

# CSS – Flexbox & Grid



# CSS Flexbox vs other Layouts

- Ressource: [3.CSS/Codebeispiele/Layout/](#)

## Flexbox

Lorem ipsum dolor sit amet consectetur adipisicing elit. Temporibus eveniet minus dolor qui laudantium molestias repudiandae reprehenderit impedit aliquid, fugit eligendi placeat, unde praesentium, vel veritatis voluptate ex corporis magni deleniti. Ipsum id at ad quo neque eos ducimus similique aliquam, modi unde soluta inventore non eius tenetur quibusdam corrupti!

Lorem ipsum dolor sit amet consectetur adipisicing elit. Cumque nobis rem ipsa voluptatibus, recusandae illum consequuntur cum, nulla dolores eligendi fuga harum labore animi dolorum asperiores voluptate praesentium beatae, quibusdam sapiente illo ullam cupiditate officiis. Qui consequuntur sit quaerat atque magni possimus, nemo, pariatur incidunt delectus laborum veritatis placeat fugit.

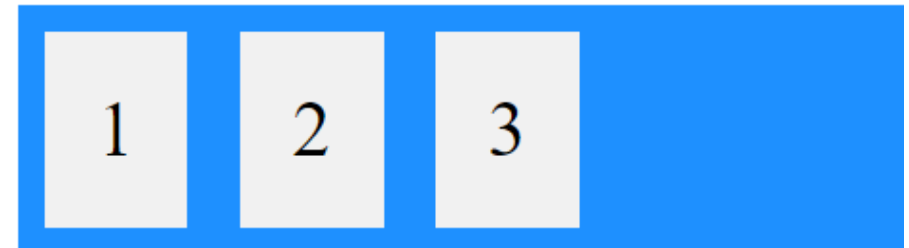
Lorem, ipsum dolor sit amet consectetur adipisicing elit. Possimus, voluptatem officia nulla minus, sunt nam aliquam quis explicabo consequatur beatae at? Eligendi corrupti quisquam enim accusantium possimus quis esse praesentium! Obcaecati repudiandae commodi, earum eaque velit voluptatum illum temporibus. Corporis quod dolorem officia, optio eos fuga. Voluptatum repellat maiores laborum rem voluptas esse. Hic consequuntur, quidem nobis suscipit iste autem quae! Itaque ut doloribus eaque quidem quam nostrum eos quibusdam, id vero optio consectetur velit ipsum architecto ad maiores modi aperiam nemo sed est molestiae? Facere velit magnam assumenda ullam veniam officia ea doloribus dolor. Atque, dolore exercitationem. Expedita, eius?

# CSS Flexbox

- Das Modul für das flexible Box-Layout erleichtert die Gestaltung flexibler, reaktionsfähiger Layoutstrukturen.
- Anwendungsbereich eher bei **1-dimensionale** Strukturen
- Ein flexibles Layout muss ein übergeordnetes Element haben, bei dem die Eigenschaft "display" auf "flex" gesetzt ist.
- Die direkten Child-Elemente des flexiblen Containers werden automatisch zu flexiblen Elementen.
- Über weitere Flex-Eigenschaften kann das Layout genau definiert werden
  - Flex-direction
  - Align-items
  - Justify-content
  - Align-content
  - ...

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

```
.flex-container > div {
  background-color: #f1f1f1;
  margin: 10px;
  padding: 20px;
  font-size: 30px;
}
```

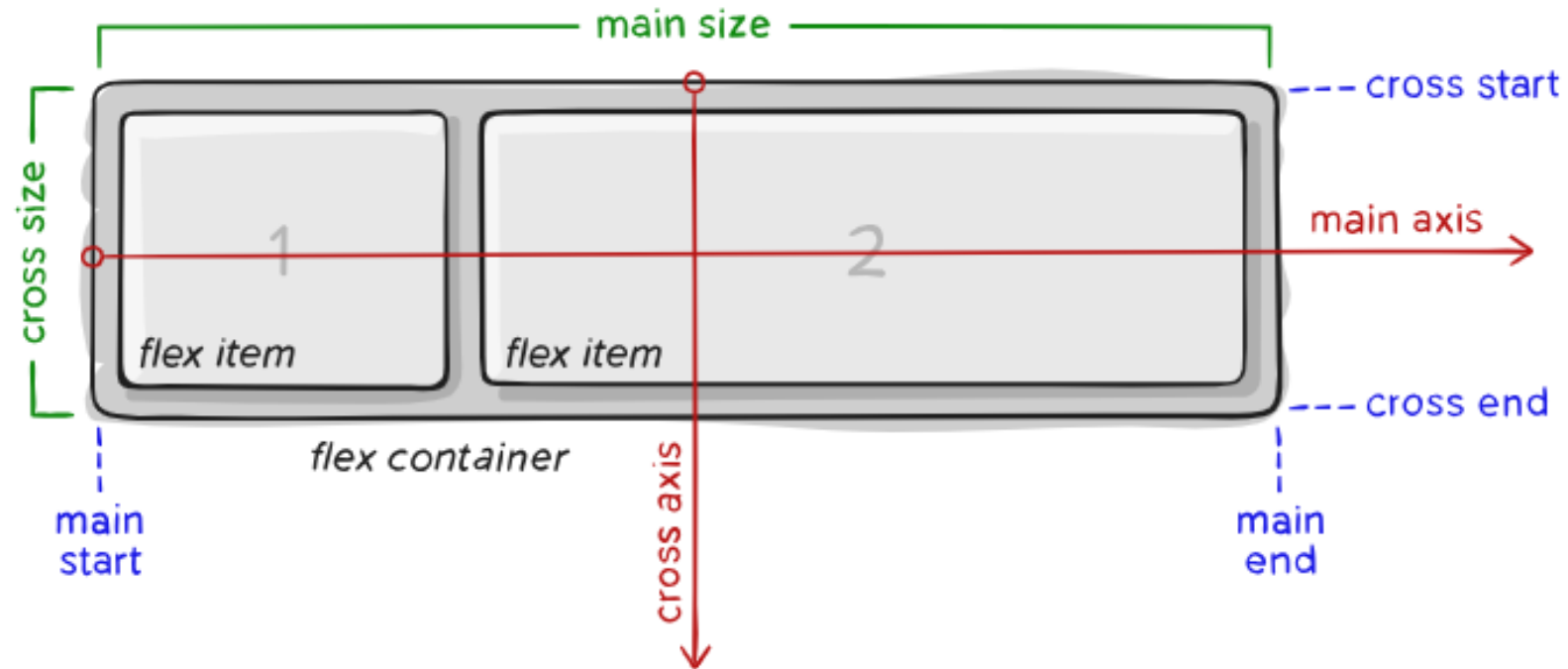


# CSS Flexbox Miniübung: Navigationbar

- Ressource: 3.CSS/Codebeispiele/Flexbox
- Code anpassen um eine Navigationbar zu erzielen

**Page Layout Methods** [HTML Table](#) [Inline-Block](#) [Absolute Positioning](#) [Float](#) [Flexbox](#)

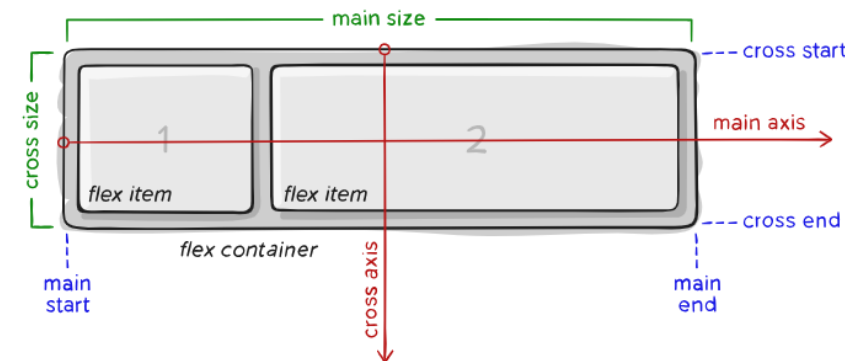
# CSS Flexbox: Layout



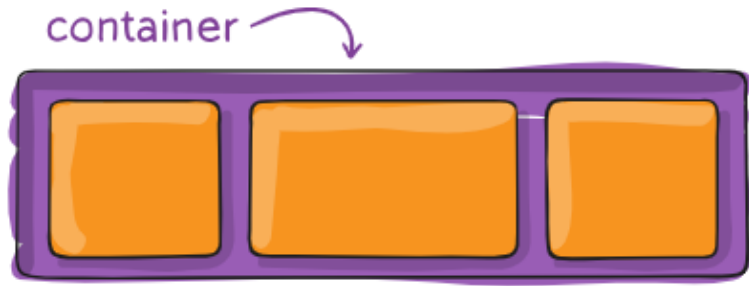
<https://css-tricks.com/snippets/css/a-guide-to-flexbox/>

# CSS Flexbox: Layout

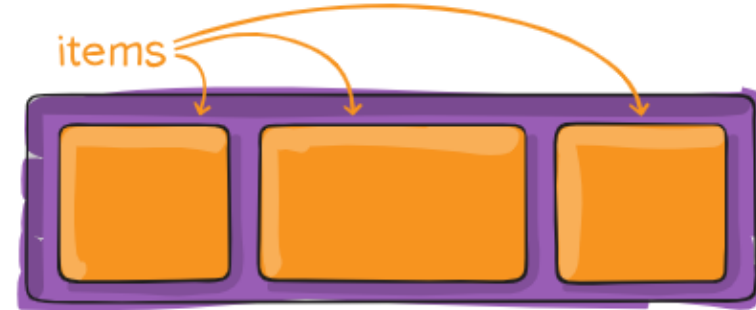
- **main-axis:** Die Hauptachse eines Flexcontainers ist die primäre Achse, entlang derer Flexelemente angeordnet werden (abhängig von flex-direction: row | column)
- **main-start | main-end:** Die Flexelemente werden im Container von main-start aus angeordnet und gehen bis zu main-end.
- **main-size:** Die Breite oder Höhe eines Flexelements, je nachdem, welche in der Hauptdimension liegt, ist die Hauptgröße des Elements. Die Hauptgröße eines Flexelements ist entweder die Eigenschaft 'width' (Breite) oder 'height' (Höhe), je nachdem, welche in der Hauptdimension liegt.
- **cross-axis:** Die Achse, die senkrecht zur Hauptachse verläuft, wird Querachse genannt. Ihre Richtung hängt von der Richtung der Hauptachse ab.
- **cross-start | cross-end:** Flexlinien werden mit Elementen gefüllt und im Container auf der cross-start-Seite beginnend und in Richtung der cross-end-Seite platziert.
- **cross-size:** Die Breite oder Höhe eines Flexelements, je nachdem, welche in der Querdimension liegt, ist die Quergröße des Elements. Die Quergröße ist entweder 'width' (Breite) oder 'height' (Höhe), je nachdem, welche in der Querdimension liegt.



# CSS Flexbox: Container & Items Properties



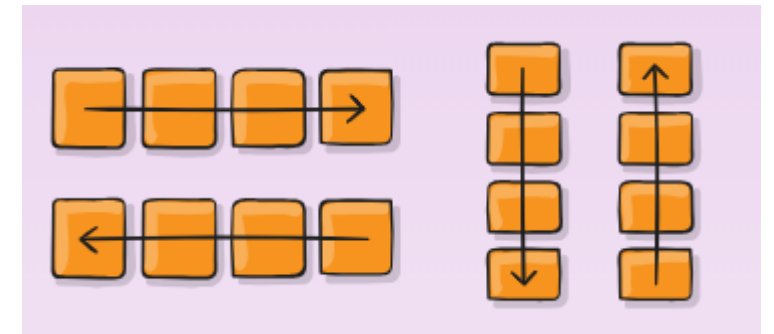
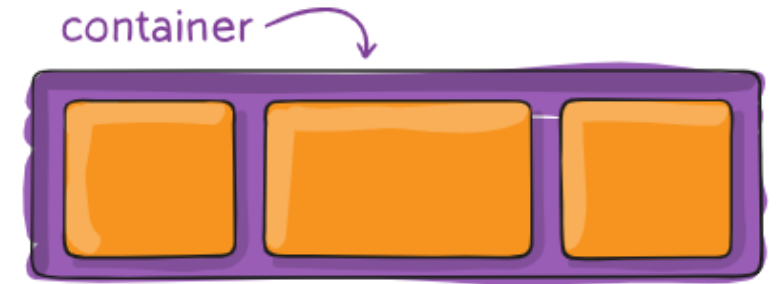
- flex-direction
- flex-wrap
- flex-flow (shorthand direction & wrap)
- justify-content
- align-items
- align-content
- gap, row-gap, column-gap



- order
- flex-grow
- flex-shrink
- flex-basis
- flex (shorthand grow&shrink&basis)
- align-self

# CSS Flexbox: flex-direction

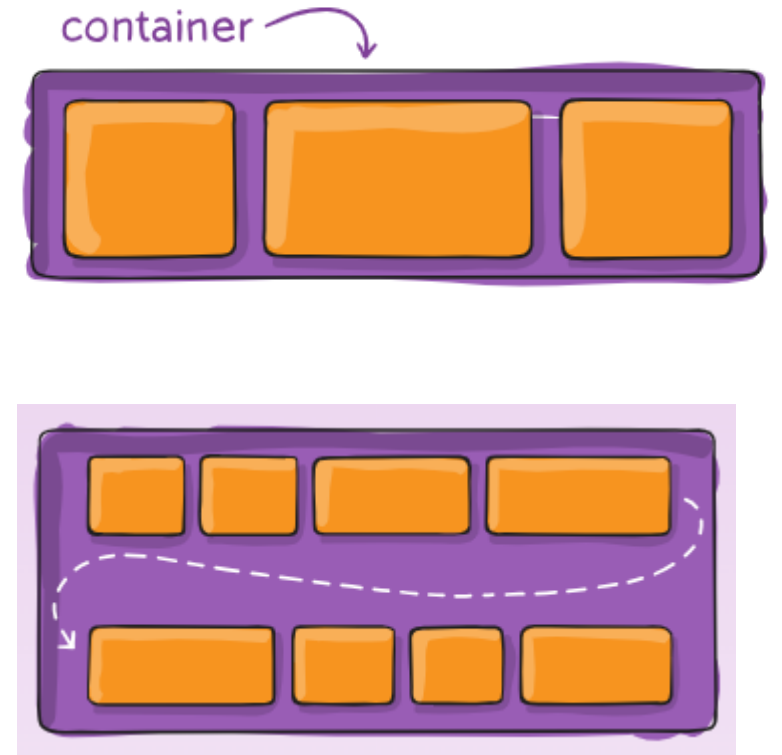
- Legt die Hauptachse fest und definiert somit die Richtung, in der Flexelemente im Flexcontainer platziert werden.
- row (Standard): Von links nach rechts in LTR (von rechts nach links in RTL).
- row-reverse: Von rechts nach links in LTR (von links nach rechts in RTL).
- column: Wie "row", aber von oben nach unten.
- column-reverse: Wie "row-reverse", aber von unten nach oben.





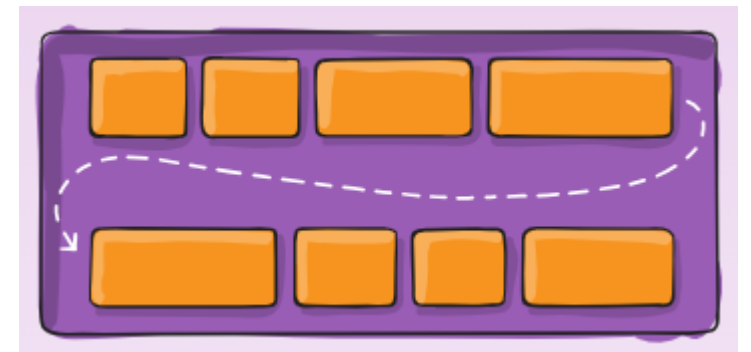
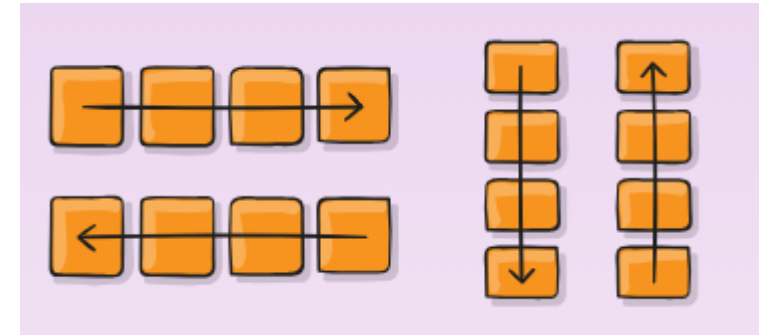
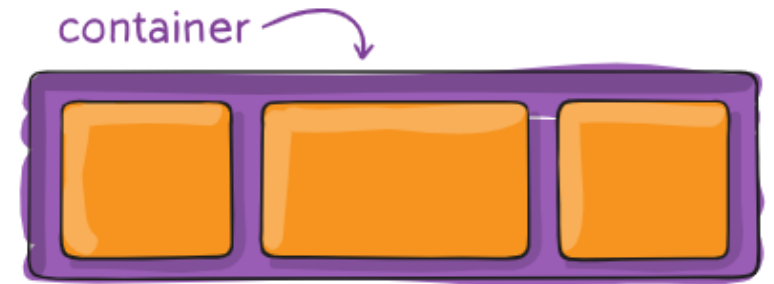
# CSS Flexbox: flex-wrap

- Standardmäßig versuchen Flexelemente alle auf eine Zeile zu passen. Durch Angabe von flex-wrap werden Elemente bei Bedarf auf die nächste Reihe verschoben
- nowrap (Standard): Alle Flexelemente werden auf einer Zeile angeordnet.
- wrap: Flexelemente werden auf mehreren Zeilen angeordnet, von oben nach unten.
- wrap-reverse: Flexelemente werden auf mehreren Zeilen von unten nach oben angeordnet.



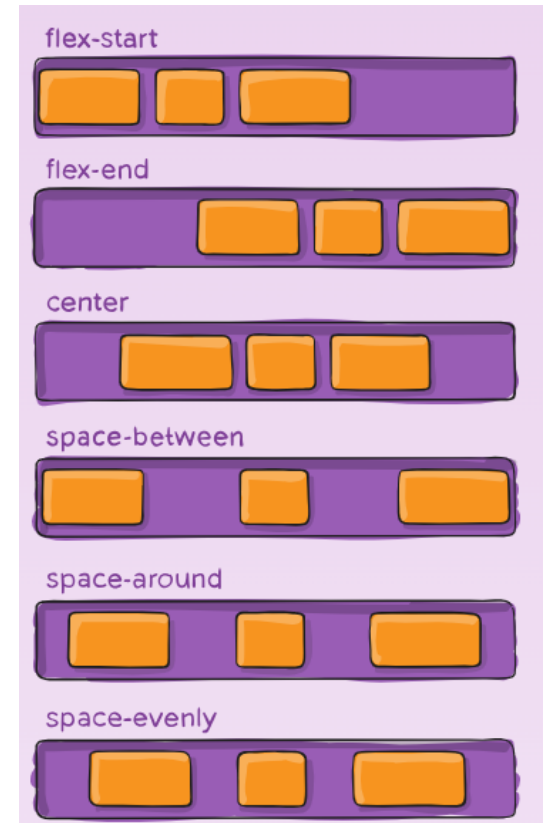
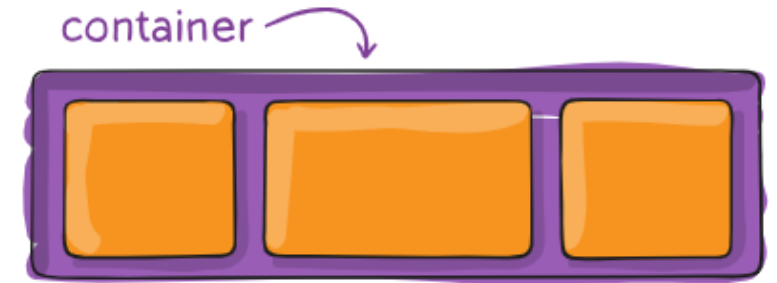
# CSS Flexbox: flex-flow

- Shorthand für die Eigenschaften „flex-direction“ und „flex-wrap“, die zusammen die Haupt- und Querachsen des Flexcontainers definieren.
- Der Standardwert ist „row nowrap“



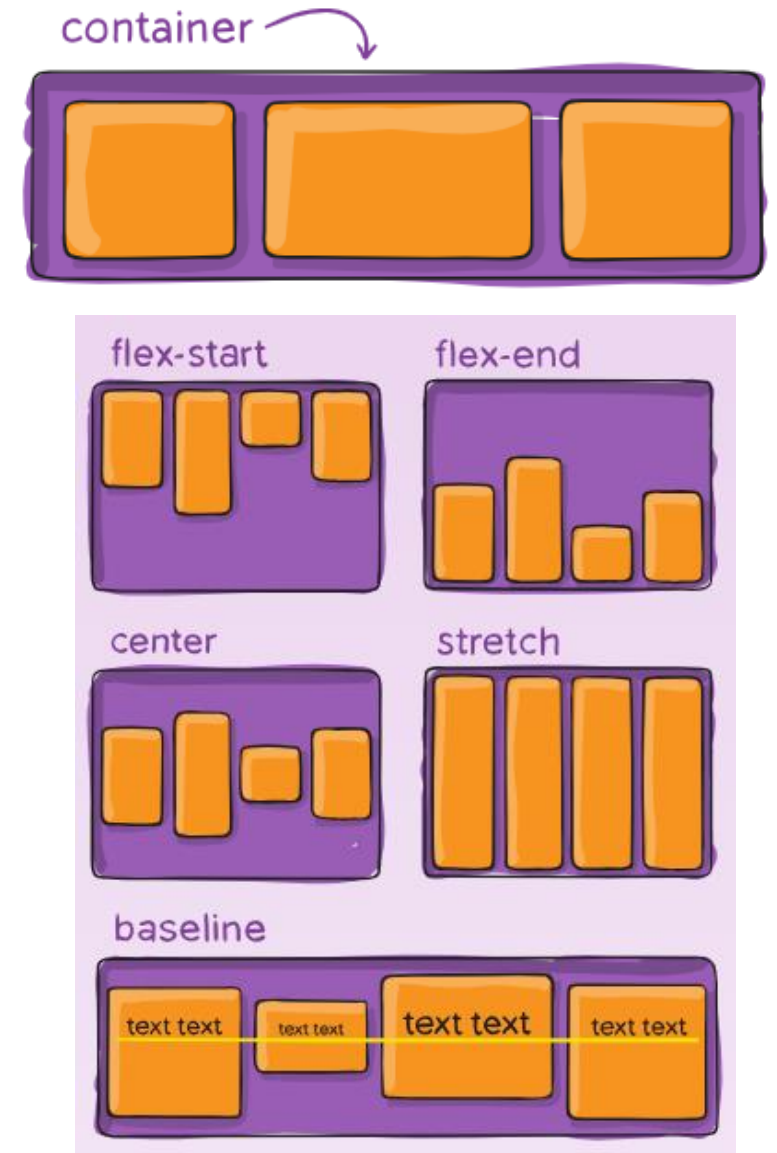
# CSS Flexbox: justify-content

- Definiert die horizontale Ausrichtung der Flexelemente auf der Hauptachse des Flexcontainers.
  - flex-start (Standard): Elemente werden zum Anfang der Flexrichtung ausgerichtet.
  - flex-end: Elemente werden zum Ende der Flexrichtung ausgerichtet.
  - center: Elemente werden in der Mitte der Linie zentriert.
  - space-between: Elemente werden gleichmäßig in der Linie verteilt; das erste Element befindet sich auf der Startlinie, das letzte Element auf der Endlinie.
  - space-around: Elemente werden so verteilt, dass der Abstand zwischen jedem beliebigen Paar von Elementen (und der Abstand zu den Rändern) gleich ist.
  - start, end, left und right können zusätzlich verwendet werden ist aber nicht empfehlenswert weil von einigen Browsern (Chrome) nicht unterstützt



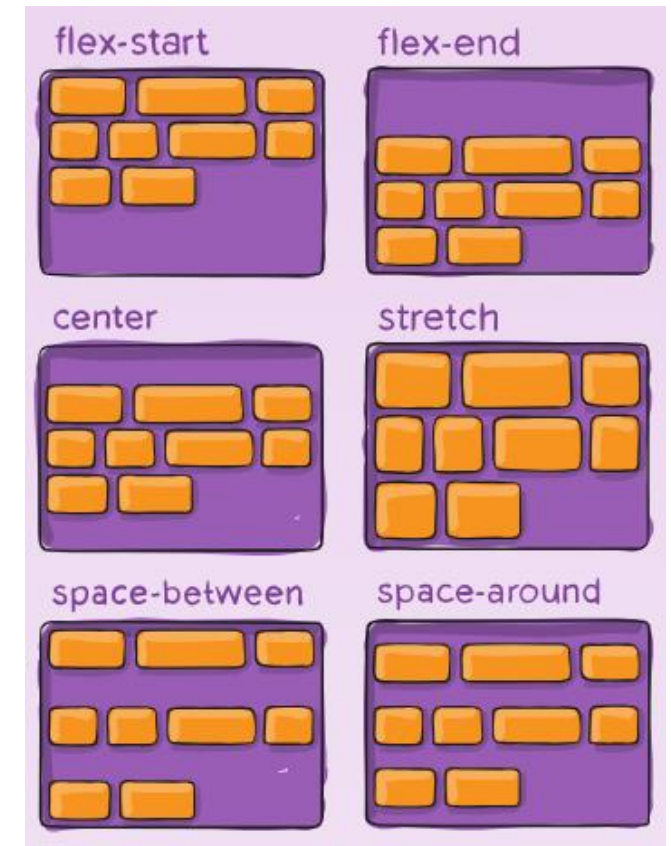
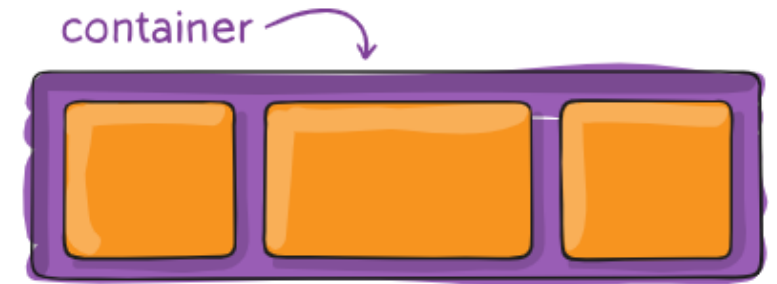
# CSS Flexbox: align-items

- Definiert das Standardverhalten dafür, wie Flexelemente auf der Querachse (senkrecht zur Hauptachse) in der aktuellen Zeile angeordnet werden.
  - stretch (Standard): Dehnen, um den Container auszufüllen (beachtet immer noch min-width/max-width).
  - flex-start: Elemente werden am Anfang der Querachse platziert.
  - flex-end: Elemente werden am Ende der Querachse platziert.
  - center: Elemente werden in der Querachse zentriert.
  - baseline: Elemente werden so ausgerichtet, dass ihre Grundlinien ausgerichtet sind.



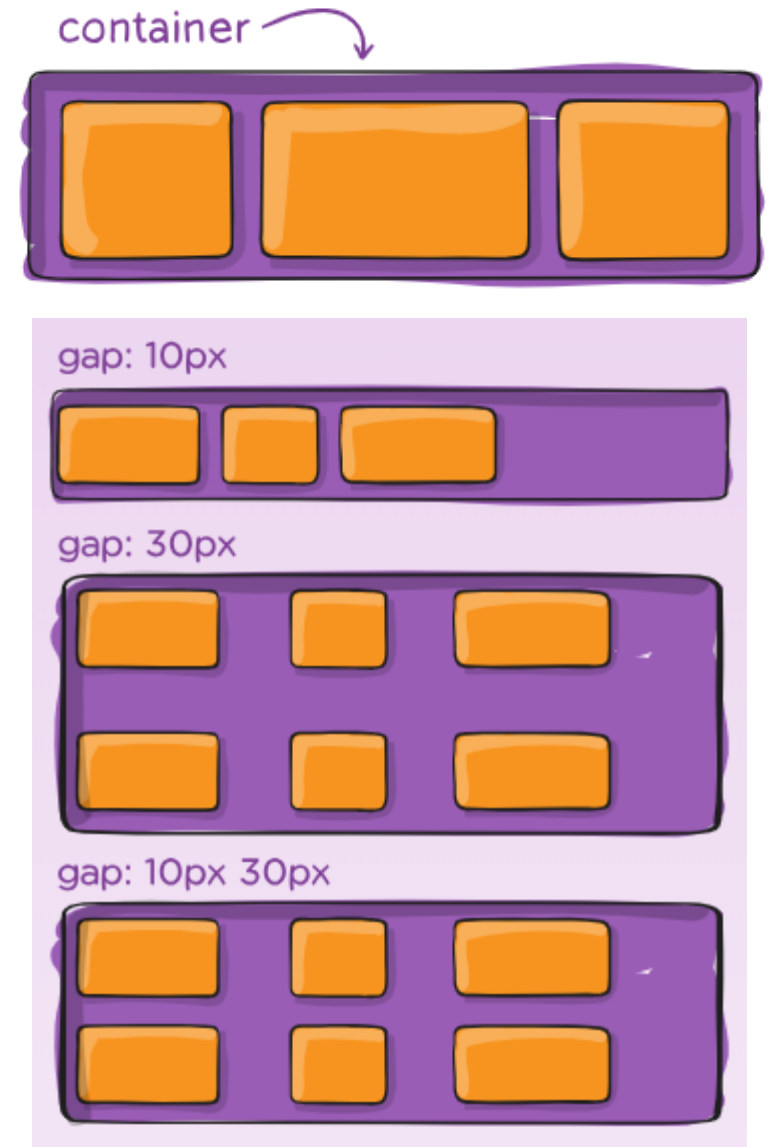
# CSS Flexbox: align-content

- Richtet die Zeilen eines Flexcontainers aus, wenn es zusätzlichen Platz auf der Querachse gibt.
- Diese Eigenschaft hat nur Auswirkungen auf mehrzeilige flexible Container, bei denen flex-wrap auf "wrap" oder "wrap-reverse" gesetzt ist. In einzeiligen Containern hat sie keine Wirkung
  - normal (Standard): Elemente werden in ihrer Standardposition angeordnet, als ob kein Wert festgelegt wäre.
  - flex-start / -end: Elemente werden am Anfang / Ende der flex-direction des Containers platziert
  - center: Elemente werden in der Mitte des Containers zentriert.
  - space-between: Elemente werden gleichmäßig verteilt; die erste Zeile ist am Anfang des Containers und die letzte Zeile am Ende.
  - space-around: Elemente werden gleichmäßig verteilt, wobei gleicher Raum um jede Zeile vorhanden ist.
  - space-evenly: Elemente werden gleichmäßig verteilt, wobei gleicher Raum um sie herum vorhanden ist.
  - stretch: Zeilen dehnen sich aus, um den verbleibenden Platz einzunehmen.



# CSS Flexbox: gap

- Die Eigenschaft `gap` steuert explizit den Abstand zwischen Flexelementen.
- Gilt nur zwischen den Elementen und nicht an den äußeren Rändern.
- Legt den Abstand zwischen den Flexelementen fest, ohne den Abstand zwischen den Elementen und dem Container rand zu beeinflussen.
- Definiert einen Mindestabstand . Wenn der Abstand zwischen den Elementen aufgrund von Eigenschaften wie justify-content: space-between; größer ist, wird die gap-Eigenschaft nur dann wirksam, wenn der Abstand zwischen den Elementen kleiner wäre.



# CSS Flexbox: Container Layout

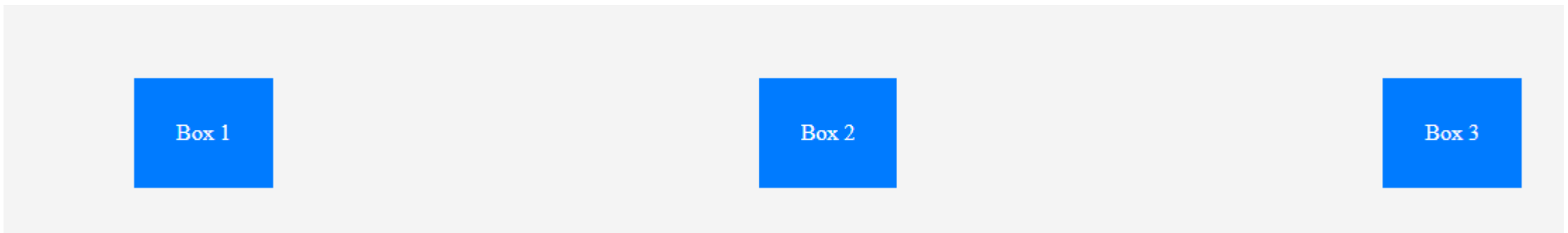
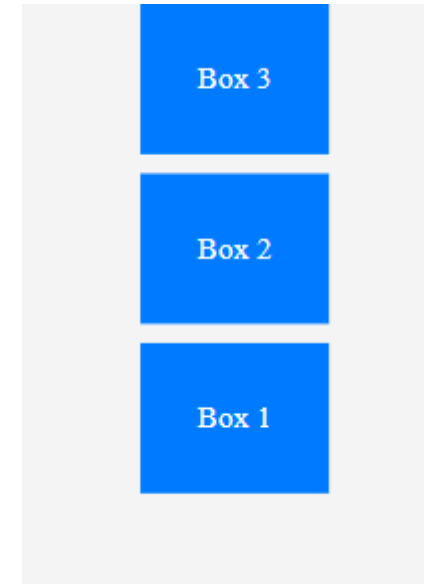


A CSS Flexbox Playground interface. At the top, a horizontal bar contains seven colored boxes labeled "Red", "Orange", "Yellow", "Green", "Blue", "Indigo", and "Purple" from left to right. Below this bar is a large, empty rectangular container with a yellow border. At the bottom of the interface, there are four controls: "Justify Content:" with a dropdown menu set to "Flex Start", "Flex Wrap:" with a dropdown menu set to "No Wrap", "Align Items:" with a dropdown menu set to "Flex Start", and "Align Content:" with a dropdown menu set to "Flex Start".

<https://appbrewery.github.io/flex-layout/>

# CSS Flexbox: Übung 1

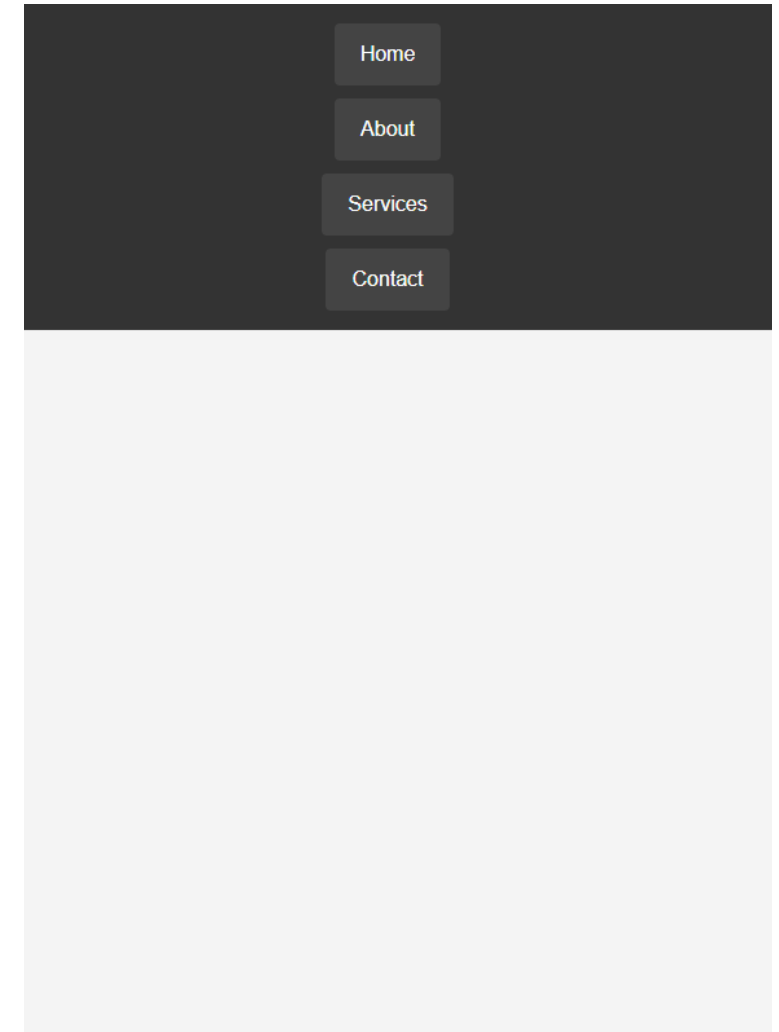
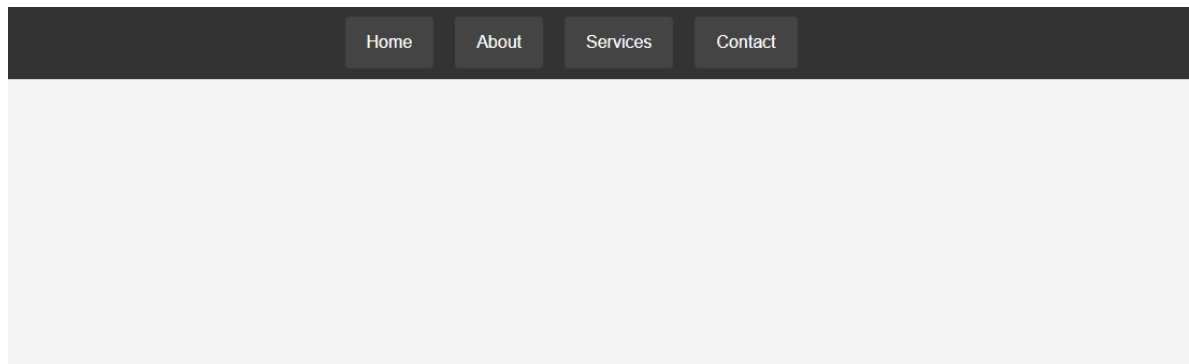
- Schreibe CSS Code um das Bild nebenan zu erzeugen. Nutze dafür dein Wissen über flexbox  
Ressourcen: </Flexbox&Grid/flexbox.html>
- Die Elemente sollen bei Bildschirmbreite größer als 600px nebeneinander mit gleichem Abstand bei weniger untereinander zentriert angeordnet werden
- Die Buttons sollen beim drüber hovern eine andere Farbe annehmen





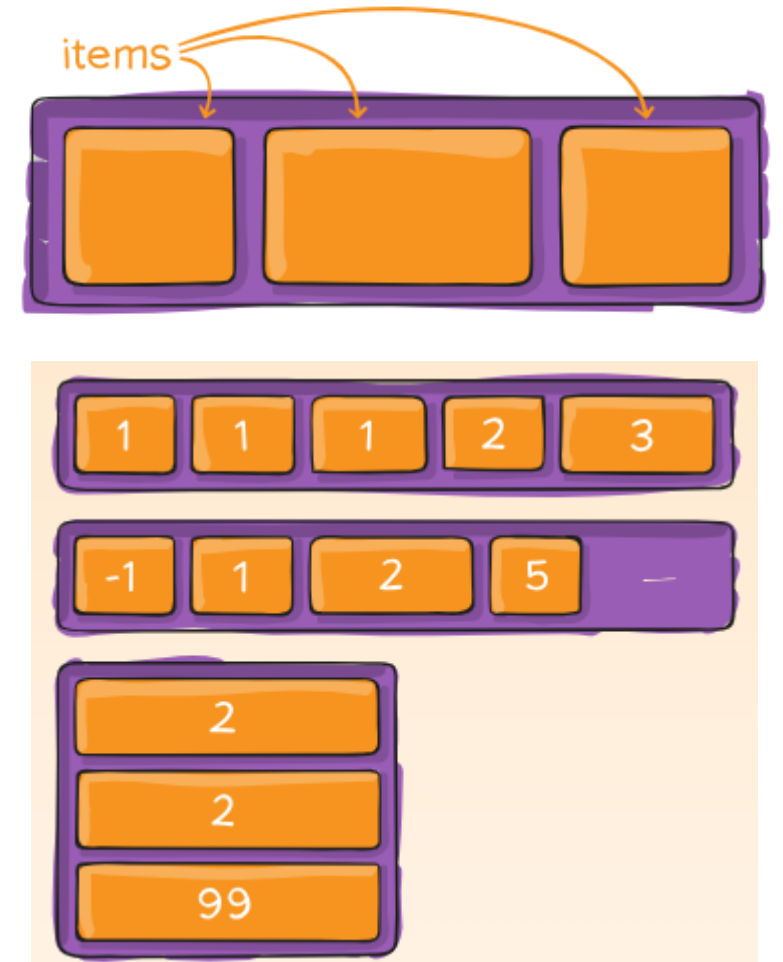
# CSS Flexbox: Übung 2

- Schreibe CSS Code um eine Navigationbar zu erzeugen. Nutze dafür dein Wissen über flexbox  
Ressourcen: /Flexbox&Grid/flexbar.html
- Die Elemente der Navigation sollen bei Bildschirmbreite größer als 600 px nebeneinander bei weniger untereinander zentriert angeordnet werden
- Die Buttons sollen beim drüber hovern eine andere Farbe annehmen



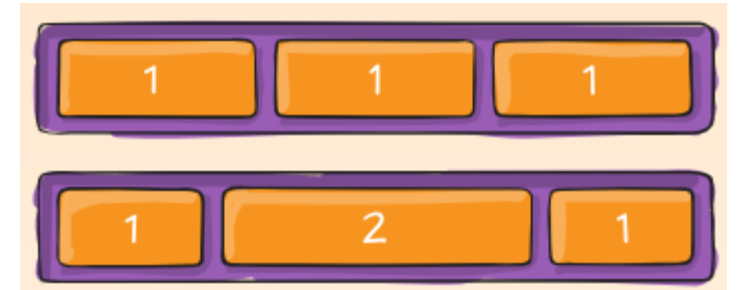
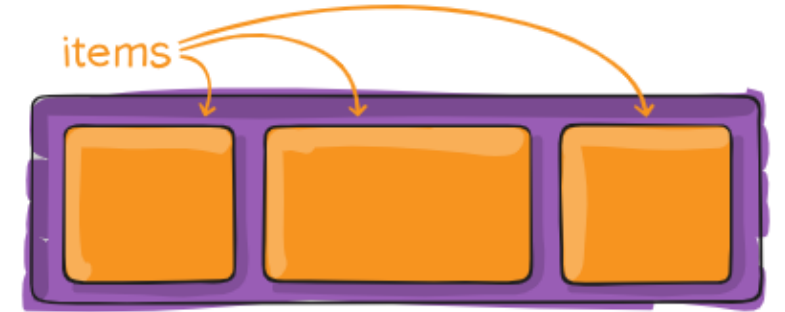
# CSS Flexbox: order

- Das Modul für das flexible Box-Layout erleichtert die Gestaltung flexibler, reaktionsfähiger Layoutstrukturen.
- Wird verwendet, um die Reihenfolge der Flexelemente unabhängig von ihrer ursprünglichen Reihenfolge im HTML-Code zu ändern. Elemente mit niedrigerer order-Wertung erscheinen zuerst, gefolgt von Elementen mit höherer Wertung.



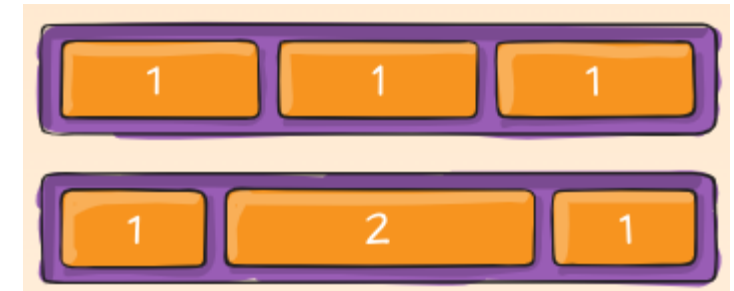
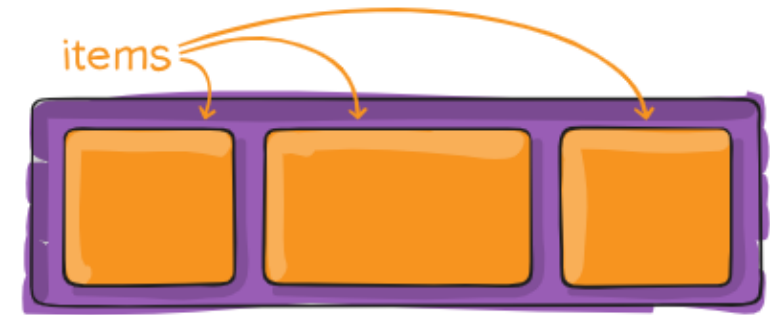
# CSS Flexbox: flexbox-grow

- Definiert die Fähigkeit eines Flexelements, sich bei Bedarf zu vergrößern.
- Es legt fest, welchen Anteil des verfügbaren Platzes im Flexcontainer das Element einnehmen sollte.
- Wenn alle Elemente `flex-grow` auf 1 gesetzt haben, wird der verbleibende Platz im Container gleichmäßig auf alle Kinder verteilt.
- Wenn eines der Kinder den Wert 2 hat, würde dieses Kind doppelt so viel Platz wie eines der anderen einnehmen



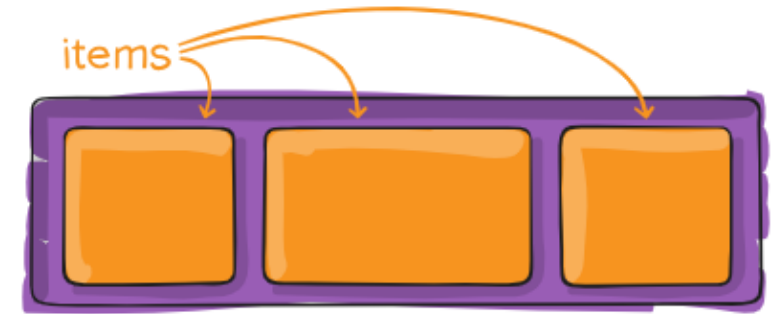
# CSS Flexbox: flexbox-shrink

- `flex-shrink` ist eine CSS-Eigenschaft, die steuert, wie stark ein Flexelement schrumpfen kann, wenn der verfügbare Platz nicht ausreicht.
- Ein Wert von 0 bedeutet, dass das Element nicht schrumpfen kann und seine ursprüngliche Größe beibehält.
- Ein Wert größer als 0 legt fest, wie stark das Element schrumpfen kann, relativ zu anderen Elementen im Container.
- Ein höherer Wert bedeutet, dass das Element stärker schrumpfen wird als Elemente mit niedrigerem Wert.
- Negative Werte sollten vermieden werden, da sie zu unvorhersehbaren Ergebnissen führen können.



# CSS Flexbox: flexbox-basis

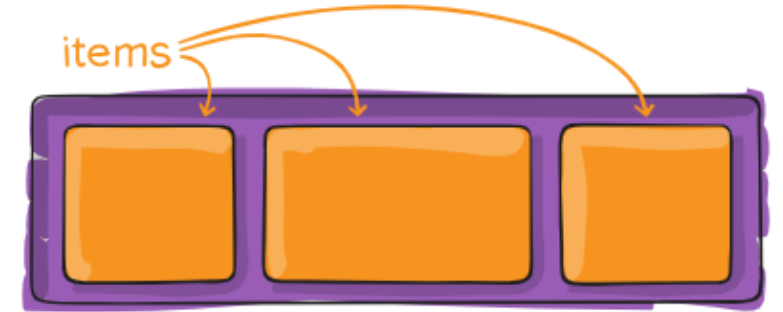
- Dies definiert die Standardgröße eines Elements, bevor der verbleibende Platz verteilt wird.
- Kann über folgende Arten definiert sein:
  - Längeneinheit (z. B. 20%, 5rem usw.) oder ein
  - Auto (Default): Die Länge entspricht der Länge des flexiblen Elements. Wenn für das Element keine Länge angegeben ist, wird die Länge entsprechend seinem Inhalt sein



```
<div class="flex-container">
  <div>1</div>
  <div>2</div>
  <div style="flex-basis: 200px">3</div>
  <div>4</div>
</div>
```

# CSS Flexbox: flex

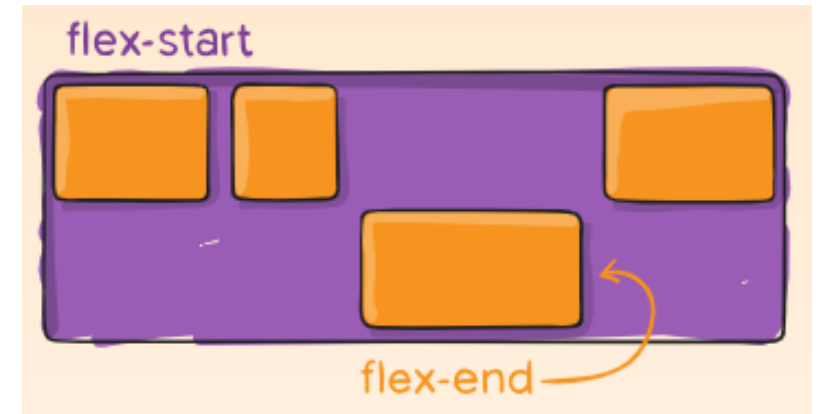
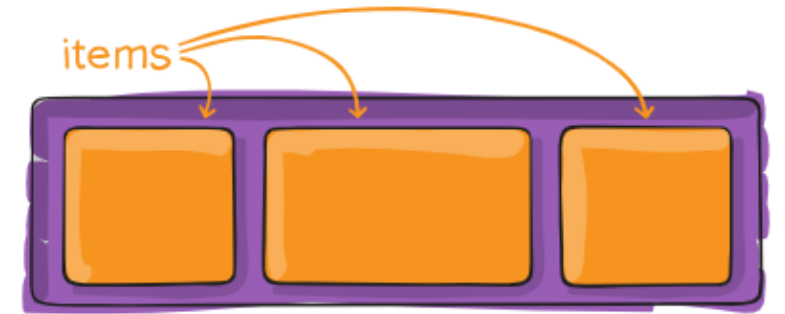
- Die Eigenschaft `flex` ist eine verkürzte Schreibweise, um die Eigenschaften `flex-grow`, `flex-shrink` und `flex-basis` in einer Zeile zu setzen. Sie erlaubt es, die Flexeigenschaften für Flexelemente auf eine kompakte Weise zu definieren.
- Die Syntax der `flex`-Eigenschaft ist:
- flex: [flex-grow] [flex-shrink] [flex-basis]
- Es wird empfohlen, die flex-Shorthand-Eigenschaft zu verwenden, anstatt die einzelnen Eigenschaften (flex-grow, flex-shrink und flex-basis) separat festzulegen.
- Die flex-Shorthand-Eigenschaft sorgt dafür, dass die anderen Werte intelligent gesetzt werden, was zu kürzerem und besser lesbarem CSS-Code führt.



```
<div class="flex-container">  
  <div>1</div>  
  <div>2</div>  
  <div style="flex: 0 0 200px">3</div>  
  <div>4</div>  
</div>
```

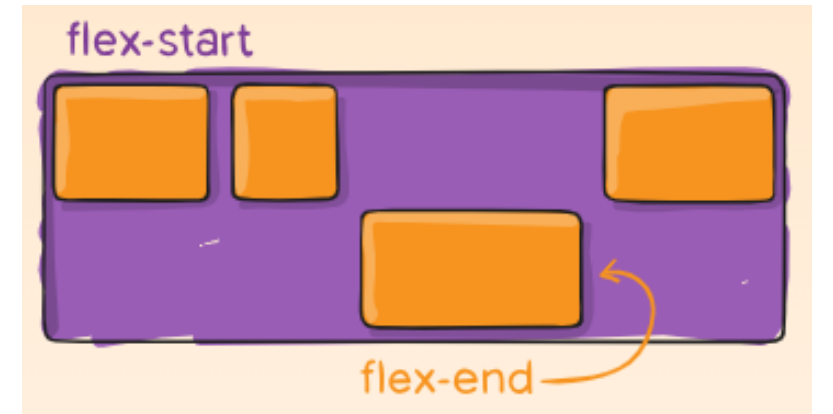
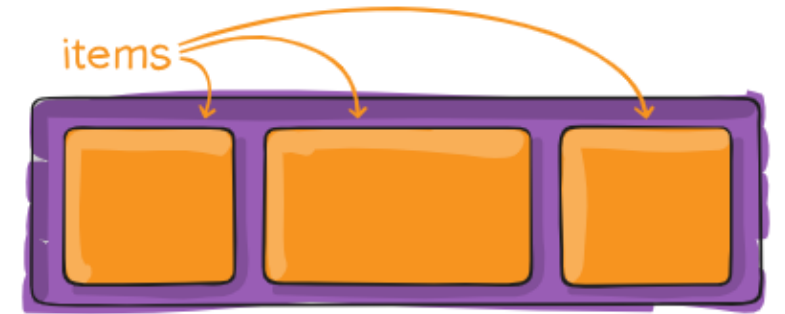
# CSS Flexbox: align-self

- Legt die Ausrichtung für das ausgewählte Element innerhalb des flexiblen Containers fest.
- Überschreibt die standardmäßige Ausrichtung, die durch die Eigenschaft align-items des Containers festgelegt wurde.



# CSS Flexbox: align-self

- Legt die Ausrichtung für das ausgewählte Element innerhalb des flexiblen Containers fest.
- Überschreibt die standardmäßige Ausrichtung, die durch die Eigenschaft align-items des Containers festgelegt wurde.





# CSS Flexbox: Flexboxfroggy

## FLEXBOX FROGGY

Manchmal reicht es nicht aus, die horizontale oder vertikale Ausrichtung eines Containers umzukehren. In so einem Fall können wir die **order**-Eigenschaft für einzelne Elemente verwenden. Standardmäßig haben alle Elemente den Wert 0, aber wir können die Eigenschaft verwenden, um ihn auf eine positive oder negative ganze Zahl zu setzen.

Benutze die **order**-Eigenschaft, um die Frösche entsprechend ihrer Seerosenblätter anzuordnen.

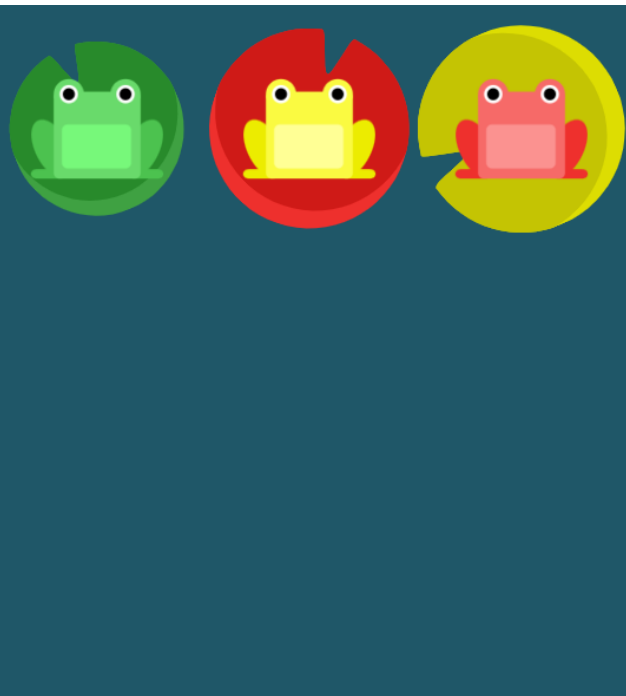
```
1 #pond {  
2   display: flex;  
3 }  
4  
5 .yellow {  
6  
7 }  
8  
9  
10
```

Weiter

Flexbox Froggy wurde gemacht von [Codepip](#) • [GitHub](#) • [Twitter](#) • [Einstellungen](#)

Möchtest du CSS grid lernen? Spiele [Grid Garden](#).

Level 14 von 24



<https://flexboxfroggy.com/#de>

# CSS Grid

- Das CSS Grid Layout-Modul bietet ein Rastersystem für das Layout, mit Zeilen und Spalten,
- Anwendung eher bei Layout-Strukturen in **2 Dimensionen**.
- Die Eigenschaft "**grid-template-columns**" definiert die Anzahl der Spalten in Ihrem Rasterlayout und kann die Breite jeder Spalte festlegen.
- Die Eigenschaft "**grid-template-rows**" definiert die Höhe jeder Zeile.
- Mit der Eigenschaft „**gap**“ wird der Abstand zwischen Zeilen und Spalten definiert (erster Wert = Zeile, zweiter Wert = Spalte)

```
.grid-container {
  display: grid;
  grid-template-columns: auto auto auto;
  grid-template-rows: 80px 200px;
  gap: 10px;
  background-color: #2196F3;
  padding: 10px;
}

.grid-container > div {
  background-color: rgba(255, 255, 255, 0.8);
  text-align: center;
  padding: 20px 0;
  font-size: 30px;
}
```

1	2	3
4	5	6

# CSS Grid: grid-template-columns/-rows



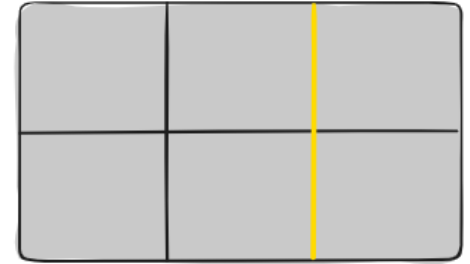
- **Grid-template-columns/-rows** definiert die Anzahl der Spalten in Ihrem Rasterlayout und kann die Breite jeder Spalte festlegen.
- Mögliche Werte für Spalten und Reihenbreite:
  - Feste Breiten: Sie können feste Breiten für Spalten in Pixeln (px), Zentimetern (cm), Millimetern (mm) oder anderen Einheiten angeben
  - Prozentuale Breiten: relativ zur Gesamtbreite des Containers. Beispiel: "25% 50% 25%" würde drei Spalten mit Breiten von 25%, 50% und 25% der Containerbreite erstellen.
  - Fr-Einheiten: Mit der Einheit "fr" (Fractional Unit) Spaltenbreiten in relativen Anteilen angeben.
  - Minmax-Funktion: Die "minmax()" Funktion ermöglicht, einen minimalen und maximalen Wert für die Spaltenbreite festzulegen. Beispiel: "minmax(100px, 1fr)" Spalte mit minimaler Breite von 100 Pixeln und maximaler Breite von 1 fr erstellen.
  - Repeat-Funktion: Die "repeat()" Funktion erstellt wiederholte Spalten mit denselben Breiten. Beispiel: "repeat(3, 1fr)" würde drei Spalten erstellen, die jeweils 1 fr breit sind.

```
.grid-container {
  display: grid;
  grid-template-columns: 80px 200px auto 30px;
  gap: 10px;
  background-color: #2196F3;
  padding: 10px;
}
```

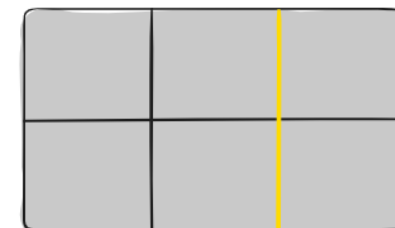
1	2	3	4
5	6	7	8

# CSS Grid: justify-content

- Die Eigenschaft "justify-content" wird verwendet, um das gesamte Raster innerhalb des Containers auszurichten.
  - "start" – richtet das Raster so aus, dass es bündig mit der Startkante des Rastercontainers ist.
  - "end" – richtet das Raster so aus, dass es bündig mit der Endkante des Rastercontainers ist.
  - "center" – richtet das Raster in der Mitte des Rastercontainers aus.
  - "stretch" – ändert die Größe der Rasterelemente, um das Raster zu strecken und den gesamten Breite des Rastercontainers auszufüllen.
  - "space-around" – platziert einen gleichmäßigen Abstand zwischen jedem Rasterelement, wobei die Ränder halb so groß sind wie die Zwischenräume an den äußeren Enden.
  - "space-between" – platziert einen gleichmäßigen Abstand zwischen jedem Rasterelement, ohne Abstand an den äußeren Enden.
  - "space-evenly" – platziert einen gleichmäßigen Abstand zwischen jedem Rasterelement, einschließlich der äußeren Enden.



# CSS Grid: justify-content



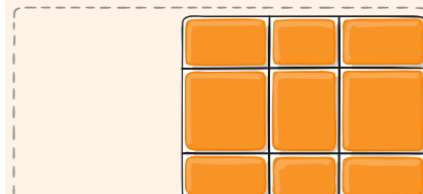
```
.container {
  justify-content: start;
}
```

grid container



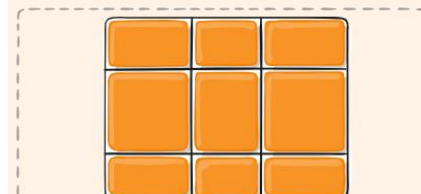
```
.container {
  justify-content: end;
}
```

grid container



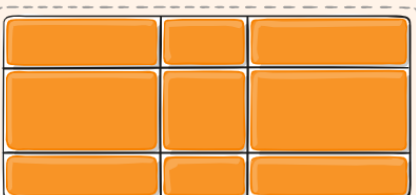
```
.container {
  justify-content: center;
}
```

grid container



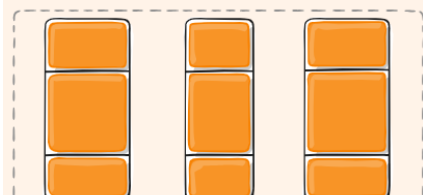
```
.container {
  justify-content: stretch;
}
```

grid container



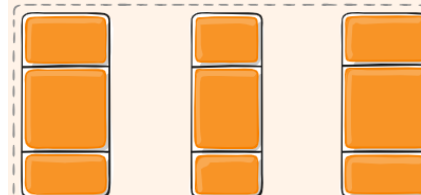
```
.container {
  justify-content: space-around;
}
```

grid container

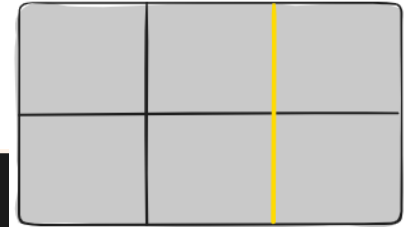


```
.container {
  justify-content: space-between;
}
```

grid container

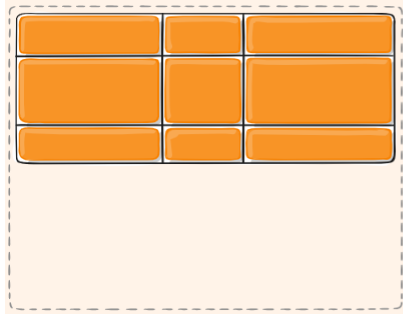


# CSS Grid: align-content



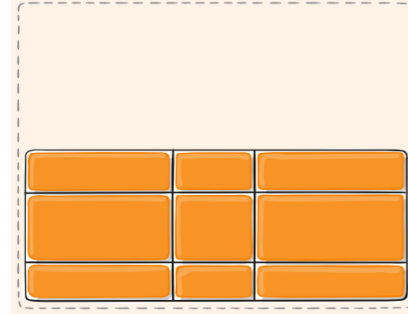
```
.container {  
  align-content: start;  
}
```

grid container



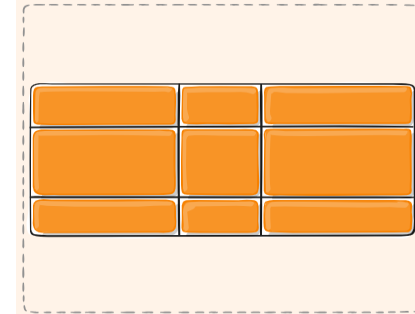
```
.container {  
  align-content: end;  
}
```

grid container



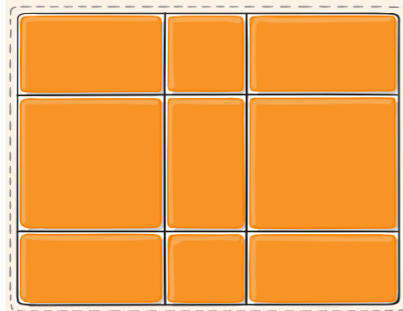
```
.container {  
  align-content: center;  
}
```

grid container



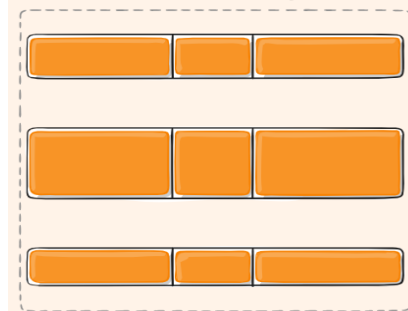
```
.container {  
  align-content: stretch;  
}
```

grid container



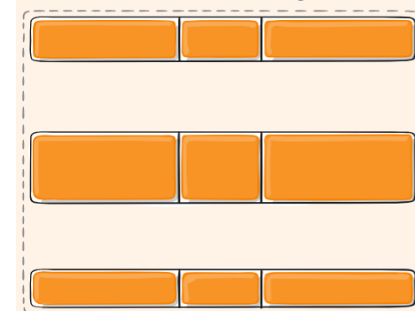
```
.container {  
  align-content: space-around;  
}
```

grid container



```
.container {  
  align-content: space-between;  
}
```

grid container

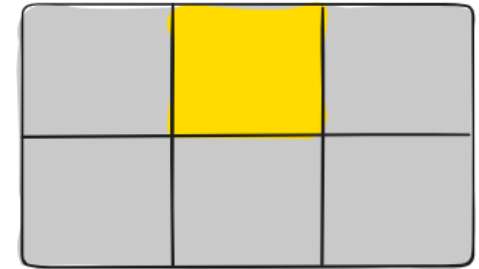


# CSS Grid: grid-column

- Die Eigenschaft "grid-column" legt fest, auf welcher Spalte oder welchen Spalten ein Element platziert wird.
- Sie definieren, wo das Element beginnen soll und wo es enden soll.
- Die Eigenschaft "grid-column" ist eine Kurzform (Shorthand-Eigenschaft) für die Eigenschaften "grid-column-start" und "grid-column-end"

```
.item1 {
  grid-column: 1 / 5;
}
```

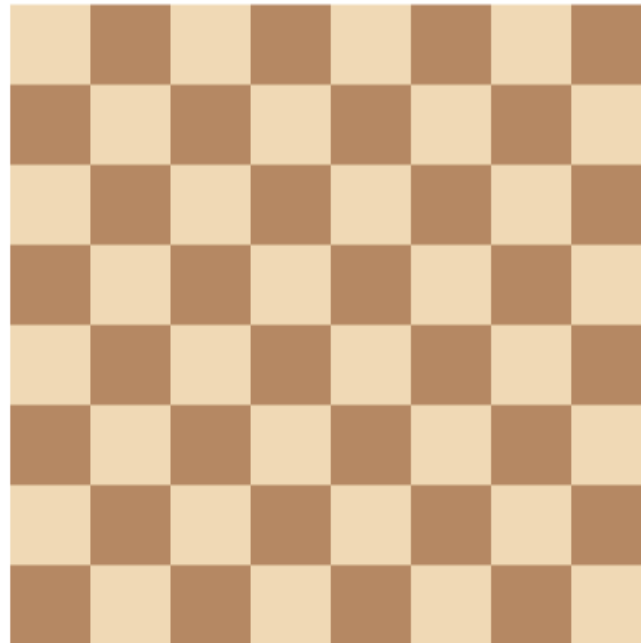
```
.item1 {
  grid-column: 1 / span 3;
}
```



1		
5	6	7
11	12	13

# CSS Grid Übung Schachbrett

- Ressourcen: Flexbox&Grid: schach.html
- Schreibe CSS Code um ein Schachbrett erscheinen zu lassen





# CSS Grid Garden

## GRID GARDEN

Level 1 von 28

Willkommen bei Grid Garden, wo du CSS Code schreibst, um deinen Karottengarten zu kultivieren! Wässere nur die Abschnitte, welche Karotten enthalten, indem du die `grid-column-start` Eigenschaft nutzt.


Zum Beispiel `grid-column-start: 3;` wässert den Abschnitt auf der dritten vertikalen Rasterlinie. Mit anderen Worten: die 3. Spalte von links im Raster.

```
1 #garden {
2   display: grid;
3   grid-template-columns: 20% 20% 20% 20% 20%;
4   grid-template-rows: 20% 20% 20% 20% 20%;
5 }
6
7 #water {
8   
9 }
10
11
12
13
14
```

Weiter

Grid Garden wurde gemacht von [Codeplop](#) • [YouTube](#) • [Twitter](#) • [GitHub](#) • [Deutsch](#)

Du möchtest CSS Flexbox lernen? Spiele [Flexbox Froggy](#).



<https://cssgridgarden.com/#de>

**Viel Erfolg beim Entwickeln!**

