert seines Elternteils

In ihrer einfachsten Form kann die **float**-Eigenschaft verwendet werden, um Text um Bilder herumzulegen.

Float


Frontend Course
127.0.0.1:5500/index.h...
Gmail
YouTube
Maps
No float property specified...
ipsum dolor sit amet consectetur adipisicing elit. Consequuntur, temporibus modi labore, minus illum officia error


Frontend Course
127.0.0.1:5500/index.h...
Gmail
YouTube
Maps
float: left;
ipsum dolor sit amet consectetur adipisicing elit. Consequuntur, temporibus modi labore, minus illum officia error

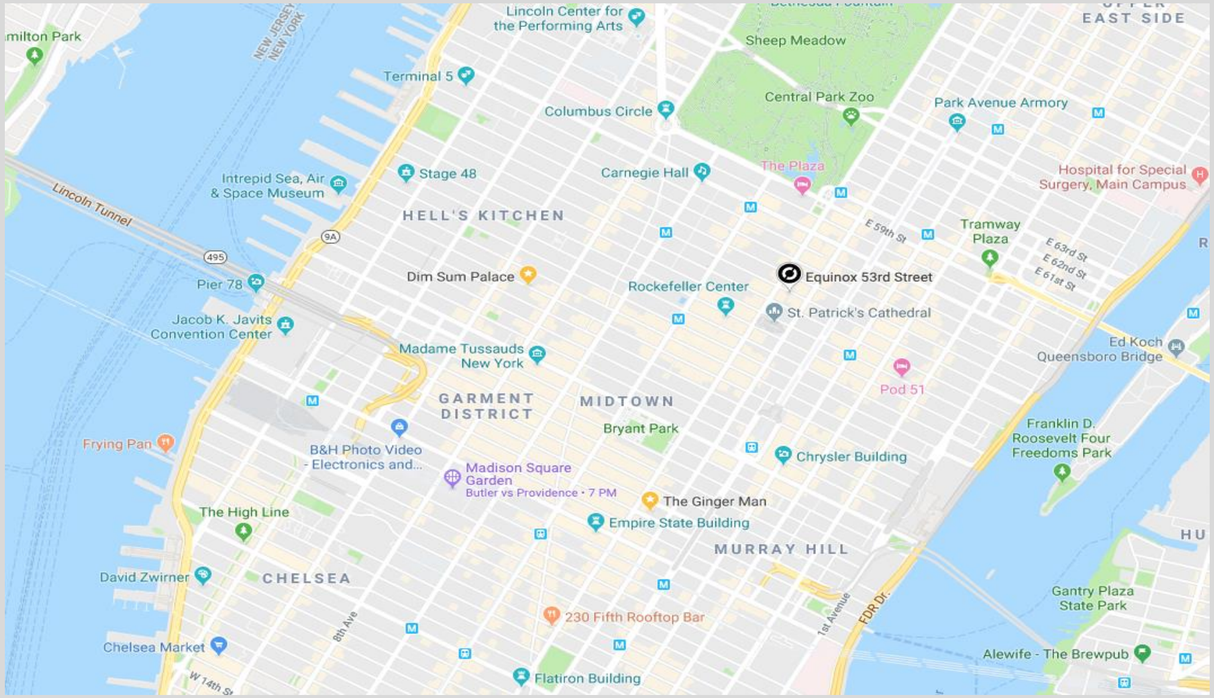

Frontend Course
127.0.0.1:5500/index.h...
Gmail
YouTube
Maps
float: right;
ipsum dolor sit amet consectetur adipisicing elit. Consequuntur, temporibus modi labore, minus illum officia error

CSS Media Query

Responsive design: Grundlagen

Damit die Inhalte einer Webseite ***schön angeordnet*** werden, benutzen Webdesigner oft ein ***Gridsystem***. Ein Gridsystem besteht aus horizontalen und vertikalen Linien.

Es bietet Konsistenz und lässt inhaltlich zusammengehörende Komponenten leicht wiederverwenden. Somit ***lernt der Nutzer***, wie die Webseite aufgebaut bzw. zu verwenden ist wesentlich schneller, als wenn er sehr lange braucht, um sich auf einer Seite zu ***orientieren***.

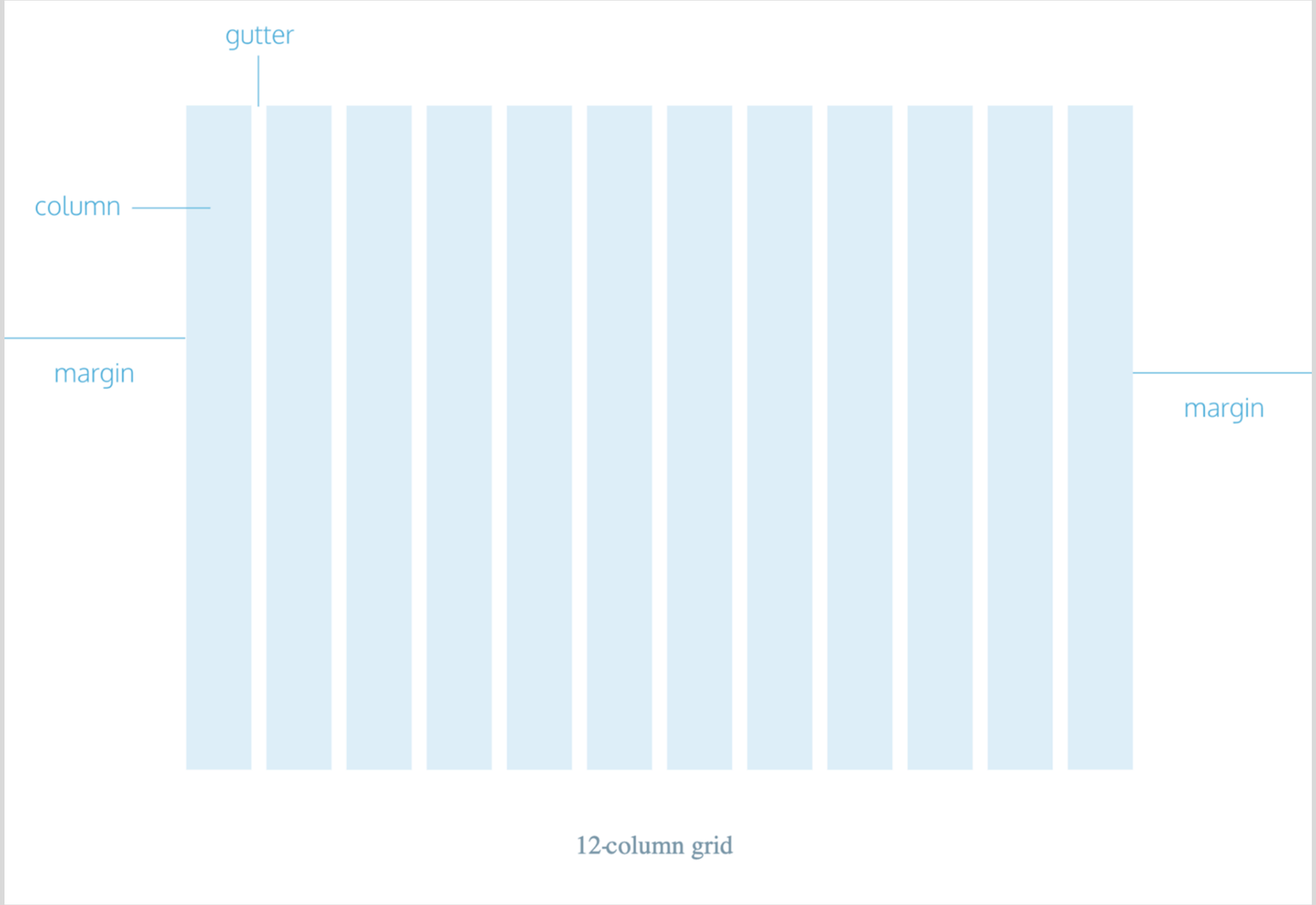


Responsive design: Grundlagen

Es gibt verschiedene Grid-Typen. Die Zeitungsbranche hat das **column-grid** erfunden, um die Inhalte zu positionieren.

Für Webseiten wird gerne das **12-column-grid** verwendet, um Inhalte mit gleichen Abständen sauber darzustellen.

Responsive design: Grundlagen



Responsive design: Grundlagen

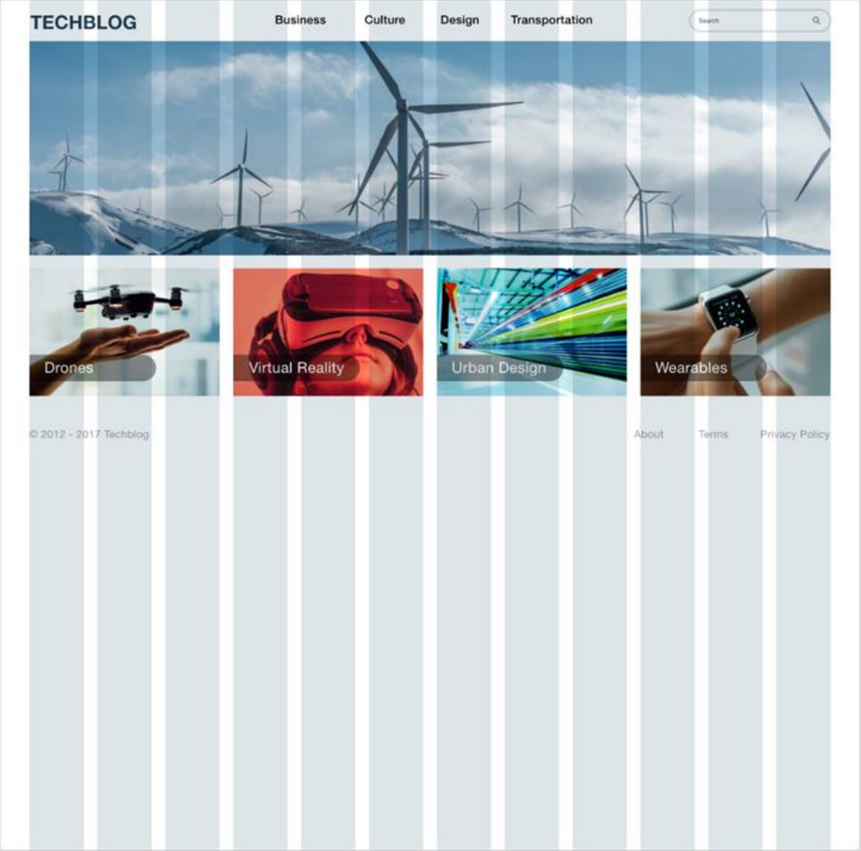
Inhalte einer Webseite werden von verschiedenen Geräten aus angeschaut. Folgende drei Gerätetypen werden unterschieden:

Breakpoint	Class	Dimensions
X-Small	*None*	< 576px
Small	`sm`	≥ 576px
Medium	`md`	≥ 768px
Large	`lg`	≥ 992px
Extra large	`xl`	≥ 1200px
Extra extra large	`xxl`	≥ 1400px

Quelle: Bootstrap 5.0 Breakpoints

Responsive design: Grundlagen

Desktop



12-column grid

Tablet

6-column grid

Mobile

3-column grid

Media Query

made it possible to define different style rules for different screen sizes (Desktop, tablet, smartphone, etc).

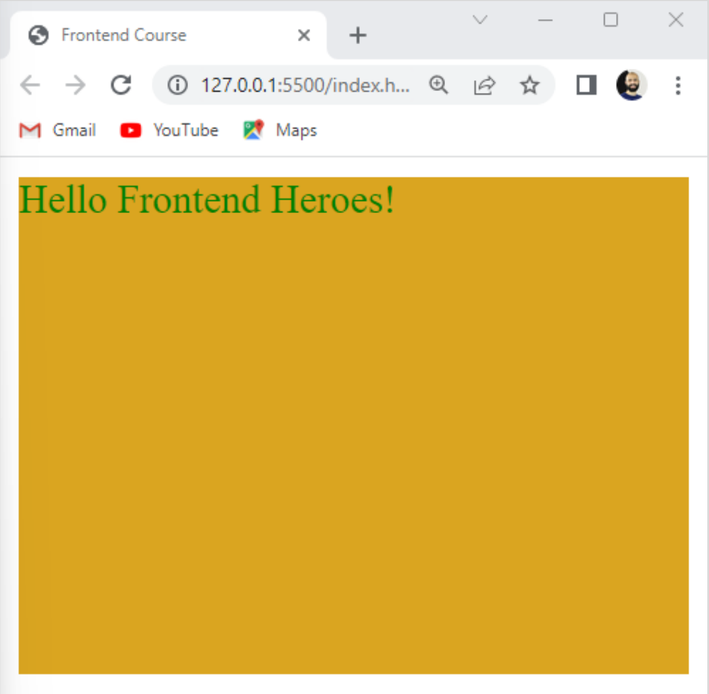
Media queries can be used to check many things, such as:

- width and height of the viewport
- width and height of the device
- orientation (is the tablet/phone in landscape or portrait mode?)
- resolution

```
@media screen and (min-width: 300px) {
  /* do something when the screen's width 300px or bigger */
}
```

Media Query

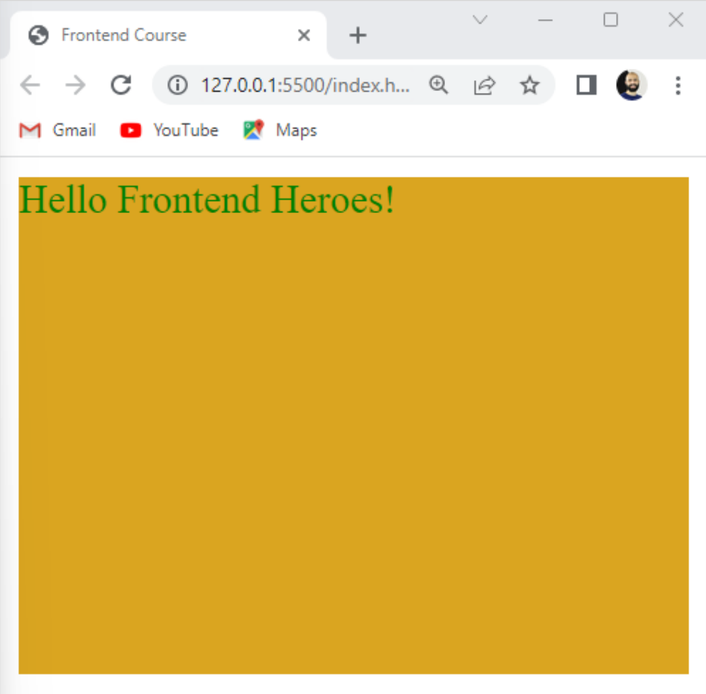
```
width: 250px;
```



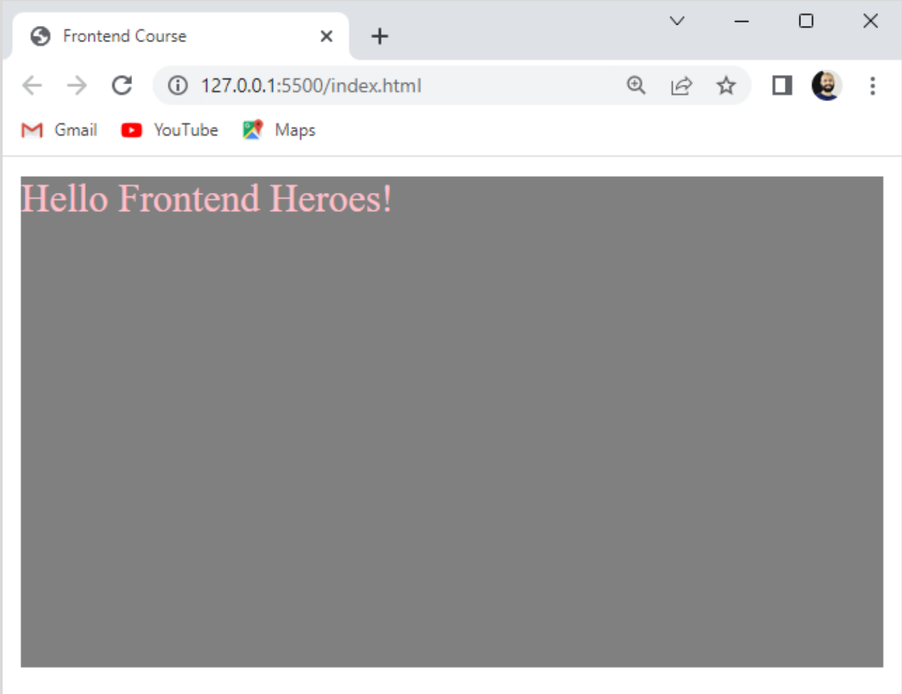

```
@media screen and (min-width: 300px) {
  div {
    background-color: gray;
    color: pink;
  }
}
```

Media Query

```
width: 250px;
```



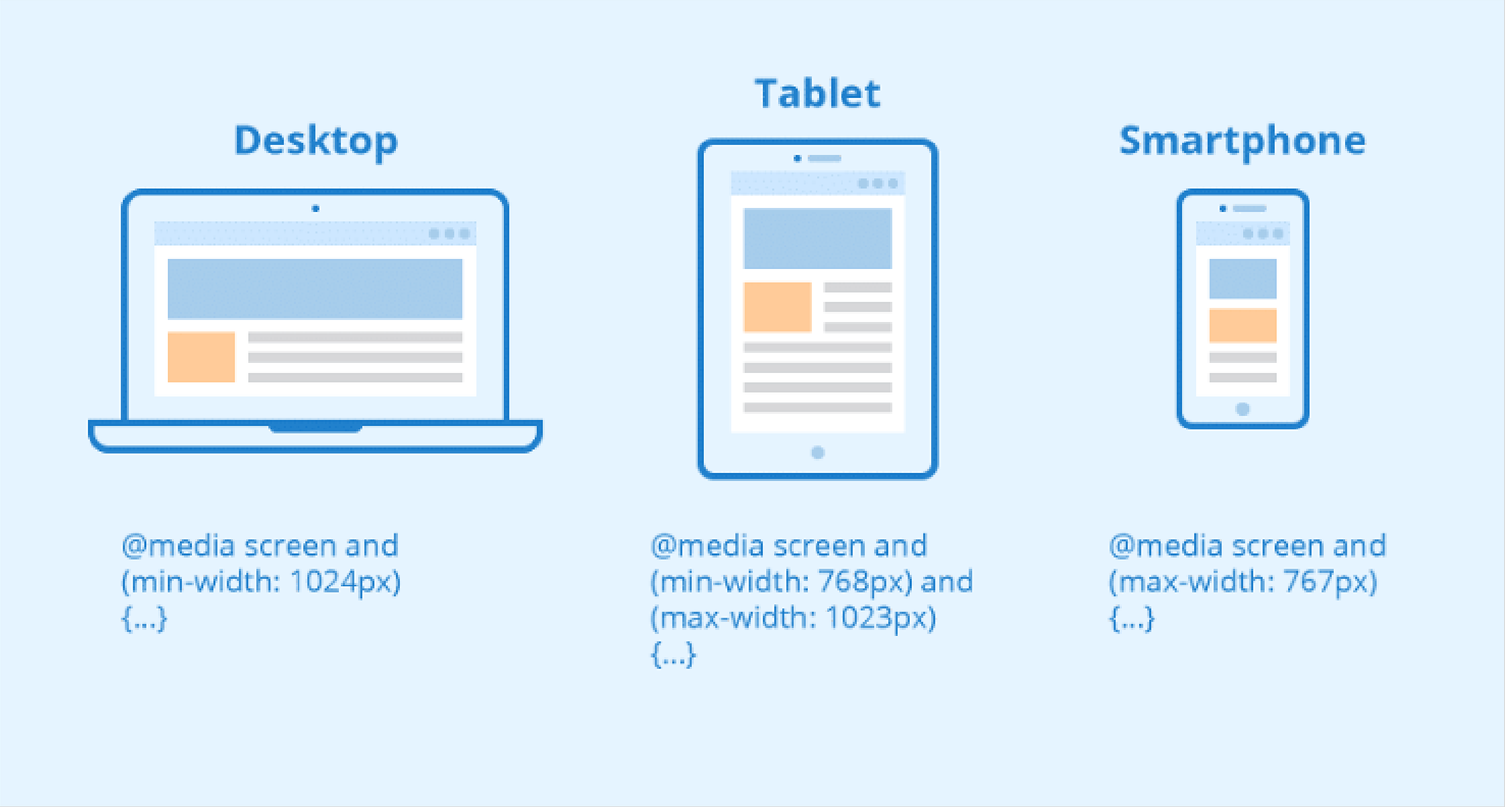

```
width: 320px;
```



Media Query

The main usage of the media query is to change the layout of website/wep-app from horizental to vertical.

NOTE:
It's recommended to build your project with "mobile first" in mind.



Picture: kinsta.com

MEDIA QUERIES: Grundlagen

Um Inhalte auf verschiedenen Geräten optimal darzustellen, kann man CSS Media-Queries verwenden. Diese bestehen aus dem *@media keyword* und *media feature*:

media keyword

media feature

```
@media(min-width: 768px) {
    p {
        color: blue;
    }
}
```

CSS Regel

MEDIA QUERIES: Anwendungen

Die Gerätegröße kann entweder als Mindestgröße, Maximalgröße oder Zwischengröße angegeben werden:

- **@media(min-width: 768px)**
  => Styles werden auf Gerätegrößen **mit mindestens 768px** angewendet, d.h. ab Tablet-Größen

- **@media(max-width: 768px)**
  => Styles werden auf Gerätegrößen **bis maximal 768px** angewendet, d.h. bis Tablet-Größen

- **@media(min-width: 768px) and (max-width: 1200px)**
  => Styles werden auf Gerätegrößen **zwischen 768px und 1200px** angewendet.

Media queries: mobile-first Ansatz

Ein üblicher Ansatz, um gerätespezifische Styles anzuwenden ist der Mobile-First Ansatz. Hierbei geht man wie folgt vor:

CSS

```
p {
    font-size: 16px;
    color: green;
}

@media(min-width: 768px) {
    p {
        color: blue;
    }
}

@media(min-width: 1200px) {
    p {
        color: red;
    }
}
```

1.Global

2.Mobile

3.Tablet

4.Desktop

Übung: media-Queries

1. Erstelle eine **Box** mit **Höhe und Breite von 200px** und **Hintergrundfarbe rot**. **Ab 768px-Viewport** soll der **Hintergrund grün** werden, **ab 920px-Viewport** soll der **Hintergrund blau** werden.

2. Erstelle einen **Text** mit **Schriftgröße 16px**. **Zwischen 768px und 980px Viewport** wird die **Schriftgröße 20px** und der Text ist in **Großbuchstaben**.

3. Erstelle eine **Box** mit **Höhe 200px** und einem **roten Rahmen von der Dicke 1px**. Die Box soll **bis maximal 500px-Viewport** nur **200px breit** sein.

4. Erstelle die **gleiche Box von Aufgabe 3** mit dem **Mobile-First Ansatz**.

5. Erstelle ein **Eltern-Element** und **4 Kind-Elemente**. Die Kind-Elemente sind **150px breit und hoch**, haben einen **grauen Rahmen** und einen **grünen Hintergrund**. Die **Kind-Elemente** sollen auf **kleinen Geräten** in einer **Spalte** angezeigt werden. **Ab 920px-Viewport** sollen sie **in eine Reihe**.

CSS Flexbox

Video Quelle:

CSS Flexbox in 5 Minuten erklärt | CSS Tutorial Deutsch

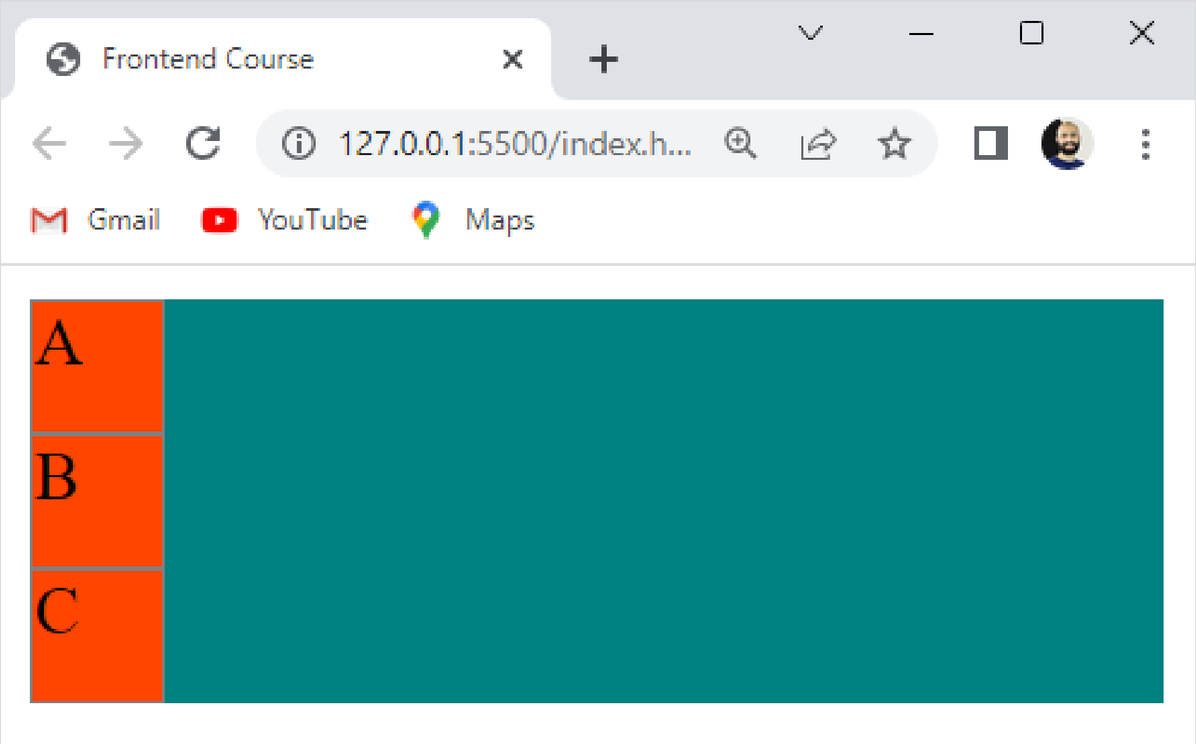
https://www.youtube.com/watch?v=dD8kgEOw1To

Flexbox/flex

Flex/flexbox:Erleichtert die Gestaltung flexibler responsiver Layout-Strukturen ohne Verwendung von float oder Positionierung.

- Um das Flexbox-Modell zu verwenden, müssen Sie zunächst einen Flex-Container definieren, indem Sie dem "div containter oder dem übergeordneten Element" display: flex; zuweisen.

```
<div class="container">
  <div class="child a">A</div>
  <div class="child b">B</div>
  <div class="child c">C</div>
</div>
```



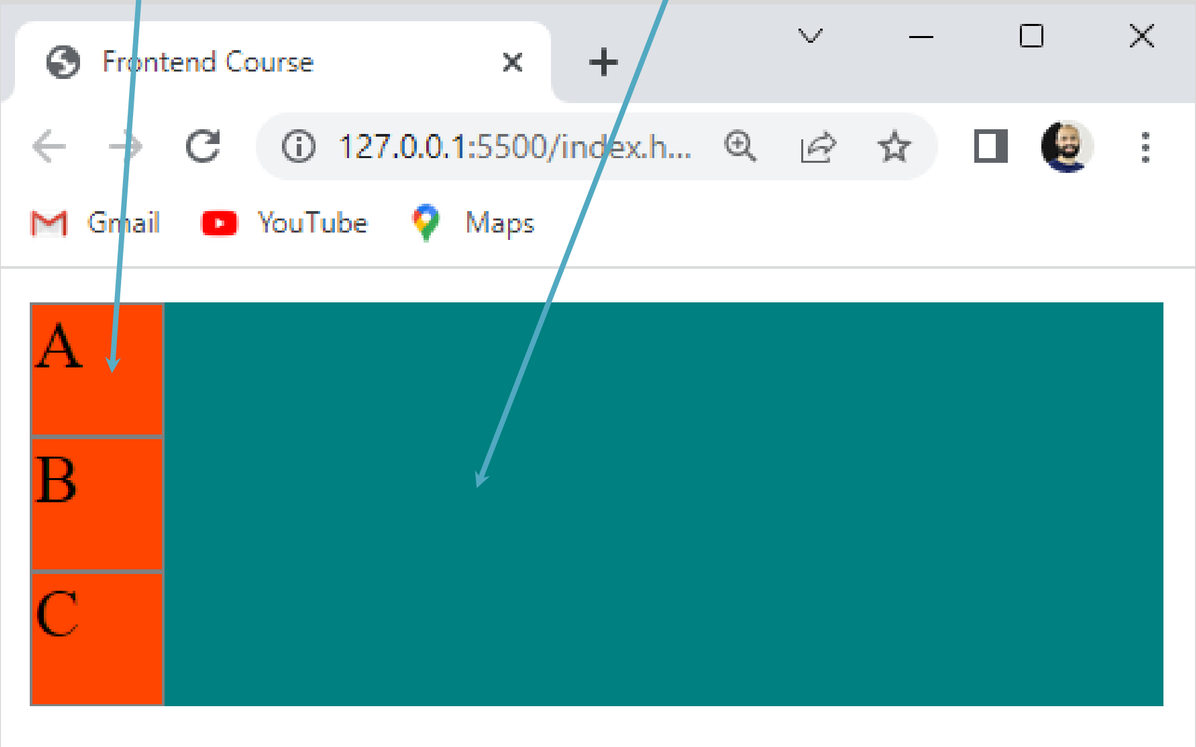
Flexbox/flex

flex item

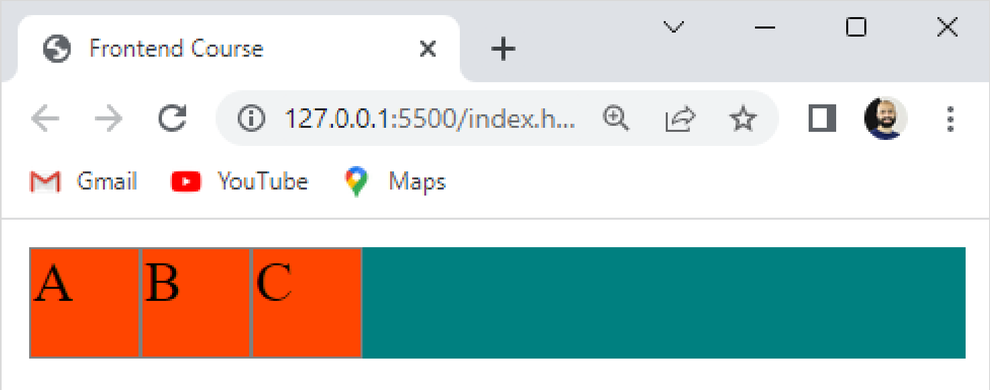
"child"

flex container

"parent"




```
.container {
  display: flex;
  background-color: teal;
}
```



CSS Flexbox

Parent/container element

Flexbox: parent element

Der Flex-Container wird flexibel, indem die Eigenschaft display auf flex gesetzt wird:

Flex properties:

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

Flexbox grundlagen

Flexbox ist eine CSS-Methode, das die Positionierung der HTML-Elemente vereinfacht. Flexbox = flexible box layout.

Es arbeitet mit zwei wichtigen Komponenten: **flex container** und **flex item**. Alle direkten Kind-Elemente eines flex containers sind flex items. Alle HTML-Elemente können ein flex container sein.

HTML

```html
<div class="parent">
    <p>Ich bin ein Sohn.</p>
    <p>Ich bin eine Tochter.</p>
</div>
```

CSS

```css
.parent {
    display: flex;
}
```

Flexbox css-eigenschaft: display

Um ein HTML-Element zu einem ***flex container*** zu machen, wird die CSS-Eigenschaft ***display*** angewendet.

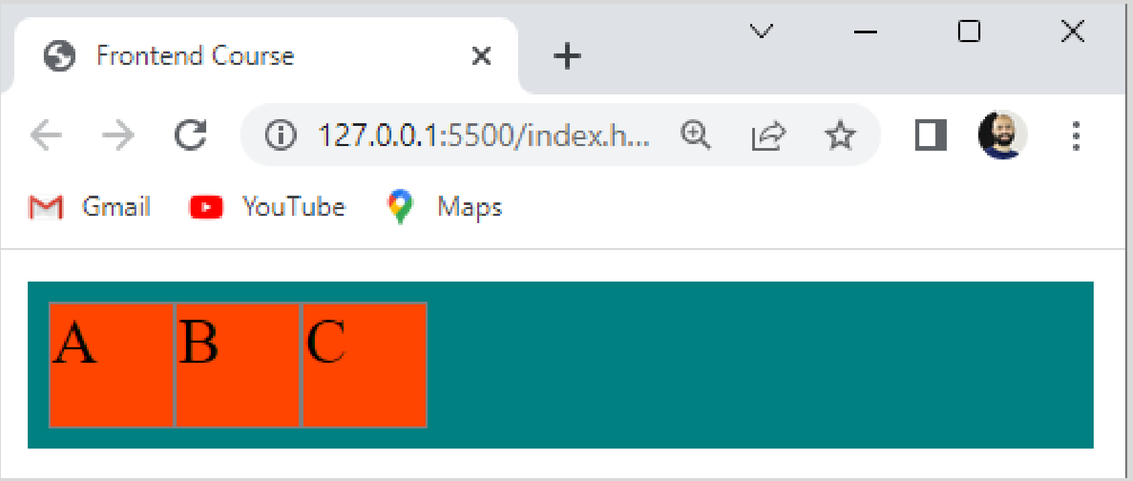
CSS-Werte:

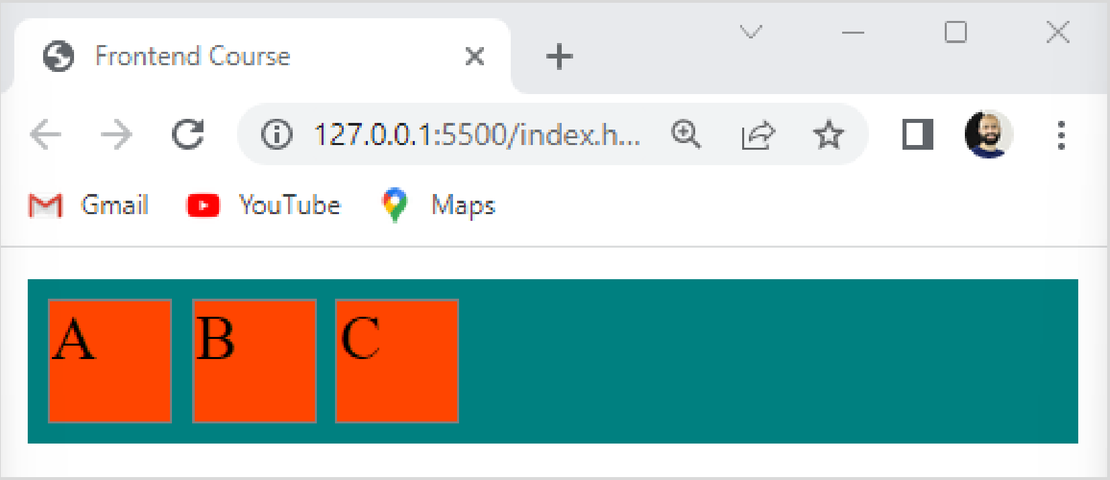
- ***flex*** => flex container als Block-Element
- ***inline-flex*** => flex container als Inline-Element

Flex parent: gap

```
.container {
  padding: 5px;
  background-color: teal;
  display: flex;
}
```

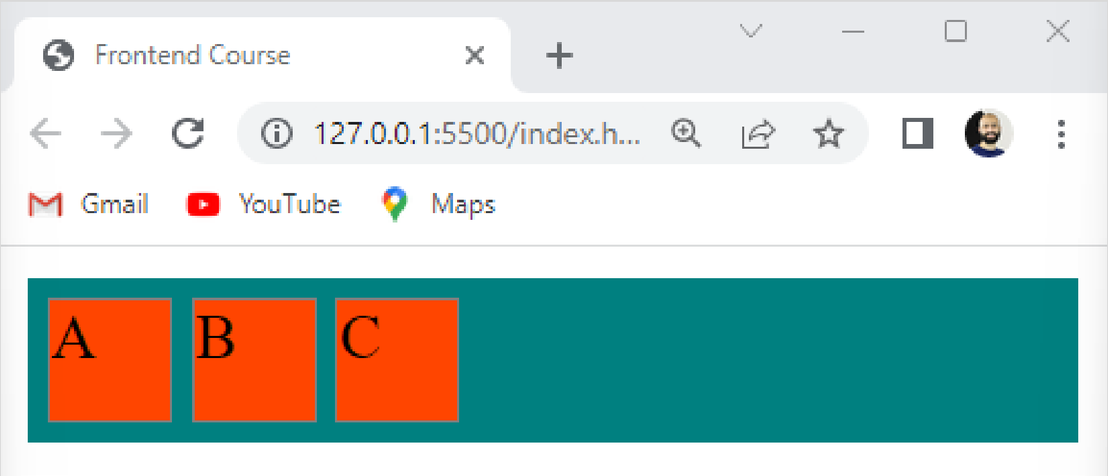
```
.container {
  padding: 5px;
  background-color: teal;
  display: flex;
  gap: 5px;
}
```



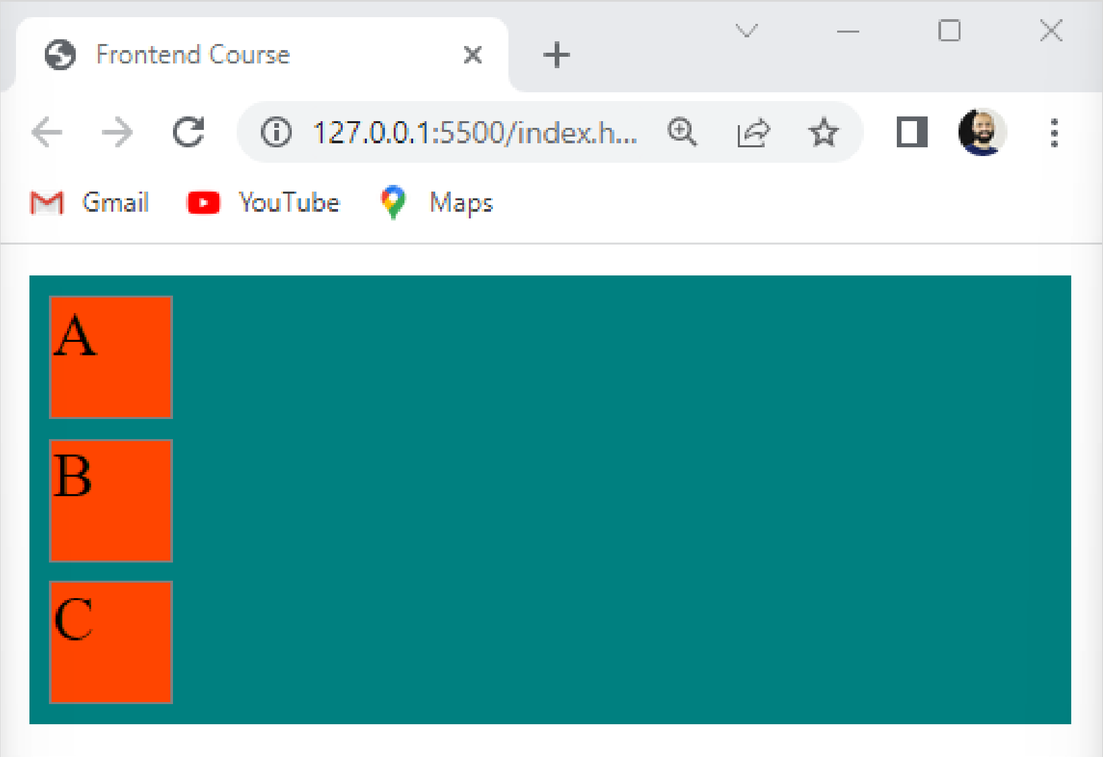


Flex parent direction

```
.container {
  padding: 5px;
  background-color: teal;
  display: flex;
  gap: 5px;
}
```



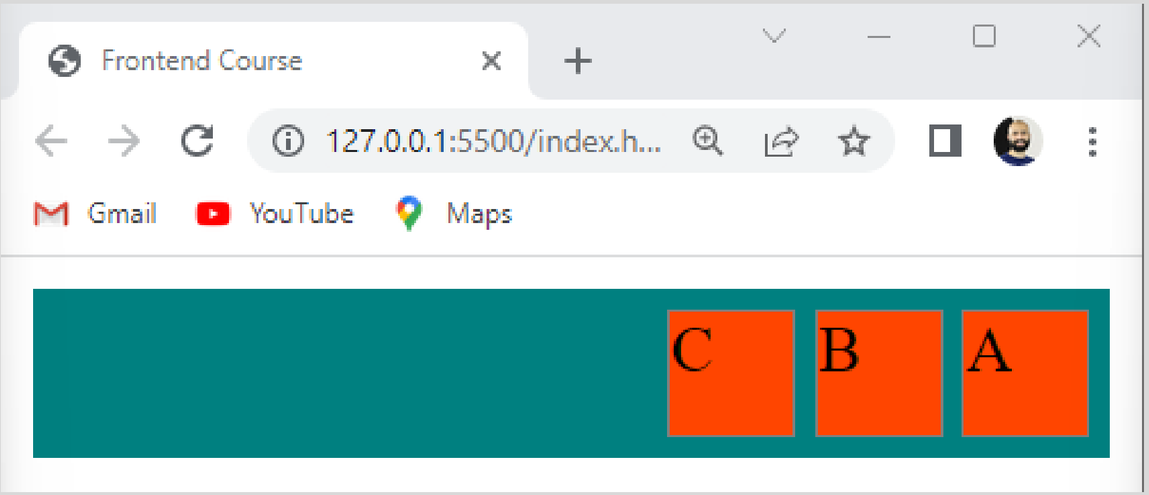

```
flex-direction: column;
```

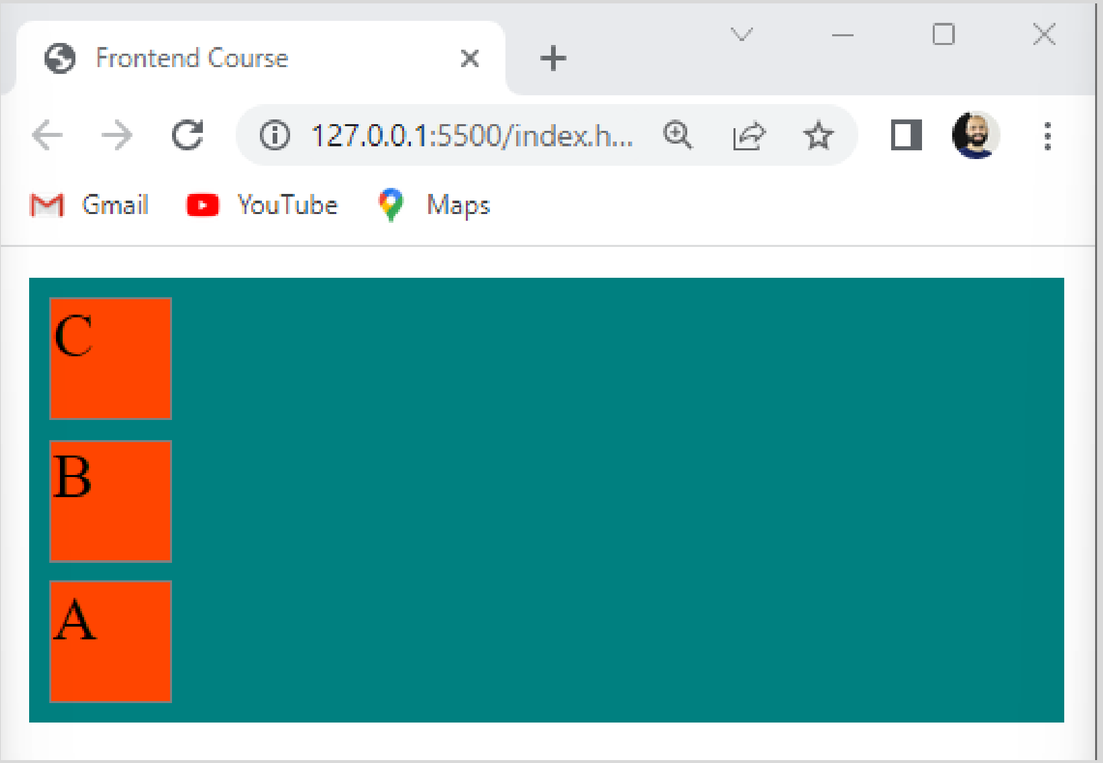


Flex parent: direction

```
flex-direction: column-reverse;
```

```
flex-direction: row-reverse;
```





Flexbox css-eigenschaft: flex-direction

Um die ***Anordnungs-Richtung*** der ***flex items*** zu bestimmen, wird die CSS-Eigenschaft ***flex-direction*** auf den flex container angewendet.

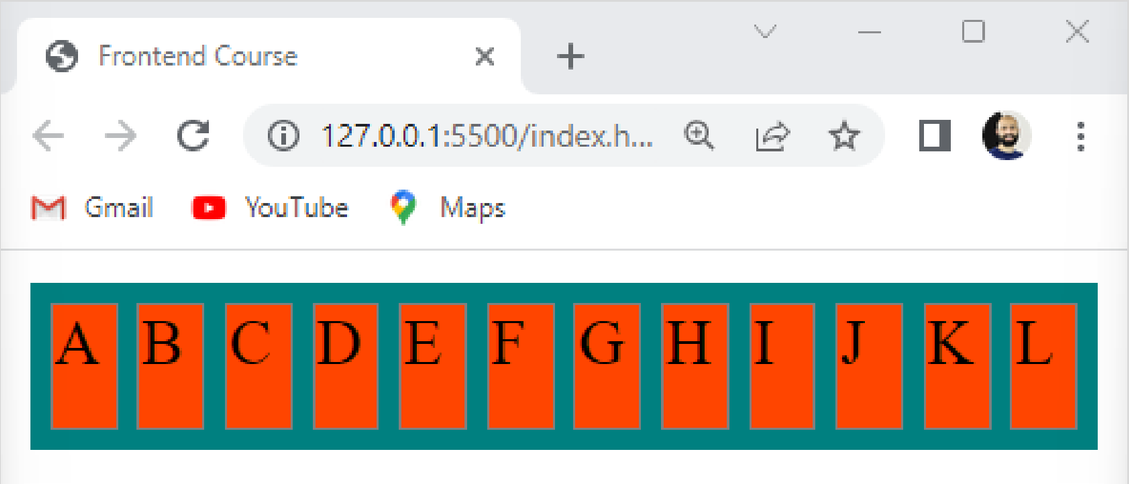
CSS-Werte:

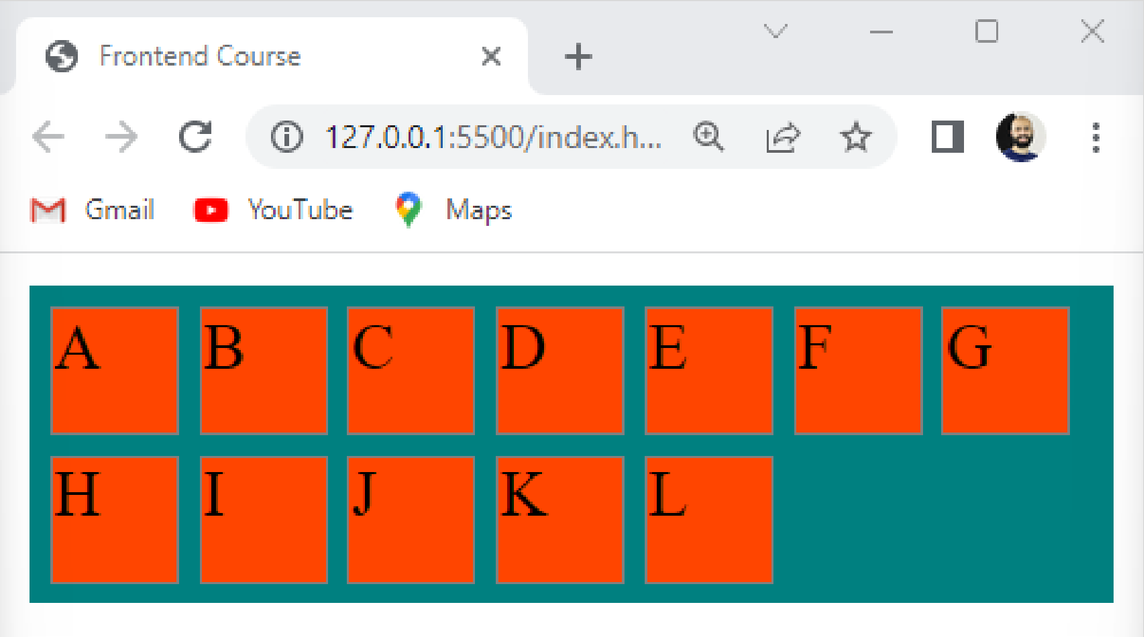
- ***row*** => flex items von links nach rechts anordnen
- ***column*** => flex items von oben nach unten anordnen

Flex parent: flex-wrap

```
flex-wrap: wrap;
```

```
flex-wrap: nowrap; /*default*/
```

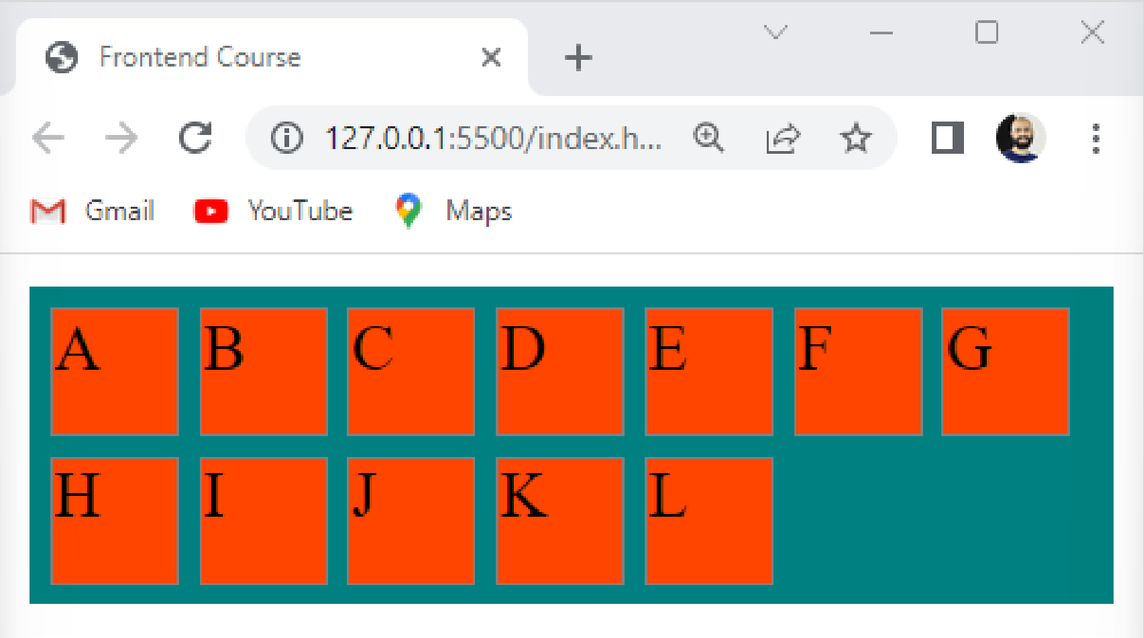




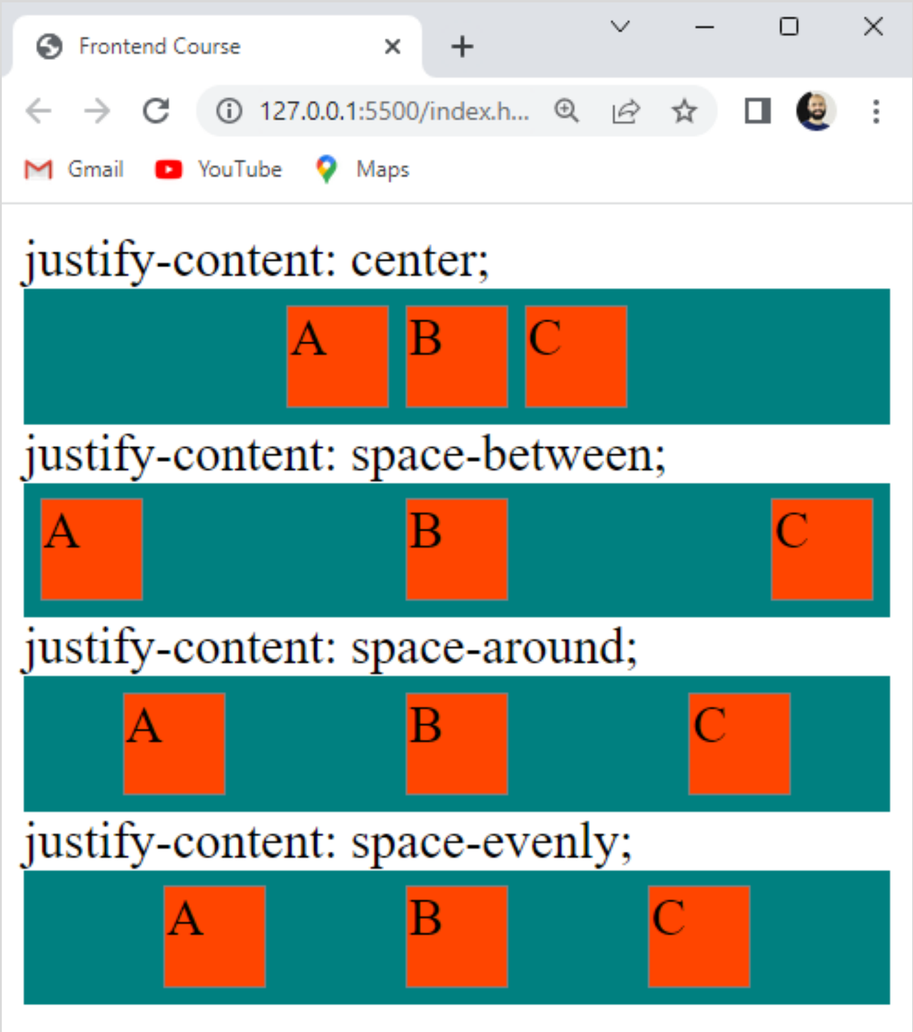
Flex parent: flex-flow

Die Eigenschaft flex-flow ist eine Kurzform für die Einstellung der Eigenschaften flex-direction und flex-wrap.

```
flex-flow: row wrap;
```



Flex parent: justify-content



Die Eigenschaft justify-content wird verwendet, um die Flex-Elemente auszurichten:

Werte:
Es gibt viele Werte für justify-content, siehe w3 für weitere Beispiele.

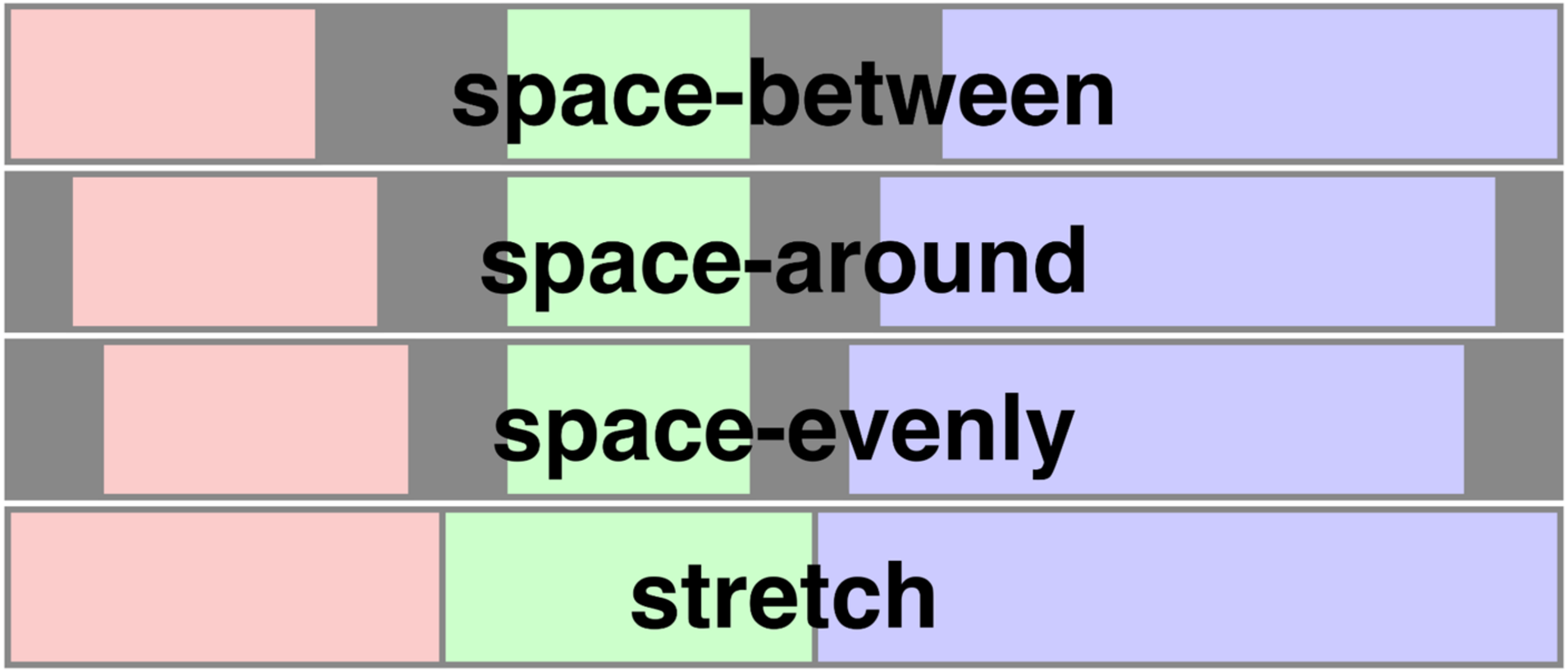


Figure 9 *The distributed alignment values*

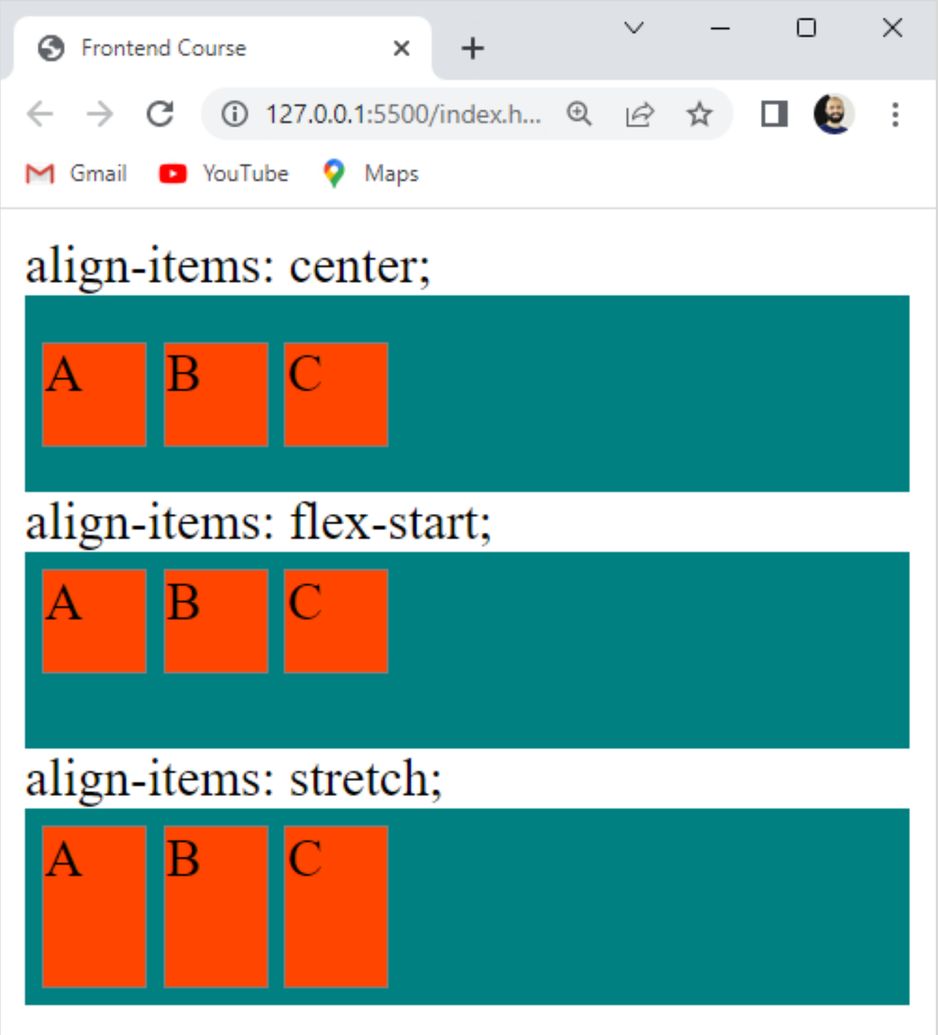
Flexbox css-eigenschaft: Justify-content

Um die ***horizontale Anordnung*** der ***flex items*** zu bestimmen, wird die CSS-Eigenschaft ***justify-content*** auf den flex container angewendet.

CSS-Werte:

- ***flex-start*** => flex items sind linksbünding
- ***flex-end*** => flex items sind rechtsbündig
- ***center*** => flex items sind zentriert
- ***space-around*** => flex items haben gleichmäßigen Abstand
- ***space-between*** => flex items haben gleichmäßigen Abstand ohne den äußersten Abständen

Flex parent: align-items



Die Eigenschaft align-items wird verwendet, um die Flex-Elemente auszurichten:

Werte:
Es gibt viele Werte für align-items, siehe w3 für weitere Beispiele.

Flexbox css-eigenschaft: align-items

Um die ***vertikale Anordnung*** der ***flex items*** zu bestimmen, wird die CSS-Eigenschaft ***align-items*** auf den flex container angewendet.

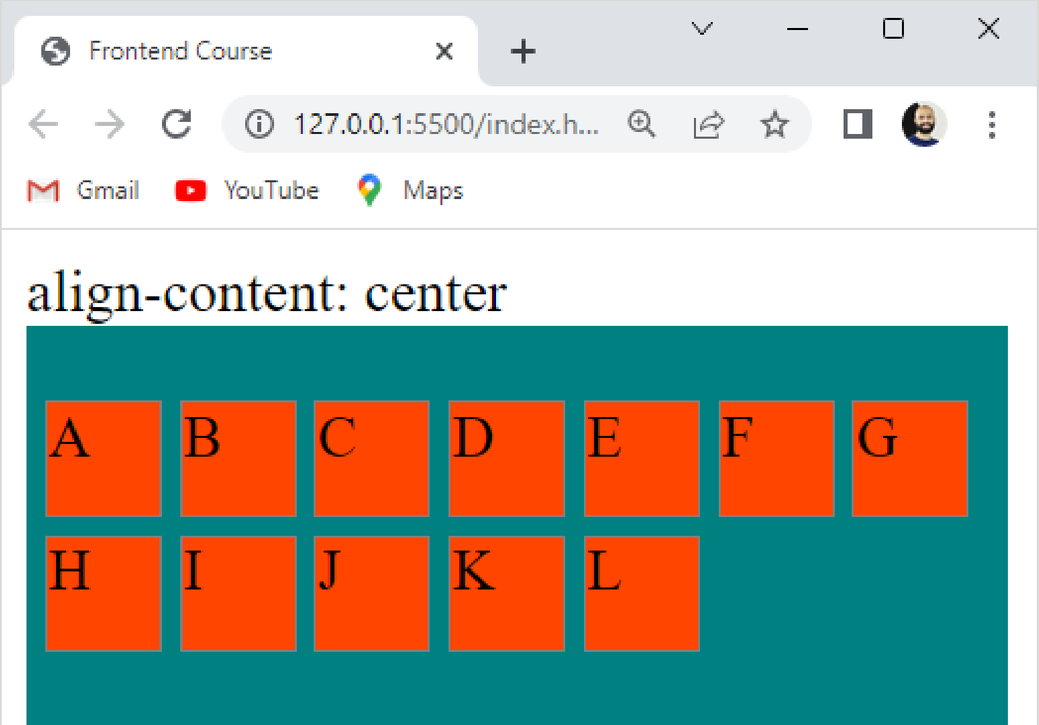
CSS-Werte:

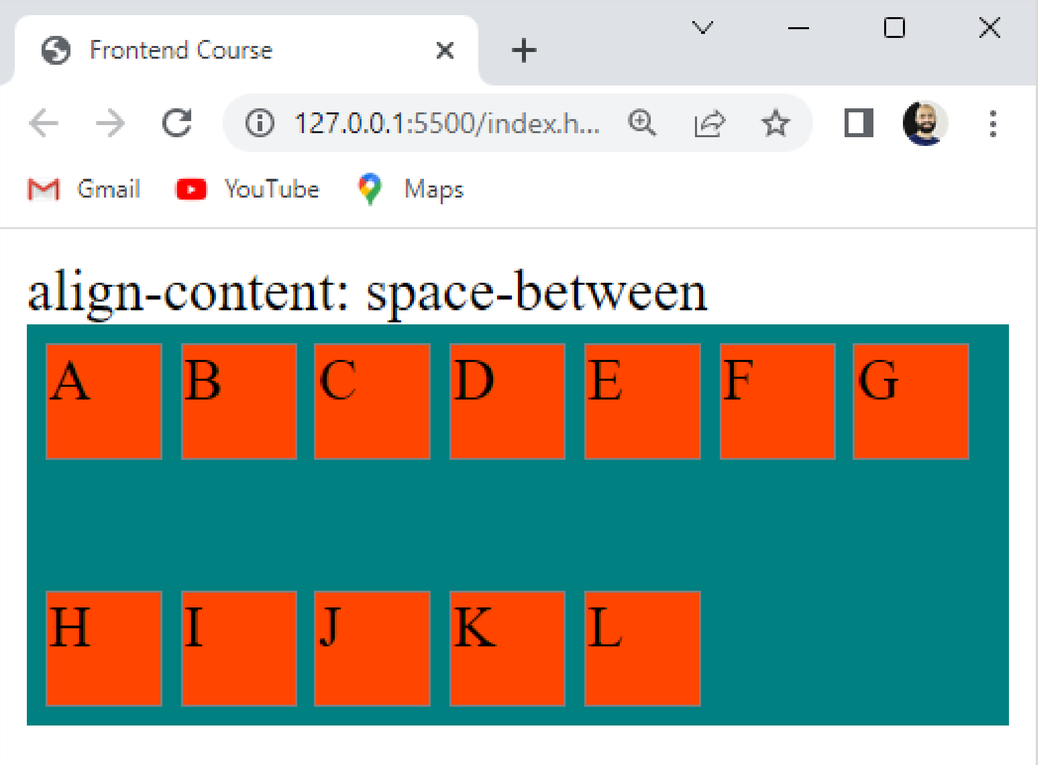
- ***flex-start*** => flex items sind oben bündig
- ***flex-end*** => flex items sind unten bündig
- ***center*** => flex items sind zentriert
- ***baseline*** => flex items sind zentriert, wobei alle Inhalte unten bündig sind
- ***stretch*** => default Verhalten: Inhalte werden von oben nach unten gedehnt (geht nur mit min-height oder ohne height CSS-Eigenschaft)

Flex parent: align-content

Die Eigenschaft align-content wird verwendet, um die Flex-Linien auszurichten.

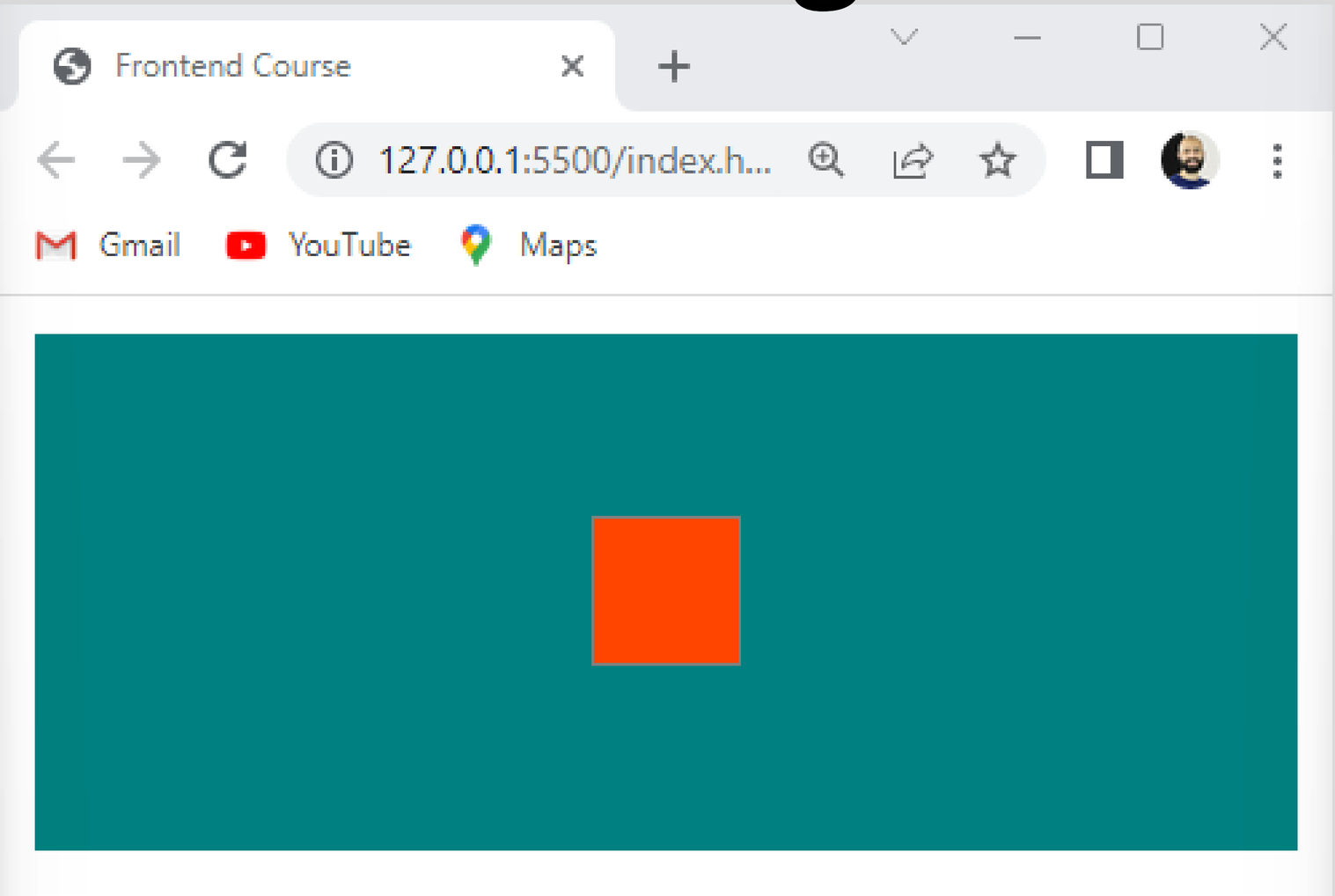
siehe w3 für weitere Eigenschaftswerte.





Flexbox: Perfect Centering

Setzen Sie im Flex-Element "parent, container" sowohl die Eigenschaften justify-content als auch align-items auf center, und das Flex-Element wird perfekt zentriert sein:



Flex:grow/shrink

Other properties for flex child:

- flex-grow
- flex-shrink
- flex-basis

CSS Flexbox

Child/item element

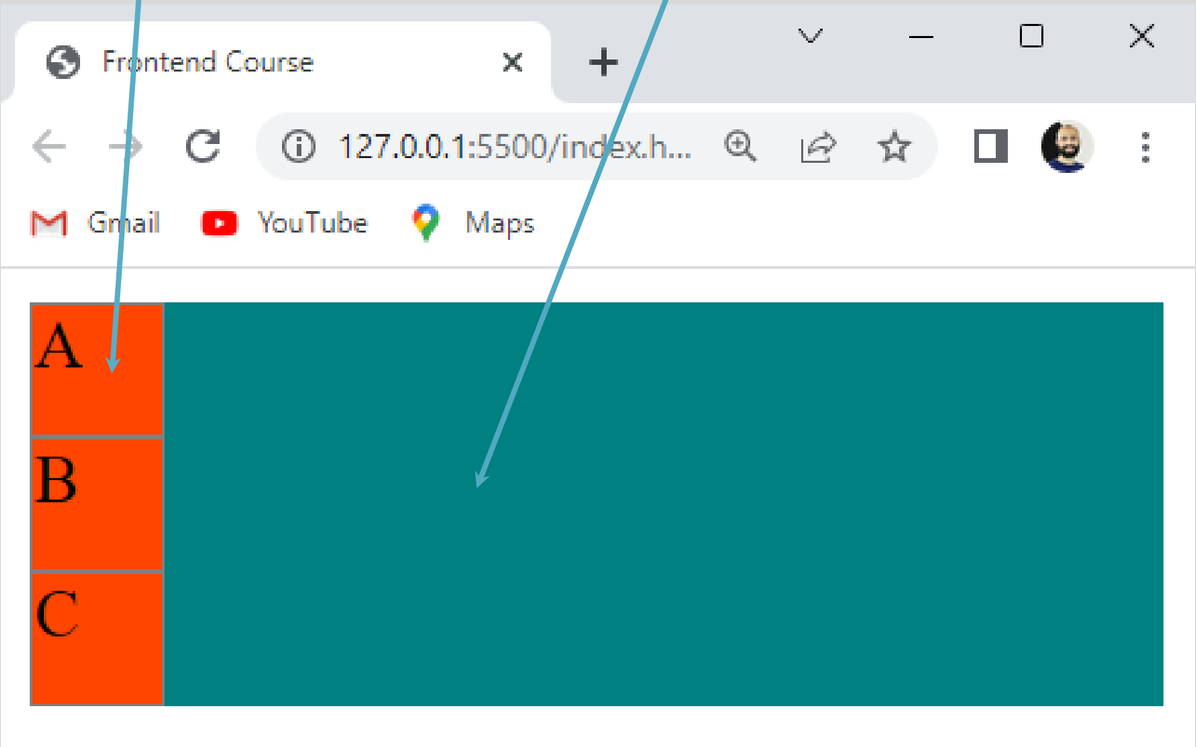
Flexbox/flex

flex item

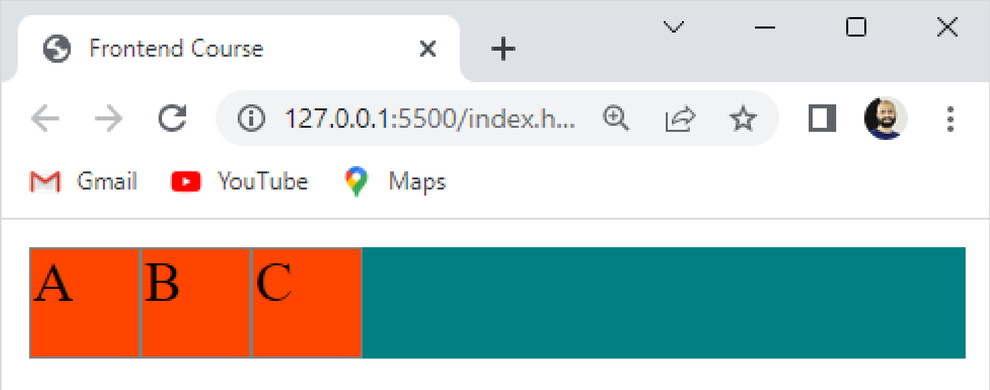
"child"

flex container

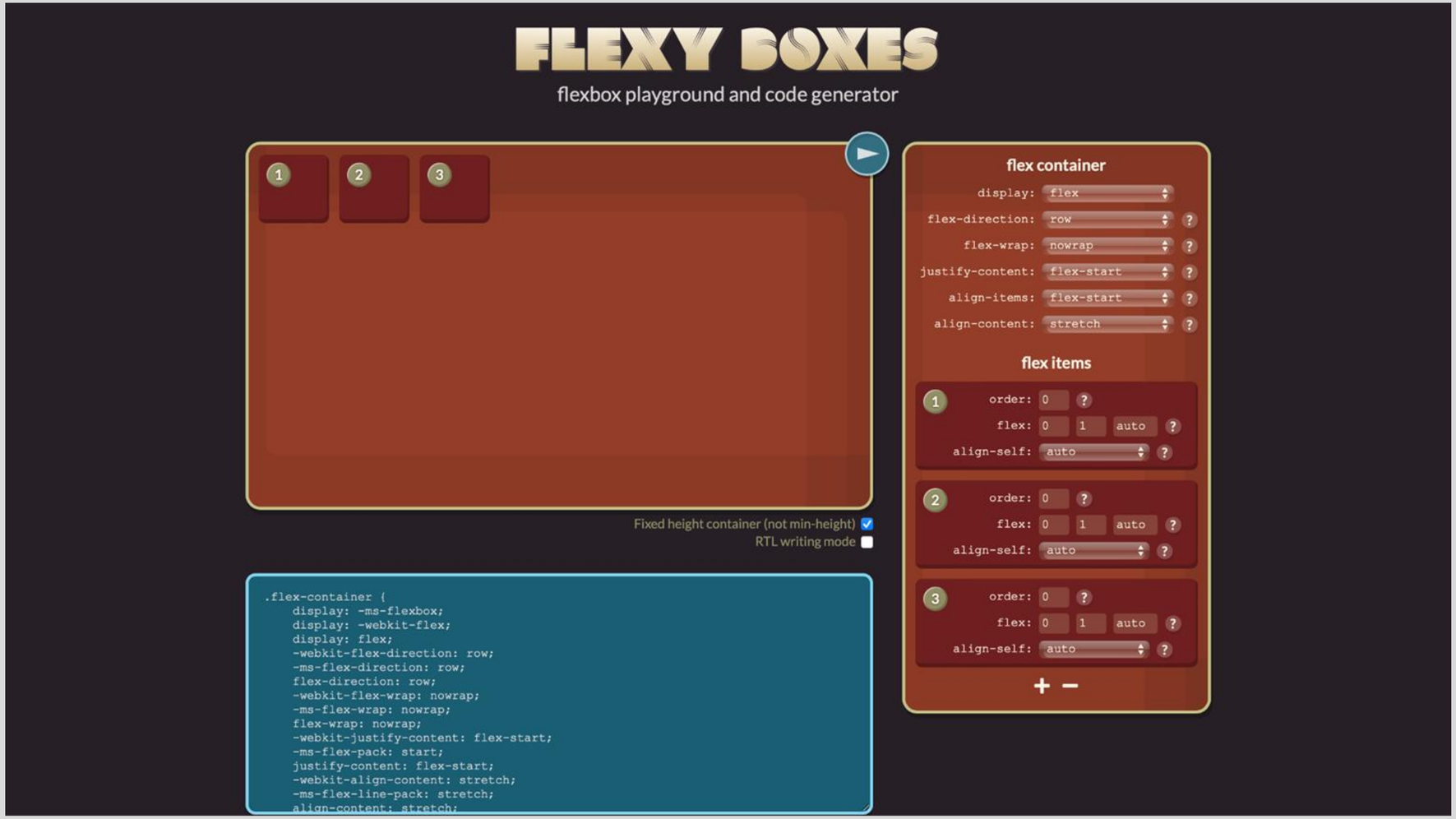
"parent"




```
.container {
  display: flex;
  background-color: teal;
}
```



https://the-echoplex.net/flexyboxes/



Übung: Kochblog

Erstelle eine Webseite :

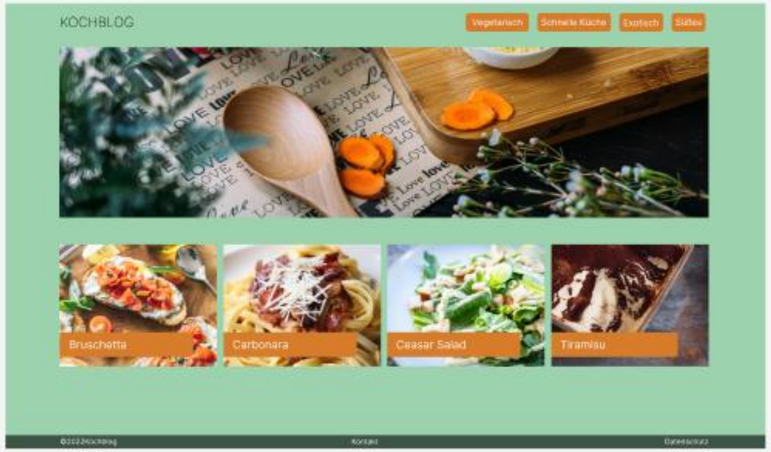
1. Vorlage https://www.figma.com/file/OEc0q1X8mGepg1lcuVca41/Kochblog?node-id=0%3A1
2. Erstelle eine Projektstruktur mit index.html und style.css
3. Verwende den “Mobile First” Ansatz, benutze Semantische Tags, benutze CSS flexbox und media query, um diese Seite “responsive” zu machen.
4. Benutze die “break points” wie in Figma angegeben (350px, 768px, 1200px)
5. Benutze die Bildquellen wie unten angegeben (nächste Seiten)

Übung: Kochblog

ab 350px

ab 768px

ab 1200px



Übung: Kochblog

Bildquellen:

1. https://images.unsplash.com/photo-1495461199391-8c39ab674295?ixlib=rb-1.2.1&ixid=MnwxMjA3fDB8MHxwaG90by1wYWdlfHx8fGVufDB8fHx8&auto=format&fit=crop&w=1770&q=80
2. https://images.unsplash.com/photo-1506280754576-f6fa8a873550?ixlib=rb-1.2.1&ixid=MnwxM[…]G90by1wYWdlfHx8fGVufDB8fHx8&auto=format&fit=crop&w=774&q=80
3. https://images.unsplash.com/photo-1550304943-4f24f54ddde9?ixlib=rb-1.2.1&ixid=MnwxMjA3[…]90by1wYWdlfHx8fGVufDB8fHx8&auto=format&fit=crop&w=1770&q=80
4. https://images.unsplash.com/photo-1627207644206-a2040d60ecad?ixlib=rb-1.2.1&ixid=MnwxM[…]G90by1wYWdlfHx8fGVufDB8fHx8&auto=format&fit=crop&w=774&q=80

Übung: Responsive Portfolio

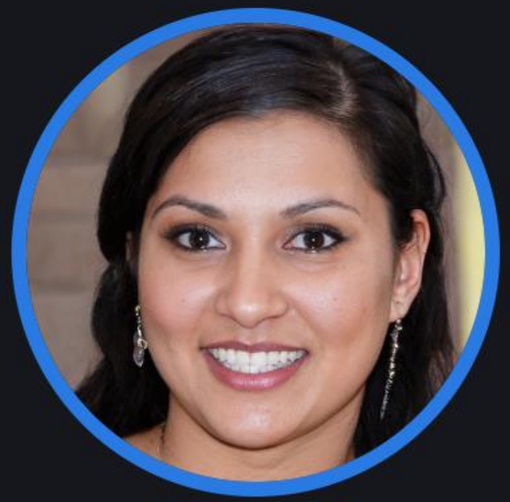
1. Projektordner mit index.html, Unterordner “images”
2. Anonymes Bild auch unter https://thispersondoesnotexist.com/
3. Lade alle Bilder herunter und lege sie in den Unterordner images
4. Lege ein CSS Datei an general_settings und schreibe dort alle Grundeinstellungen hinein
5. Lege für jeden Abschnitt deiner Seite (Header, section 1, section 2, footer) eine eigene CSS Datei an
6. Benutze die Schriftart Poppins (ohne Italic) (Import Methode in CSS Datei) mit https://fonts.google.com/specimen/Poppins?query=poppins
7. Lege die Farbcodes als CSS Variablen ab als main-brand-color, primary-color, secondary-color, light-secondary-color, text-color, secondary-text-color.
8. Effekt bei Link in Navigation mit Pseudo Element ::before :: after
9. Hamburger Menu unter https://fonts.google.com/icons?icon.style=Rounded
10. Benutze die Bildquellen wie unten angegeben (nächste Seiten)

Übung: Responsive Portfolio

Über mich Dienstleistungen Mehr Informationen

Mein Name, Webdeveloper

Lorem ipsum dolor sit amet, consectetur adipisicing elit. Quis, facilis. Repudiandae vero provident mollitia ea doloribus porro deserunt minus quod hic officiis accusantium, harum sint iusto molestias magni impedit eius, delectus facilis in maiores veniam ipsum blanditiis sapiente ullam. Fugit?



Über mich

Lorem ipsum dolor sit amet consectetur adipisicing elit. Tempore aperiam quaerat aliquid incidunt saepe, sed sequi aliquam accusamus laudantium sit consequuntur quam numquam atque velit eligendi tenetur facilis. Blanditiis dolore quidem quam voluptates provident accusantium amet cum dicta repellendus eius tenetur deserunt ea laboriosam fugit esse, fuga sequi, illum soluta! Natus fugiat, itaque optio sit laborum quibusdam repellat magni debitis consequatur totam aliquam placeat. Odio accusantium numquam exercitationem beatae velit ex autem qui pariatur. Aspernatur quaerat in maxime eveniet deleniti, id repellendus fugit delectus. Doloribus fuga nam voluptatem perferendis ex, numquam rem doloremque accusantium corporis. In doloremque velit aut officia laboriosam veniam odit facere deserunt, alias reprehenderit, molestiae dignissimos? Repellendus doloremque, reiciendis libero perspiciatis officia consectetur dignissimos quod necessitatibus aspernatur aliquam, doloribus enim ut pariatur quasi qui? Veniam, sunt quas.

Dienstleistungen

Beratung

Lorem ipsum dolor sit amet consectetur adipisicing elit. Voluptatem eveniet, perspiciatis dicta qui sint adipisci eaque aliquid expedita necessitatibus laboriosam minus ratione rerum aut officiis suscipit est labore minima. Quo?

Buchen

Beratung

Lorem ipsum dolor sit amet consectetur adipisicing elit. Voluptatem eveniet, perspiciatis dicta qui sint adipisci eaque aliquid expedita necessitatibus laboriosam minus ratione rerum aut officiis suscipit est labore minima. Quo?

Buchen

Beratung

Lorem ipsum dolor sit amet consectetur adipisicing elit. Voluptatem eveniet, perspiciatis dicta qui sint adipisci eaque aliquid expedita necessitatibus laboriosam minus ratione rerum aut officiis suscipit est labore minima. Quo?

Buchen

Mehr Informationen

Instagram

Lorem ipsum dolor sit, amet consectetur adipisicing elit. Dolorum iure rem totam magni ab cumque!

Youtube

Lorem ipsum dolor sit, amet consectetur adipisicing elit. Dolorum iure rem totam magni ab cumque!

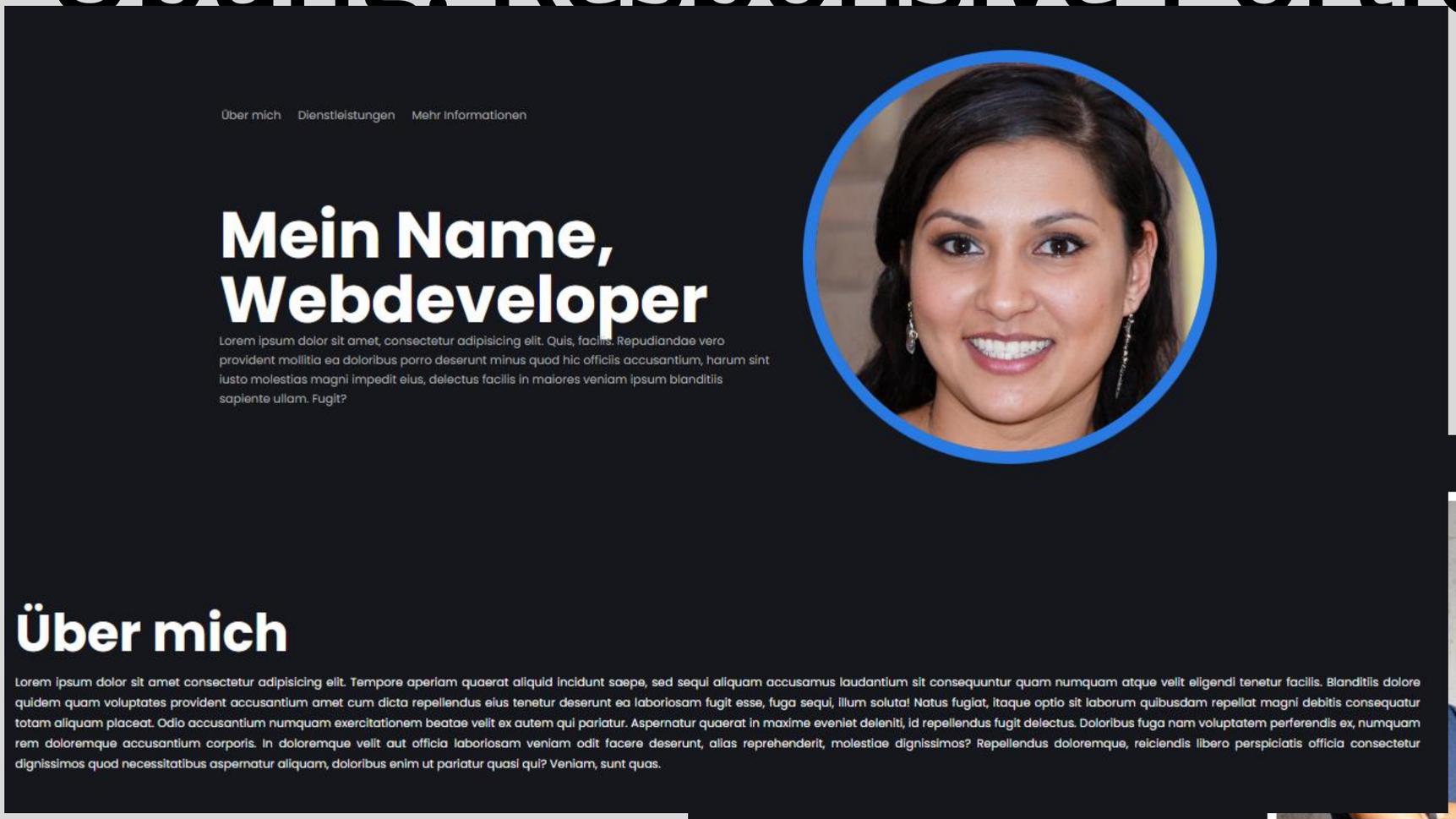
Telegram

Twitter

Kontakt

Lorem ipsum dolor, sit amet consectetur adipisicing elit. Delectus aspernatur necessitatibus fuga ex, corrupti dicta sint non rem quibusdam. Dolor, qui dolorem! Magni harum officiis doloribus veniam suscipit dignissimos exercitationem!

Übung: Responsive Portfolio



Übung: Responsive Portfolio

Übung: Responsive Portfolio

Bildquellen:

1. https://www.pexels.com/de-de/foto/mann-der-blaues-t-shirt-mit-rundhalsausschnitt-tragt-2379005/
2. https://thispersondoesnotexist.com/
3. https://www.pexels.com/de-de/foto/portratfoto-des-lachelnden-mannes-mit-gekreuzten-armen-die-vor-der-weissen-wand-stehen-2379004/
4. https://www.svgrepo.com/svg/271094/instagram
5. https://www.svgrepo.com/svg/271091/telegram
6. https://www.svgrepo.com/svg/126780/twitter
7. https://www.svgrepo.com/svg/134513/youtube

Farbpalette

- #297AE3;
- #15171C;
- #232933;
- #2E3744;
- white;
- #B3B4B6;

https://www.youtube.com/watch?v=qFdJsYiCcPI

CSS Flexbox

Debugging with Chrome

Flexbox: Debugging

https://firefox-source-docs.mozilla.org/devtools-user/page_inspector/how_to/examine_flexbox_layouts/index.html

https://developer.chrome.com/docs/devtools/css/flexbox/

CSS Emmet Abkürzungen

Video Quelle:

Lerne Emmet Abbreviations in 8 Minuten | 3x schneller HTML Code schreiben | Tutorial Deutsch

https://www.youtube.com/watch?v=4bO7gC9VCsA

CSS Variabeln

Video Quelle:

CSS Variables Tutorial (English)

https://www.youtube.com/watch?v=oZPR_78wCnY

https://codepen.io/WebDevSimplified/pen/wNyobw
https://www.w3schools.com/css/css3_variables.asp

CSS Pseudo Elemente

::before ::after

Video Quelle:

Before und After Pseudo Elemente | CSS Tutorial Deutsch

https://www.youtube.com/watch?v=-bXE9_dJKwA

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

https://www.w3schools.com/cssref/sel_before.php

https://www.w3schools.com/cssref/sel_after.php

CSS Button animieren

Video Quelle:

CSS Button Slide Hover Transition Effekt | CSS Tutorial Deutsch

https://www.youtube.com/watch?v=KpqWBgwiEDM

CSS Transitions

Video Quelle:

CSS Transitions erklärt | CSS Tutorial Deutsch

https://www.youtube.com/watch?v=zZ-oZNMGpzA

CSS Animations

Video Quelle:

CSS Animations Tutorial Deutsch

https://www.youtube.com/watch?v=s_f1aJ3xNic

CSS

Grid

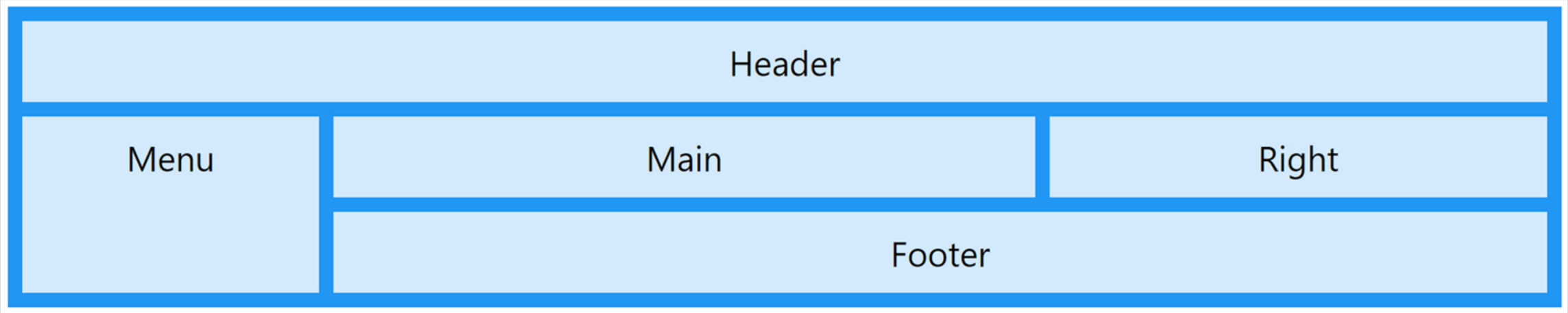
Video Quelle:

CSS Grid Layouts in 7 Minuten erklärt | CSS Tutorial Deutsch

https://youtu.be/UuHvG-1Np0w

Grid

Das CSS-Grid-Layoutmodul bietet ein rasterbasiertes Layoutsystem mit Zeilen und Spalten, das die Gestaltung von Webseiten erleichtert, ohne dass Floats und Positionierung verwendet werden müssen.



Grid

Ein **Grid-Layout** besteht aus einem übergeordneten Element und einem oder mehreren untergeordneten Elementen.

```html
<div class="grid-container">
  <div class="grid-item">1</div>
  <div class="grid-item">2</div>
  <div class="grid-item">3</div>
  <div class="grid-item">4</div>
  <div class="grid-item">5</div>
  <div class="grid-item">6</div>
  <div class="grid-item">7</div>
  <div class="grid-item">8</div>
  <div class="grid-item">9</div>
</div>
```

1	2	3
4	5	6
7	8	9

Grid: properties

Ein HTML-Element wird zu einem Grid-Container, wenn seine Display-Eigenschaft auf **Grid** oder **Inline-Grid** gesetzt ist.

Grid-Container bestehen aus Grid-Elementen, die in **Spalten** und **Zeilen** angeordnet sind.

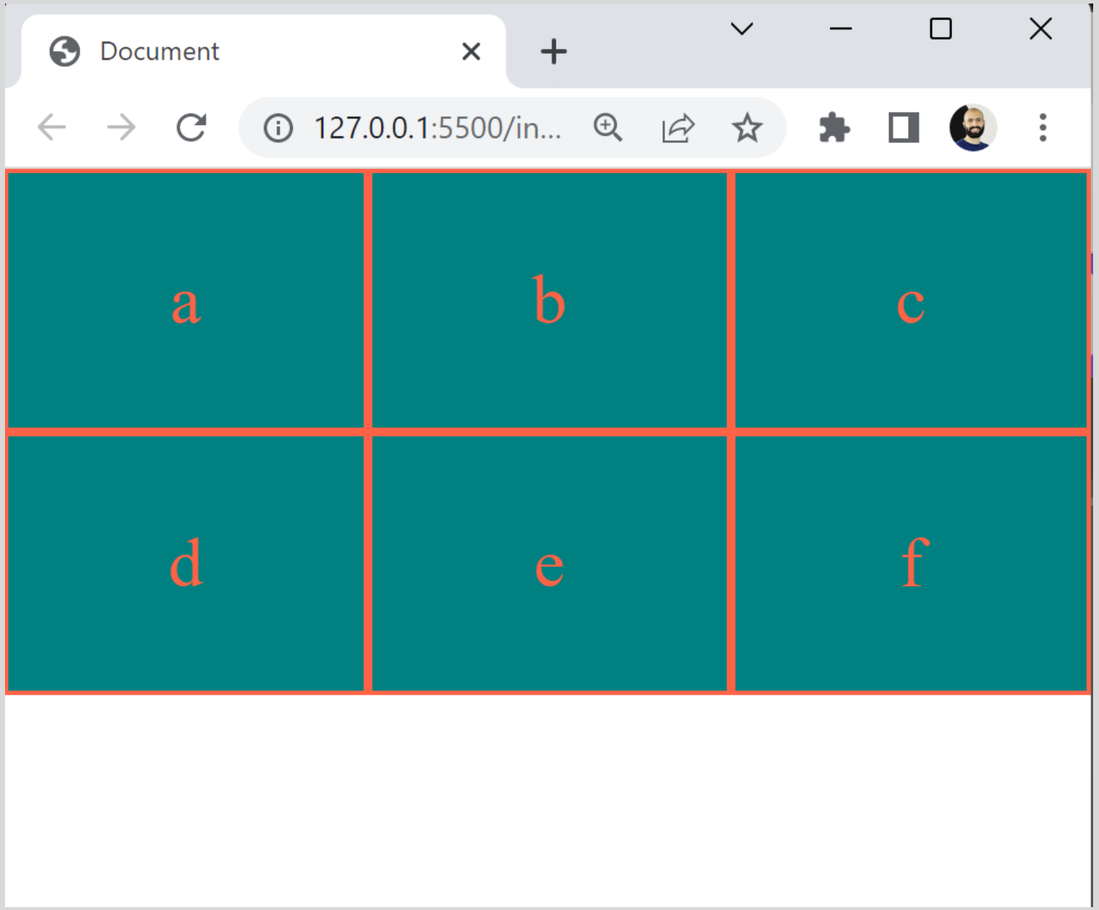
```
.grid-container {
    display: grid;
}
```

Alle direkten **untergeordneten element** des **Grid-Container** werden automatisch zu **Grid-Item.**

grid-template-columns

You can use the grid-template-columns property to specify the number of columns in your grid layout.

```
.grid-container {
  display: grid;
  grid-template-columns: auto auto auto;
}
```

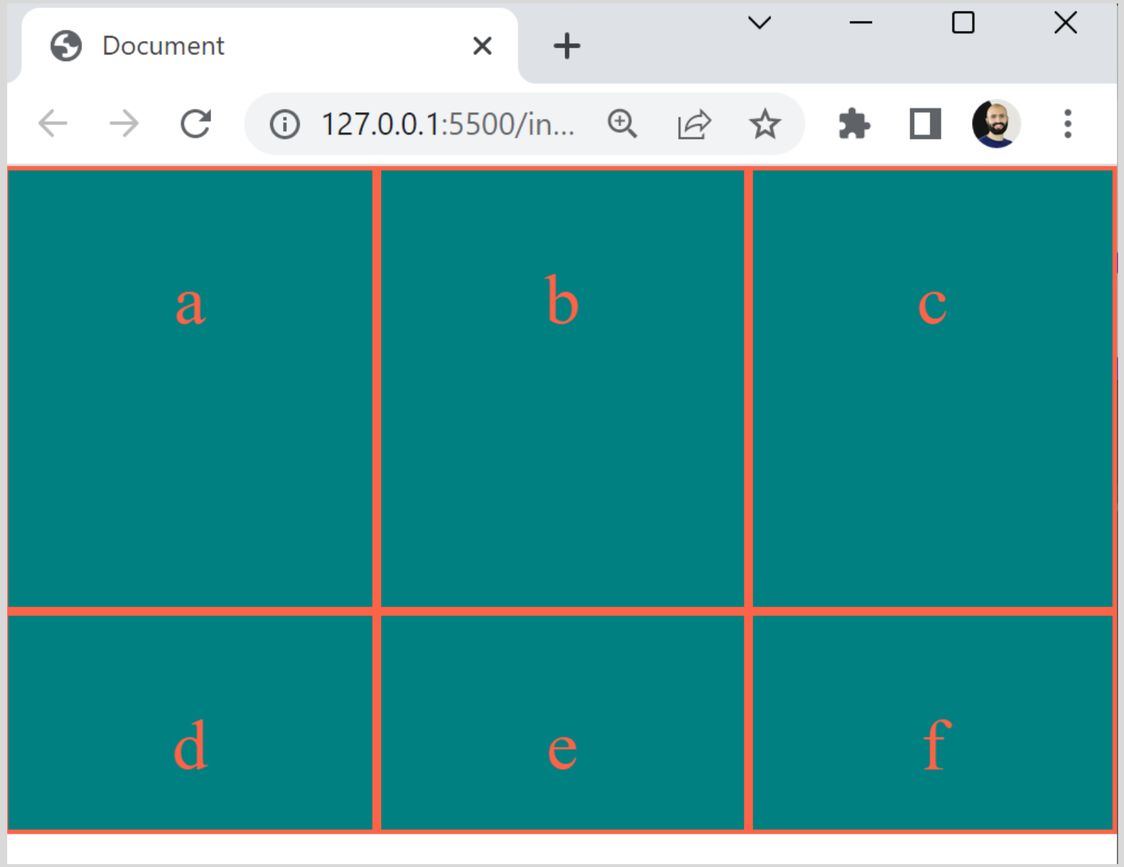


If you want your grid layout to contain 4 columns, specify the width of the 4 columns, or "auto" if all columns should have the same width.

grid-template-rows

Sie können die Eigenschaft grid-template-rows verwenden, um die Anzahl der Zeilen in Ihrem Grid-Layout anzugeben.

```
.grid-container {
  display: grid;
  grid-template-columns: auto auto auto;
  grid-template-rows: 100px 50px;
}
```



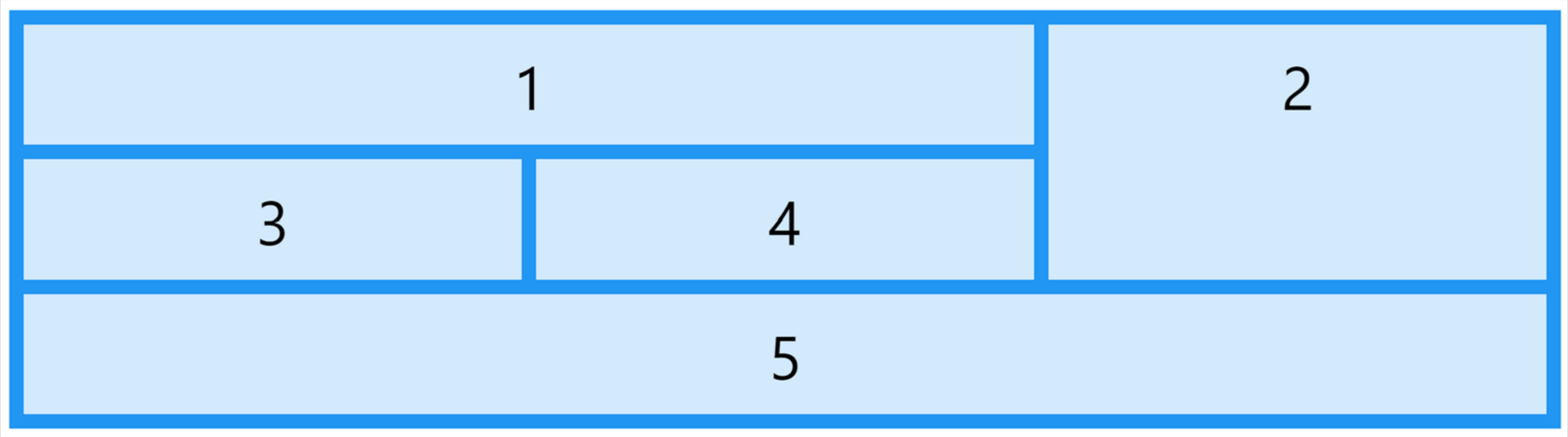
Grid-parent: properties

justify-content wird verwendet, um das gesamte Raster innerhalb des Containers horizontale auszurichten.

align-content wird verwendet, um das gesamte Raster innerhalb des Containers vertikal auszurichten.

Grid-item/child

Standardmäßig hat ein Container ein Grid-Element für jede Spalte und jede Zeile, aber Sie können die Grid-Elemente so gestalten, dass sie sich über mehrere Spalten und/oder Zeilen erstrecken.

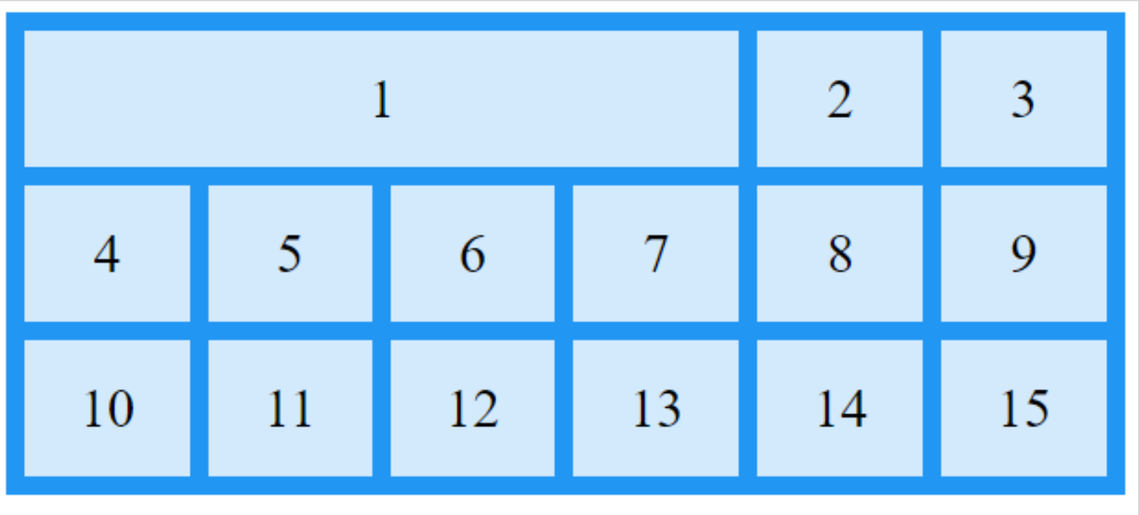


Grid-item/child: grid-column

The grid-column definiert, in welcher(n) Spalte(n) ein Element platziert werden soll.
Sie bestimmen, wo das Element beginnt und wo es enden soll.

The grid-column ist eine Abkürzung Eigenschaft für die Eigenschaften grid-column-start und grid-column-end.

```
.grid-container {
  display: grid;
  grid-template-columns: auto auto auto auto auto a
  gap: 10px;
  background-color: #2196F3;
  padding: 10px;
}
.item1 {
  grid-column: 1 / 5;
}
```

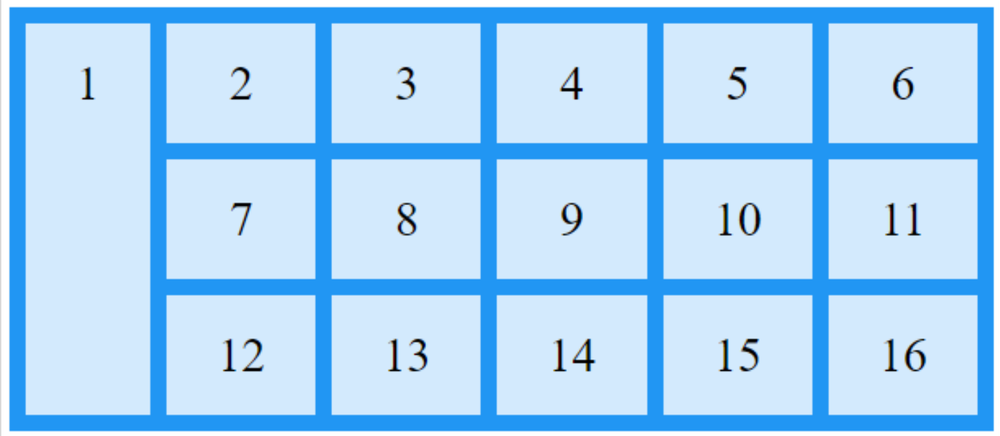


Grid-item/child: grid-row

grid-row definiert, in welcher Zeile ein Element platziert werden soll.
Sie legen fest, wo das Element beginnt und wo es enden soll.

grid-row eigenschaft ist eine Kurzform für die Eigenschaften grid-row-start und grid-row-end.

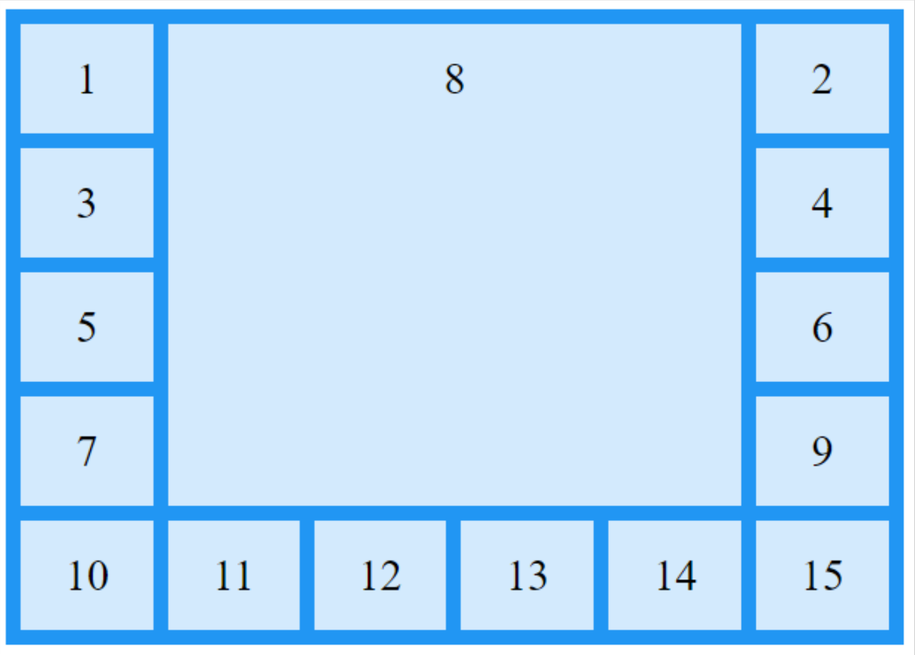
```
.grid-container {
  display: grid;
  grid-template-columns: auto auto auto auto auto auto;
  gap: 10px;
  background-color: #2196F3;
  padding: 10px;
}
.item1 {
  grid-row: 1 / 4;
}
```



Grid-item/child: grid-area

grid-area kann als Shorthand-Eigenschaft für die Eigenschaften grid-row-start, grid-column-start, grid-row-end und grid-column-end verwendet werden.

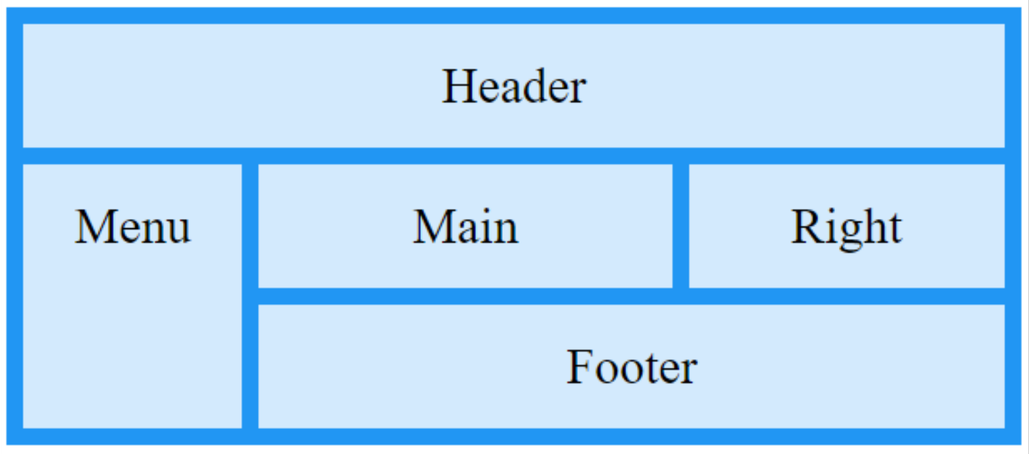
```
.grid-container {
  display: grid;
  grid-template-columns: auto auto auto auto auto auto;
  gap: 10px;
  background-color: #2196F3;
  padding: 10px;
}
.item8 {
  grid-area: 1 / 2 / 5 / 6;
}
```



Grid: grid-area

grid-area Eigenschaft kann auch verwendet werden, um grid-items einen Namen zu geben. Auf benannte Grid-Items kann über die Eigenschaft grid-template-areas des Grid-Containers verwiesen werden.

```
.item1 { grid-area: header; }
.item2 { grid-area: menu; }
.item3 { grid-area: main; }
.item4 { grid-area: right; }
.item5 { grid-area: footer; }
.grid-container {
  display: grid;
  grid-template-areas:
    'header header header header header header'
    'menu main main main right right'
    'menu footer footer footer footer footer';
  gap: 10px;
  background-color: #2196F3;
  padding: 10px;
}
```



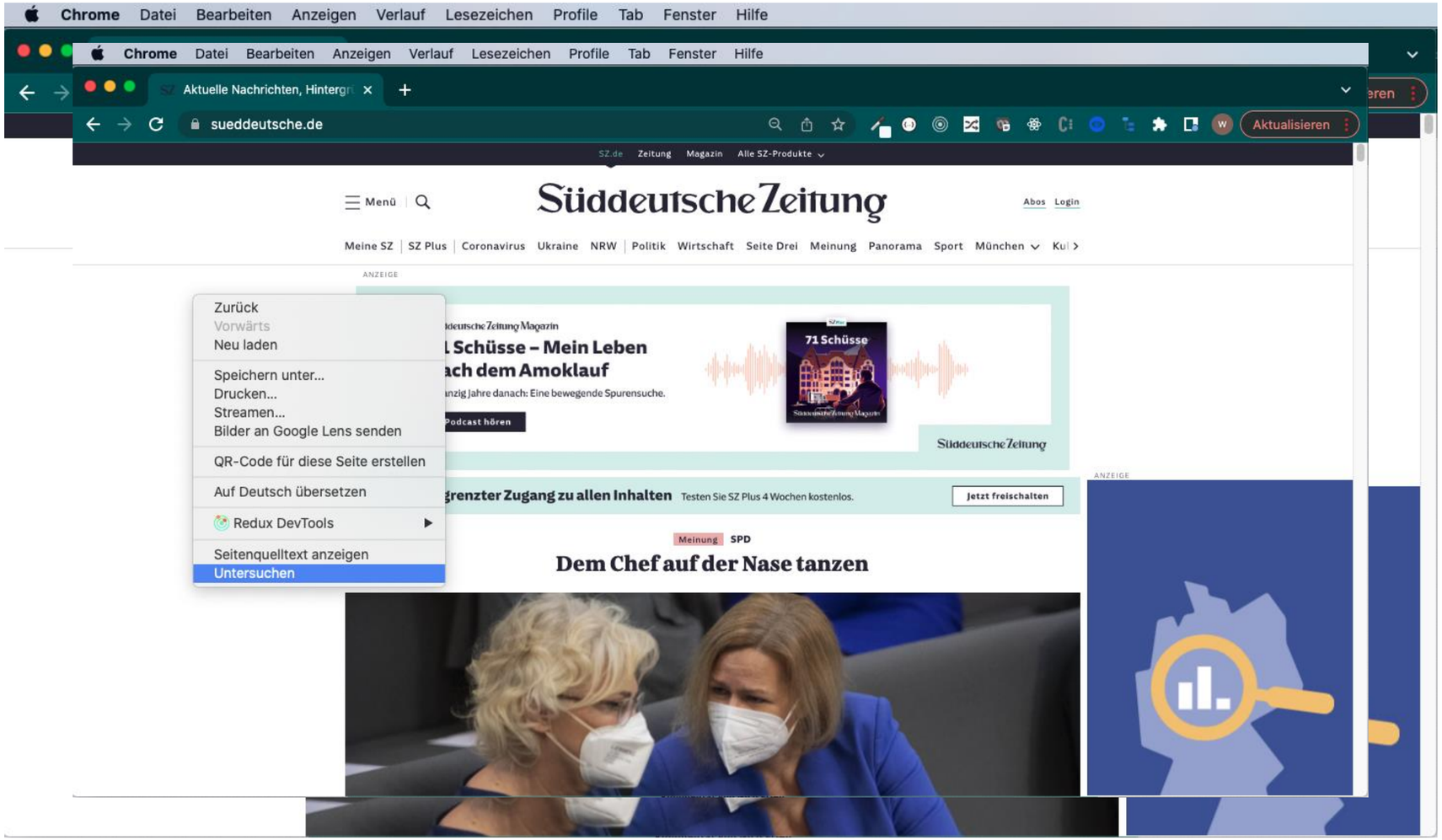
https://cssgridgarden.com

CSS framework

Debugging Dev tools

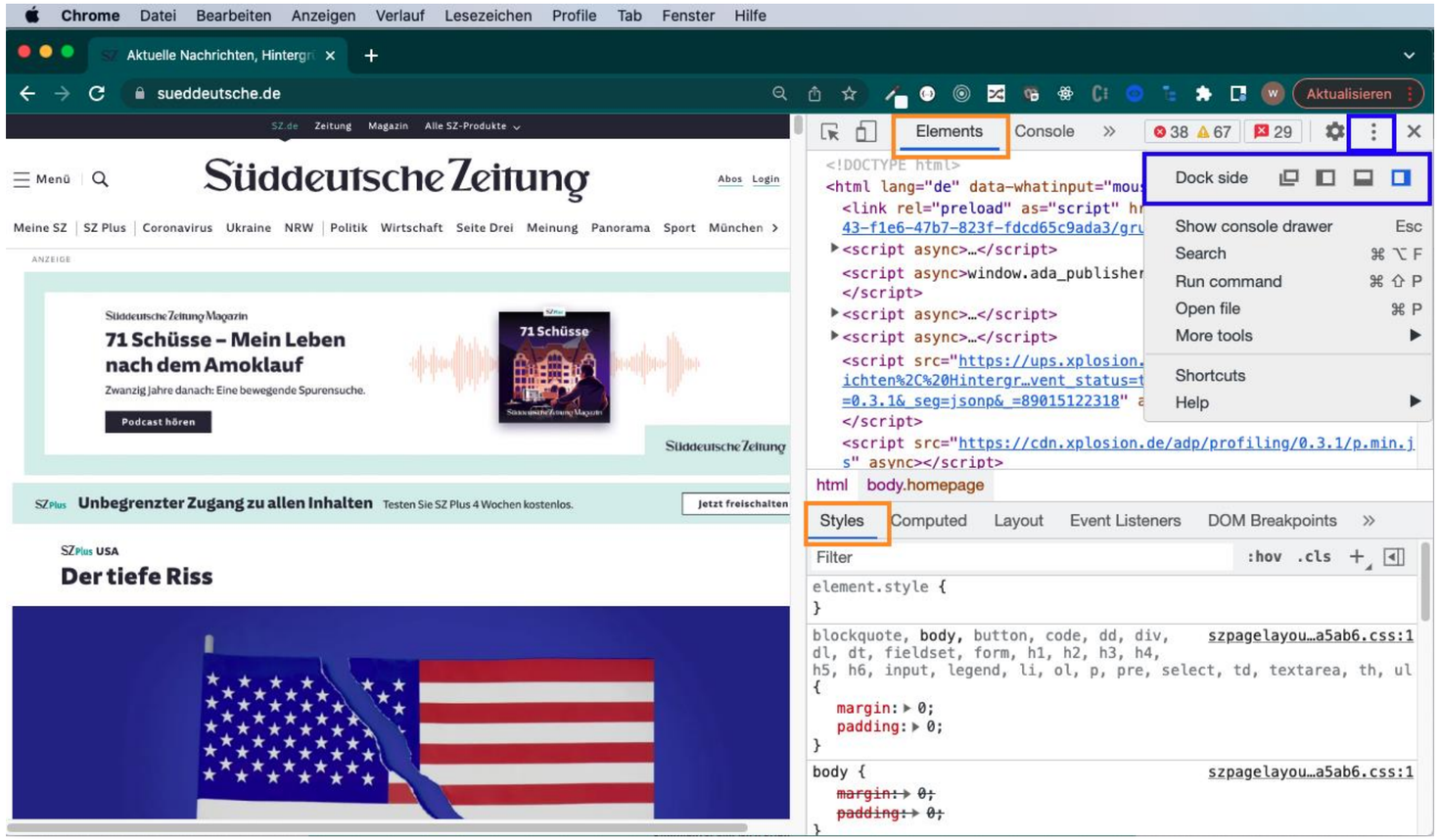
Chrome devtools: inspect mode öffnen

Um die Seite zu untersuchen, kann man mit “***Rechtslick => Untersuchen***” die DevTools öffnen.



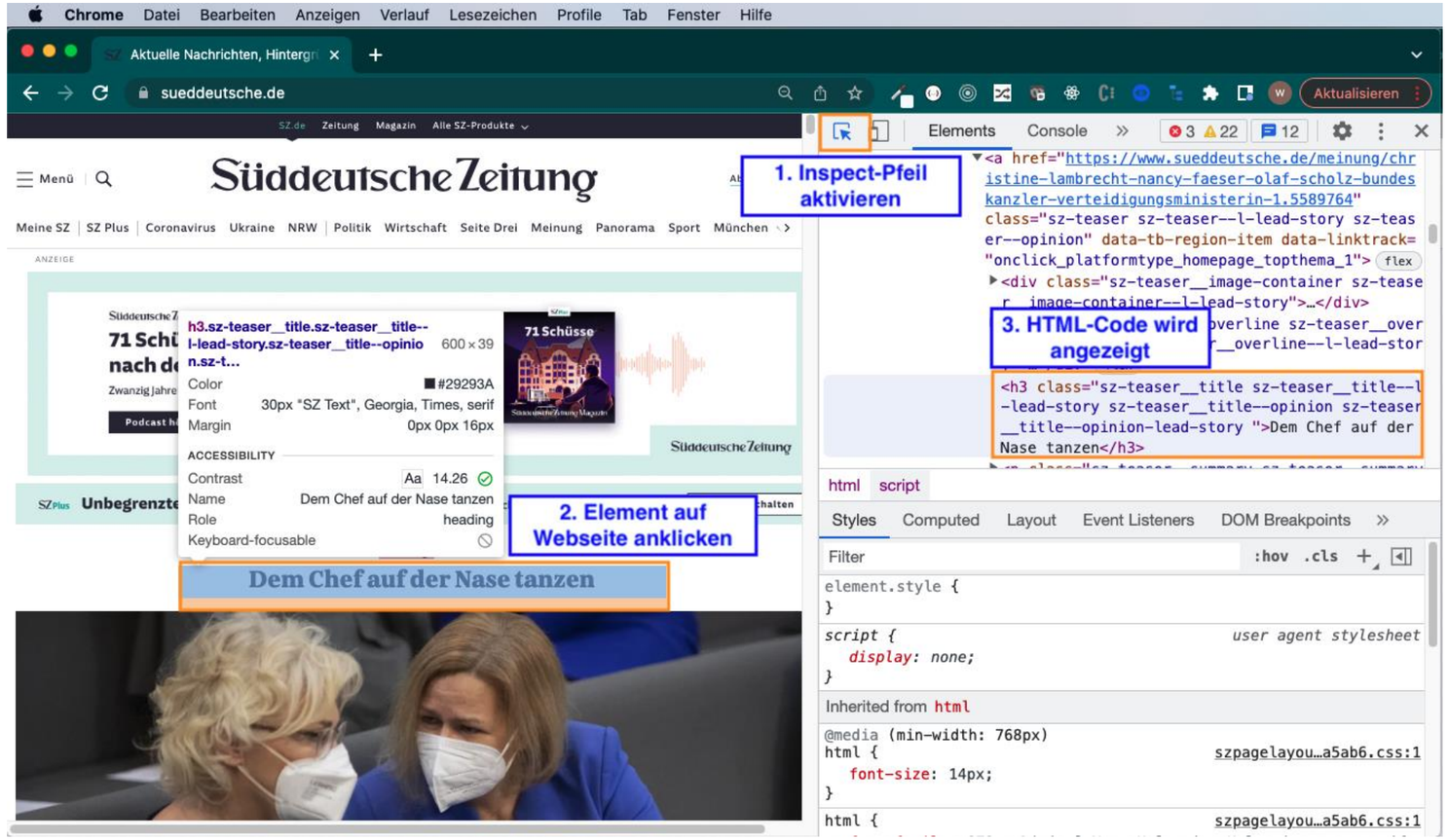
Chrome devtools: Struktur

Achte darauf, dass der obere Tab “***Elements***” und der untere Tab “***Styles***” ausgewählt sind.
Über das ***Drei-Punkte-Menü*** kann man auswählen, wie sich das DevTools Fenster öffnet (rechts, links, unten, neues Fenster).



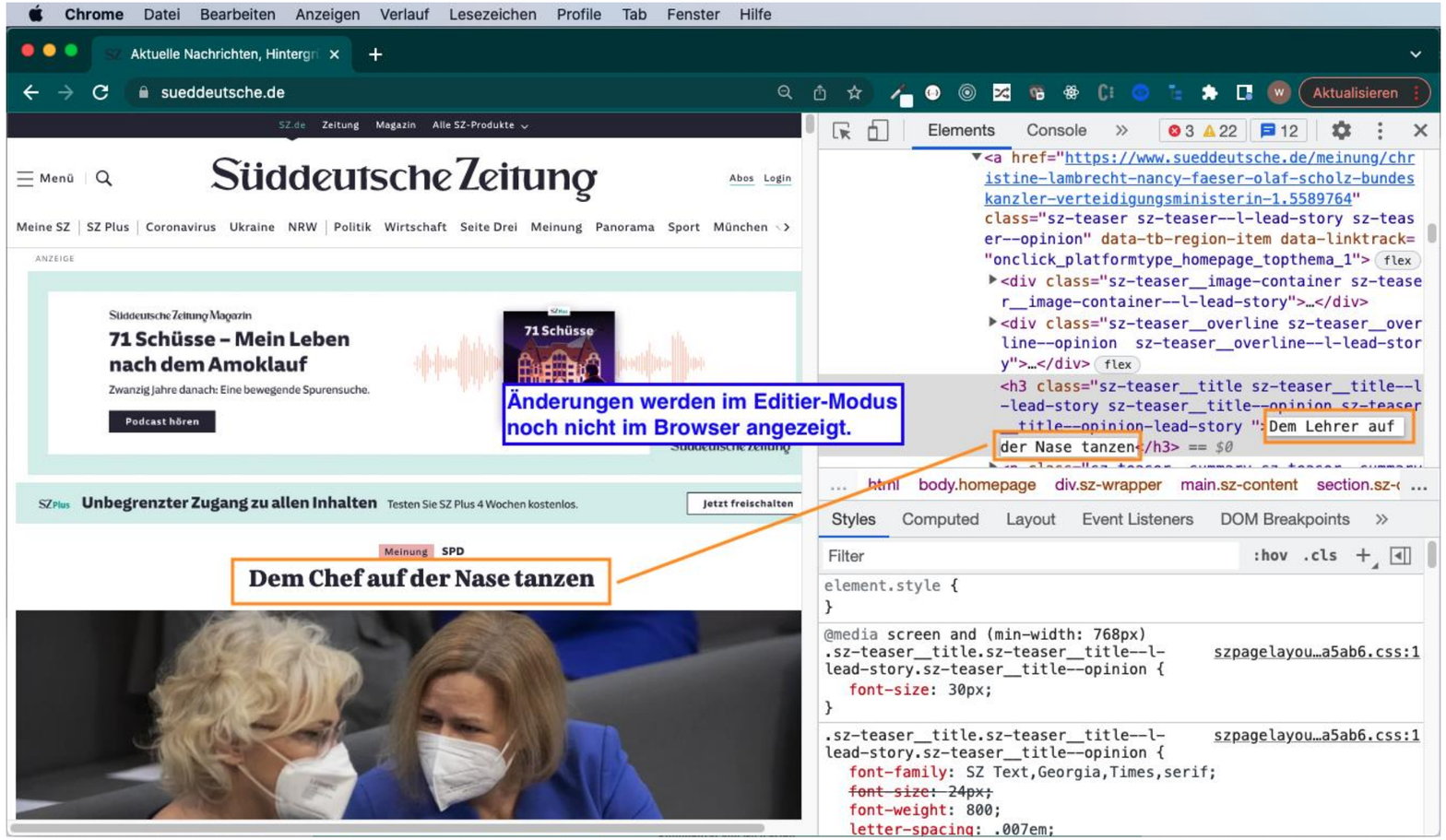
Chrome devtools: HTML-Element auswählen

Aktiviere den ***Inspect-Pfeil***, um HTML-Elemente auf der Webseite zu untersuchen. Sobald du auf ein Element auf der Webseite klickst, wird der dazugehörige Code im Elements-Tab angezeigt.



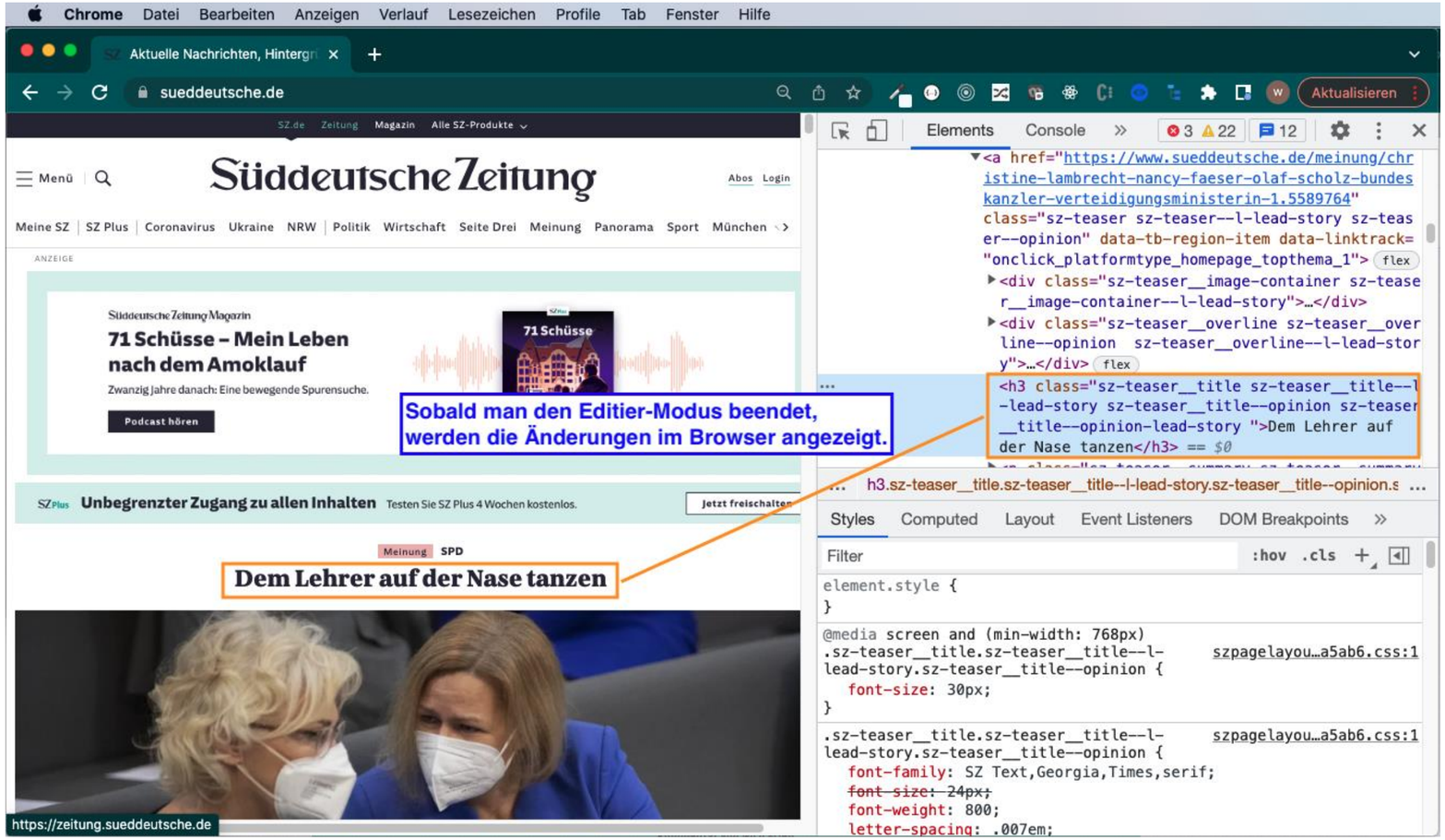
Chrome devtools: HTML-Element editieren

Um ein bestehendes HTML-Element zu editieren, kann man auf den gewünschten Text doppel-klicken, um ihn zu verändern.



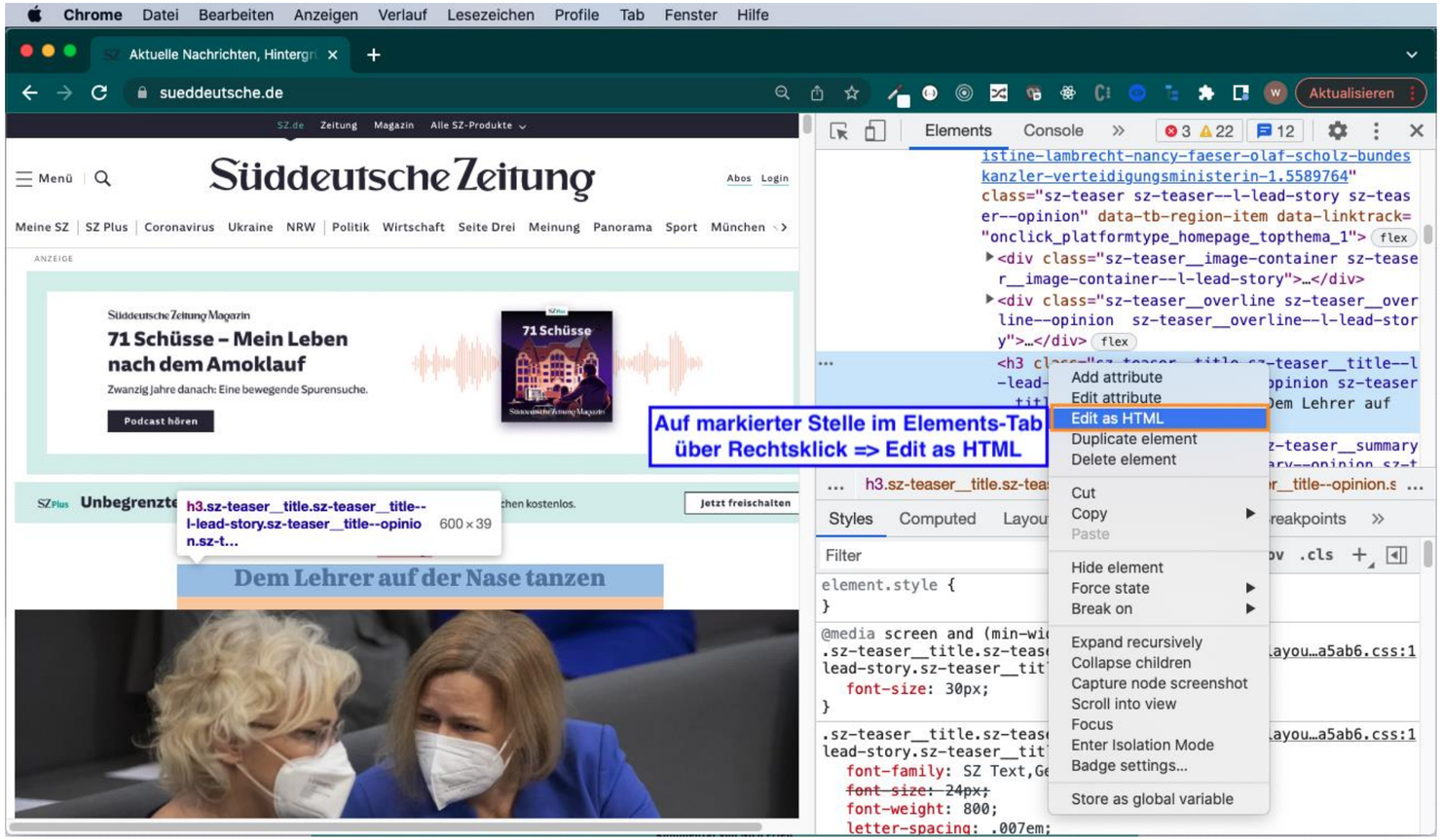
Chrome devtools: HTML-Element editieren

Nach der Änderung außerhalb des Textfeldes klicken. Somit wird die Änderung übernommen und im Browser angezeigt.



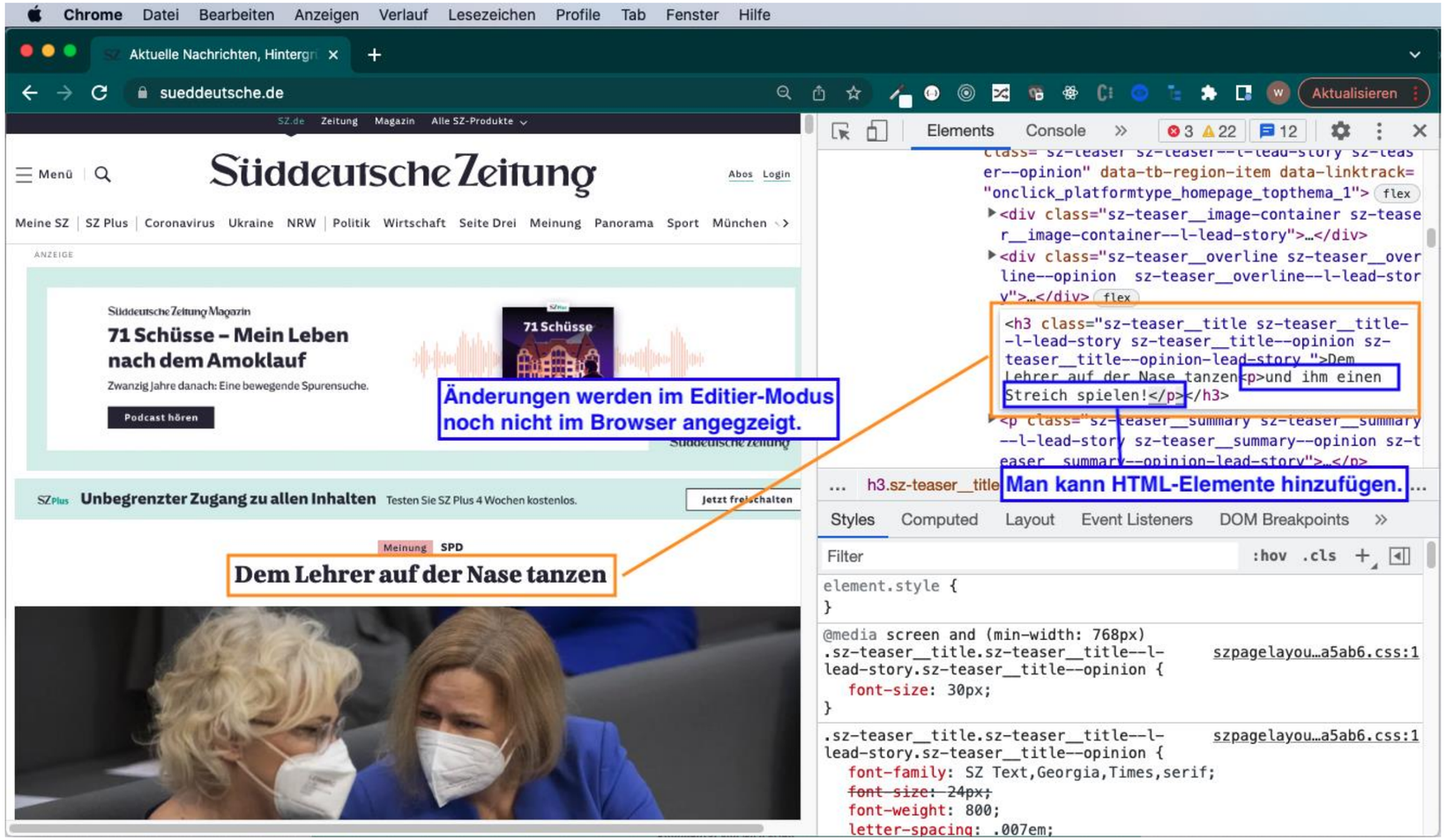
Chrome devtools: HTML-Element hinzufügen

Um ein HTML-Element hinzuzufügen, kann man die Funktion “***Edit as HTML***” nutzen.



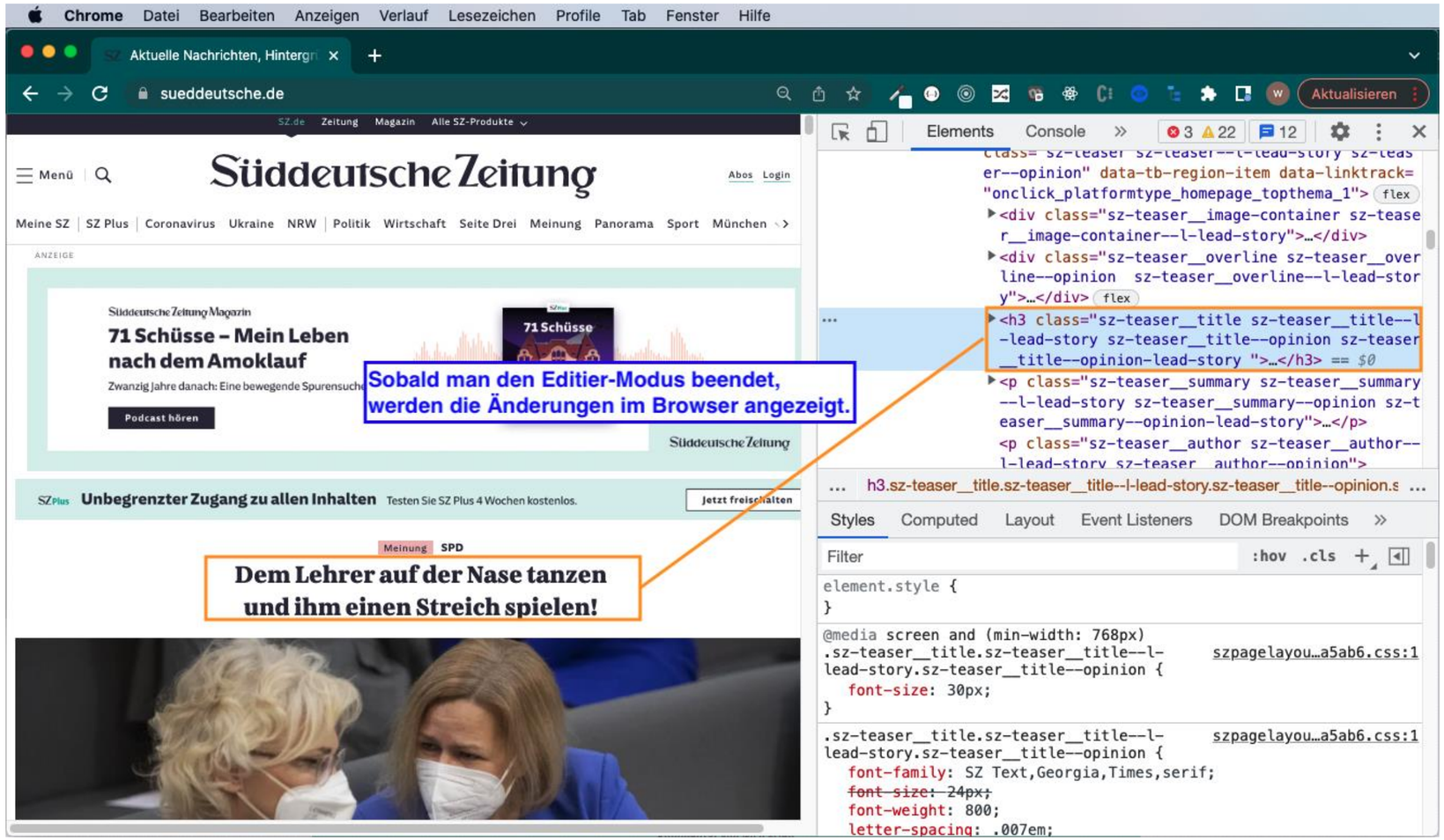
Chrome devtools: HTML-Element hinzufügen

Es können HTML-Elemente hinzugefügt werden unter Anwendung der öffnenden und schließenden tags.



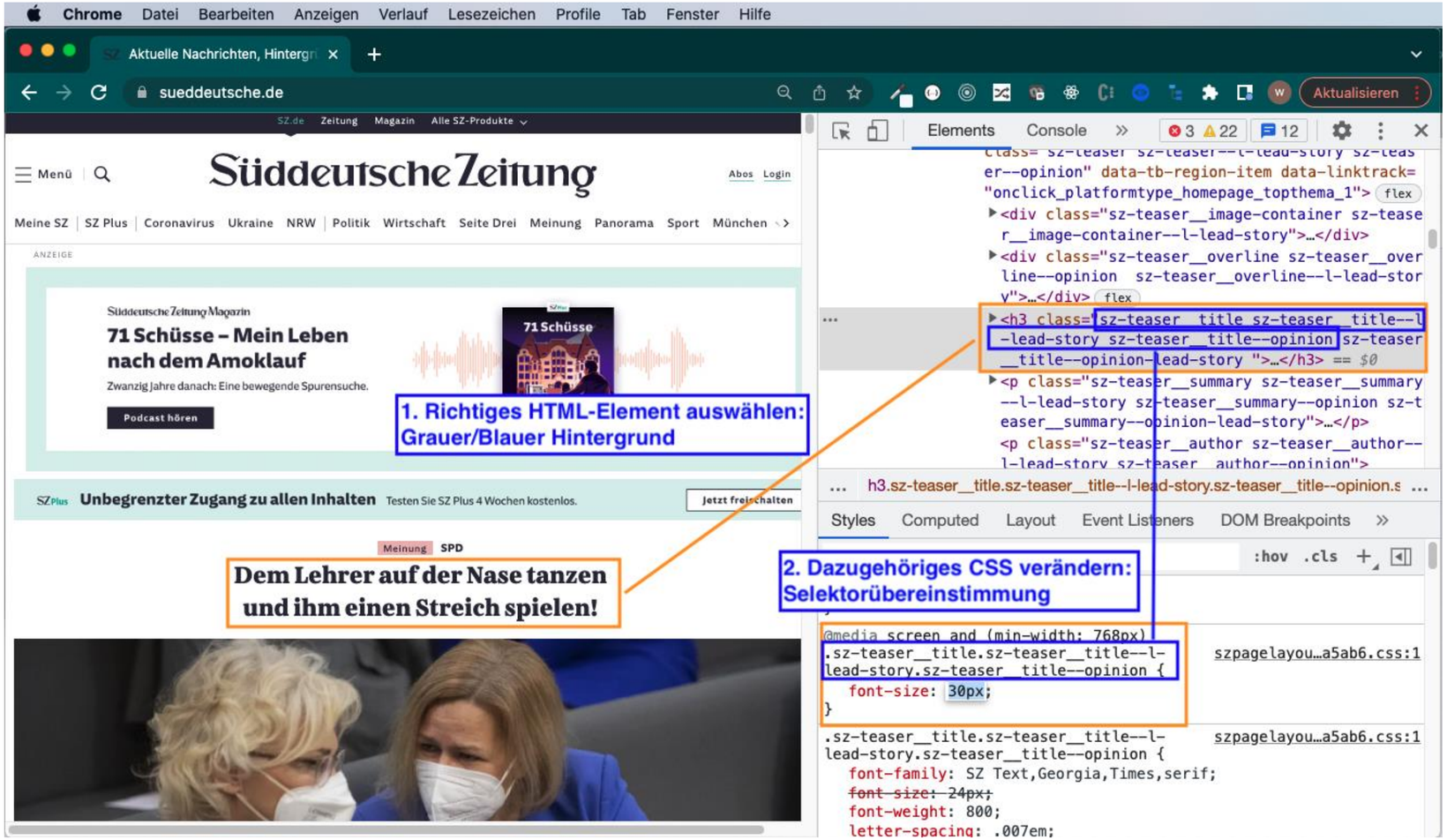
Chrome devtools: HTML-Element hinzufügen

Nach der Änderung außerhalb des Textfeldes klicken. Somit wird die Änderung übernommen und im Browser angezeigt.



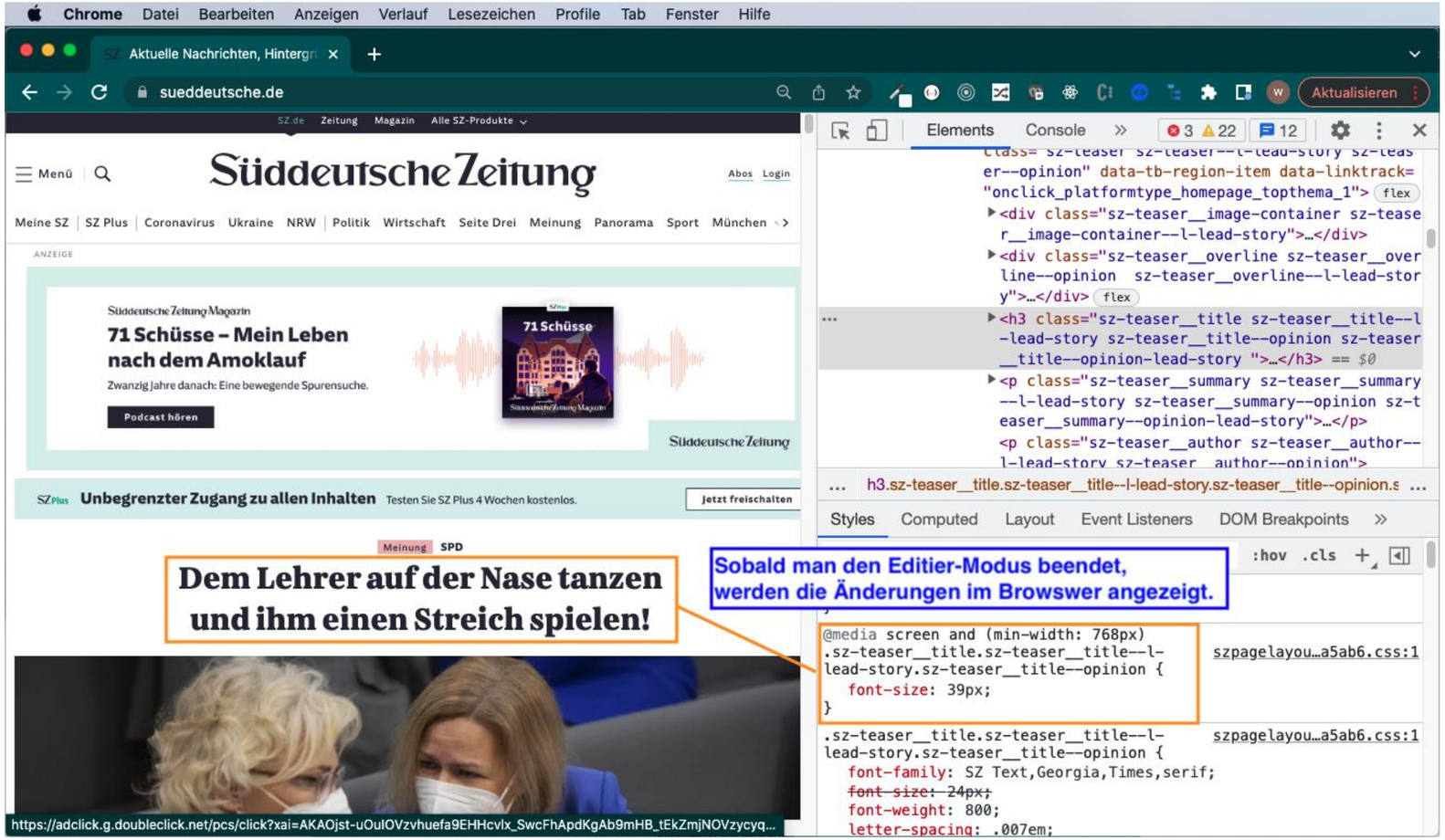
Chrome devtools: CSS editieren

Um bestehenden CSS-Code zu editieren, muss das gewünschte HTML-Element ausgewählt werden. Danach können beliebige CSS-Deklarationen verändert werden.



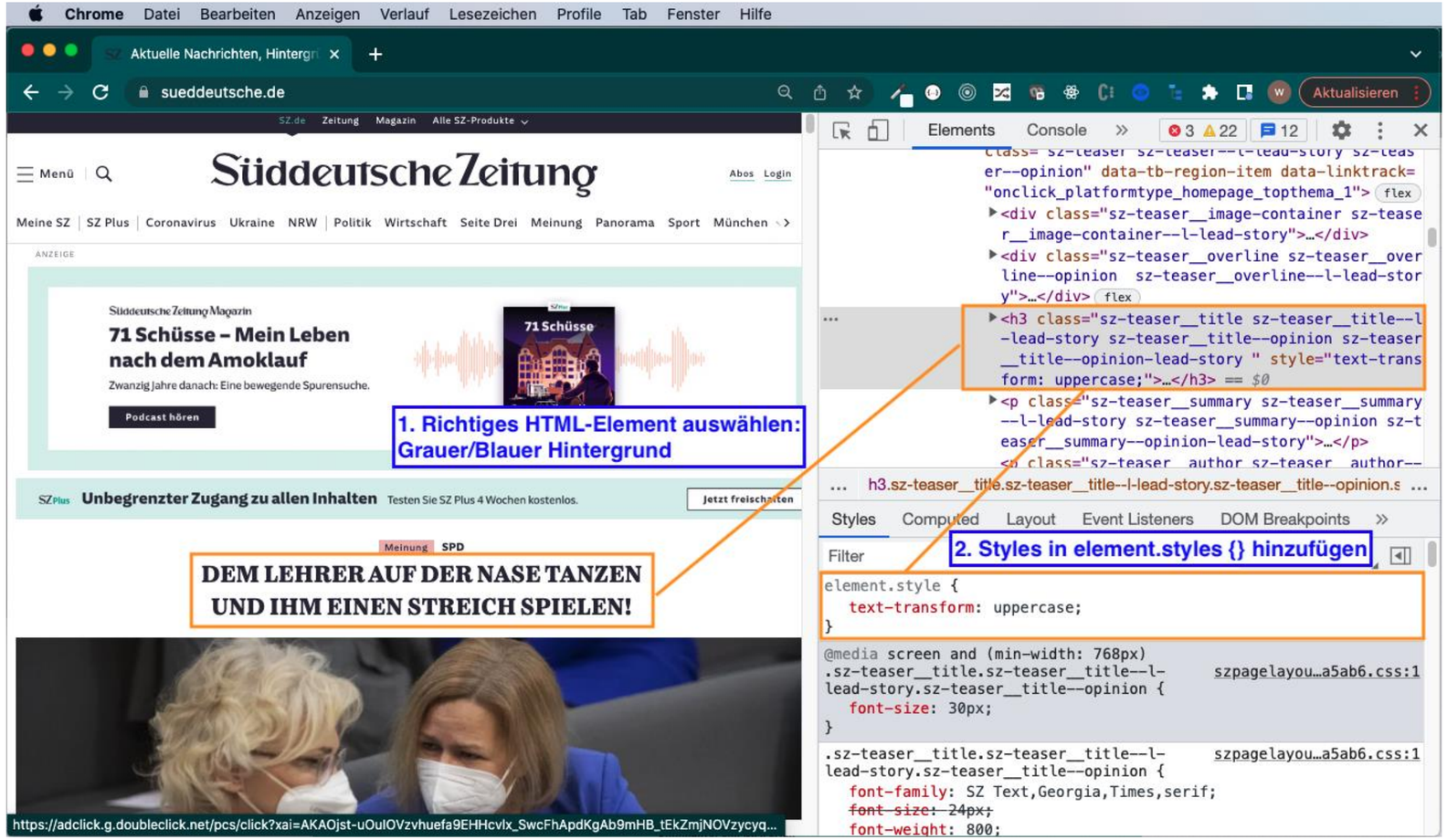
Chrome devtools: CSS editieren

Nach der Änderung außerhalb des Textfeldes klicken. Somit wird die Änderung übernommen und im Browser angezeigt.



Chrome devtools: CSS Hinzufügen

Damit nur das ausgewählte Element zusätzliche styles erhält, kann man den Selektor ***element.style {}*** verwenden.



Übung: Süddeutsche Zeitung

1. Die **Überschrift “Süddeutsche Zeitung”** soll die **Farbe #81d1c6** haben.
2. Die **Top-Navigation** soll die **Hintergrundfarbe #d3d3d3** haben.
1. Der **Body** soll die **Hintergrundfarbe #f8f8ff** haben.
2. Die **Klasse “sz-wrapper”** soll die **Hintergrundfarbe #81d1c6** haben.
3. Die **Überschrift des Hauptartikels** soll die **Schrifart monospace** haben.
4. Alle **Teaser-Überschriften** sollen die **Schriftart monospace** haben.