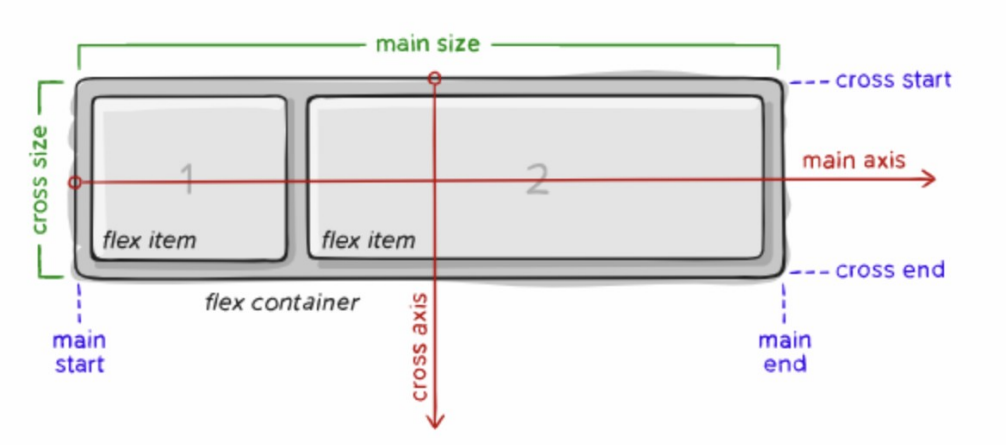
Flexbox Layout Module

Flexbox Layout Module

Das **Flexbox-Layout-Modul** zielt darauf ab, eine effizientere Methode zur Anordnung, Ausrichtung und Verteilung des Platzes zwischen den Elementen in einem Container bereitzustellen, selbst wenn deren Größe unbekannt und/oder dynamisch ist (daher das Wort "Flex").

Die Hauptidee hinter dem Flex-Layout ist es, dem Container die Möglichkeit zu geben, die Breite/Höhe (und die Reihenfolge) seiner Elemente zu ändern, um den verfügbaren Platz bestmöglich auszufüllen (hauptsächlich, um sich an alle Arten von Anzeigegeräten und Bildschirmgrößen anzupassen)

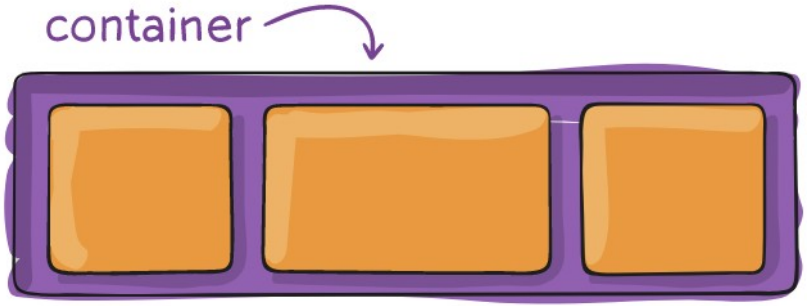
Flexbox Lavout Module



Die **MAIN-AXIS** eines Flex-Containers ist die primäre Achse, entlang der die Flex-Elemente angeordnet werden. Sie muss nicht unbedingt horizontal sein; dies hängt von der Eigenschaft flex-direction ab.

Die **CROSS-AXIS** steht senkrecht zur Hauptachse und wird Querachse genannt. Ihre Richtung hängt von der Richtung der Hauptachse ab.

Flexbox - Flex-Container

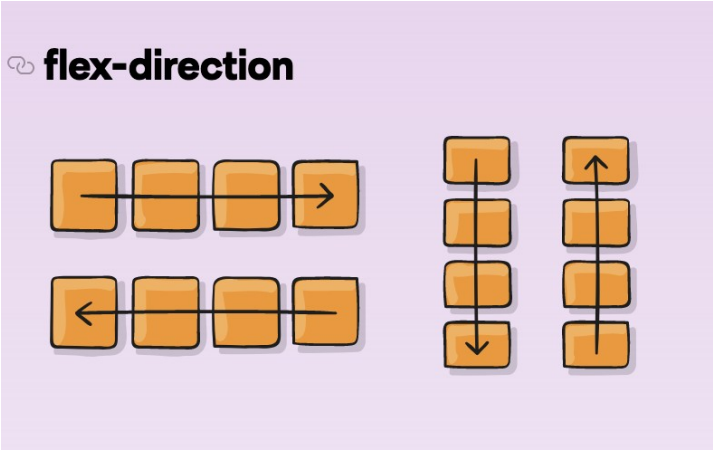



```
div {
   display: flex | inline-flex;
}
```

Definiert einen Flex-Container; Block oder inline.

Er aktiviert einen Flex-Kontext für alle seine direkten Kinder.

Flexbox - Flex-Container

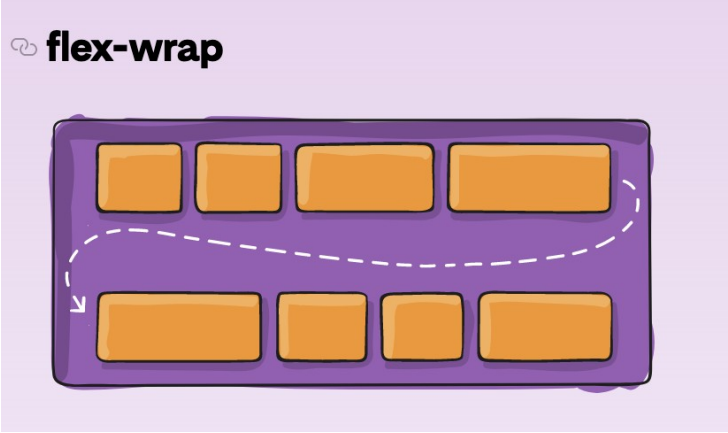



```
div {
   flex-direction: row | row-reverse |
                   column | column-reverse;
}
```

row (default): inks nach rechts
row-reverse: von rechts nach links
column: oben nach unten
column-reverse: von unten nach oben

Festlegung der **MAIN-AXIS**.
Bei row | row-reverse: horizontal.
Bei column | column-reverse: vertikal.

Flexbox - Flex-Container



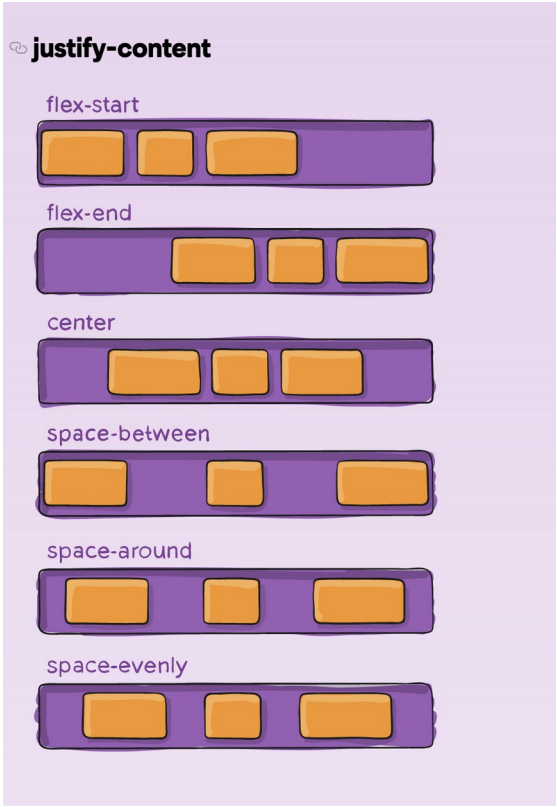

```
div {
   flex-wrap: nowrap | wrap | wrap-reverse
}
```

nowrap (default):
Alle Flex-Kinder in einer Zeile (möglicher overflow).

wrap:
Flex-Kinder werden auf mehrere Zeilen umbrochen, von oben nach unten.

wrap-reverse:
Flex-Kinder werden in mehreren Zeilen von unten nach oben umbrochen.

Flexbox - Flex-Container



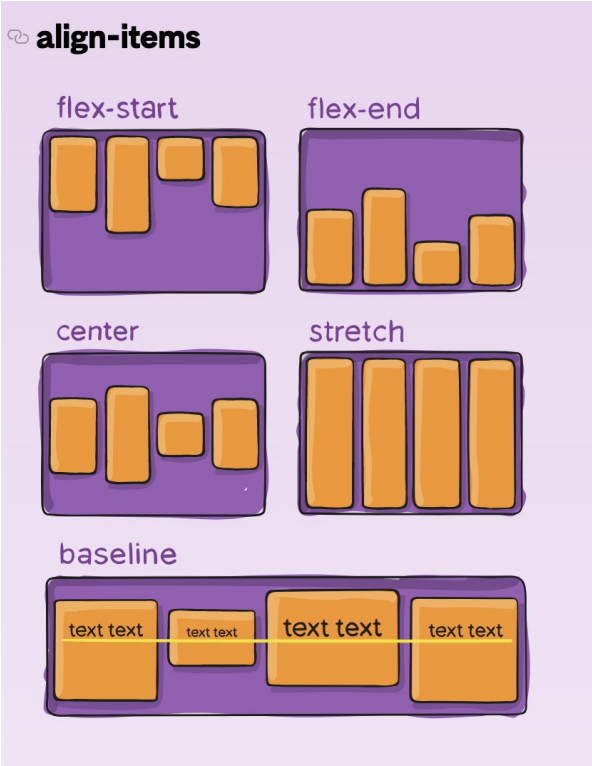

```
div {
   justify-content:
      flex-start | flex-send | center |
      space-between | space-around | space-evenly
}
```

flex-start (default)

Richtet die Flex-Kinder auf der **MAIN-AXIS** aus.

Bei row | row-reverse horizontal.
Bei column | column-reverse vertikal.

Flexbox - Flex-Container



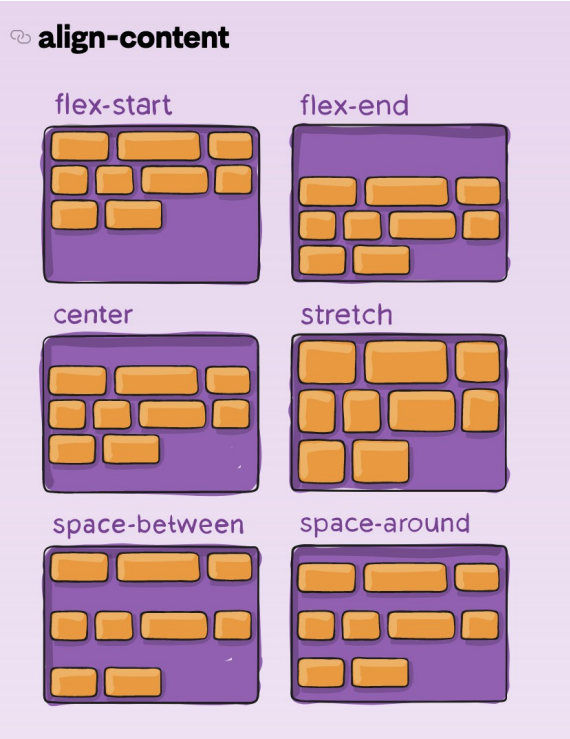

```
div {
   align-items:
      flex-start | flex-send | center |
      stretch | baseline
}
```

stretch (default)

Richtet die Flex-Kinder auf der **CROSS-AXIS** aus.

Bei row | row-reverse vertikal.
Bei column | column-reverse horizontal.

Flexbox - Flex-Container



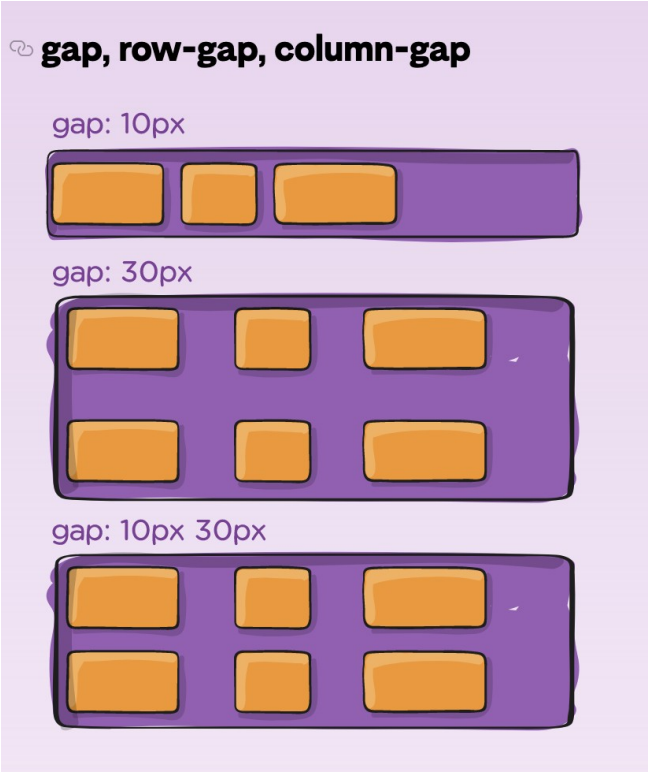

```
div {
   align-content:
      flex-start | flex-send | center |
      space-between | space-around
}
```

flex-start (default)

Richtet die Flex-Kinder in **MULTI-LINE** Flex-Containers aus.

Also bei `flex-wrap: wrap | wrap-reverse`
Nicht bei `flex-wrap: nowrap`

Flexbox - Flex-Container



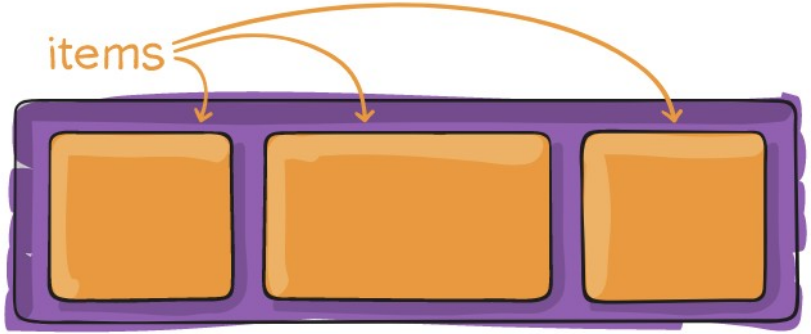

```
div {
  gap: 10px; /* both */
  gap: 10px 20px; /* row-gap column-gap */

  row-gap: 10px;
  column-gap: 20px;
}
```

Steuert explizit den Abstand zwischen Flex-Kinder.

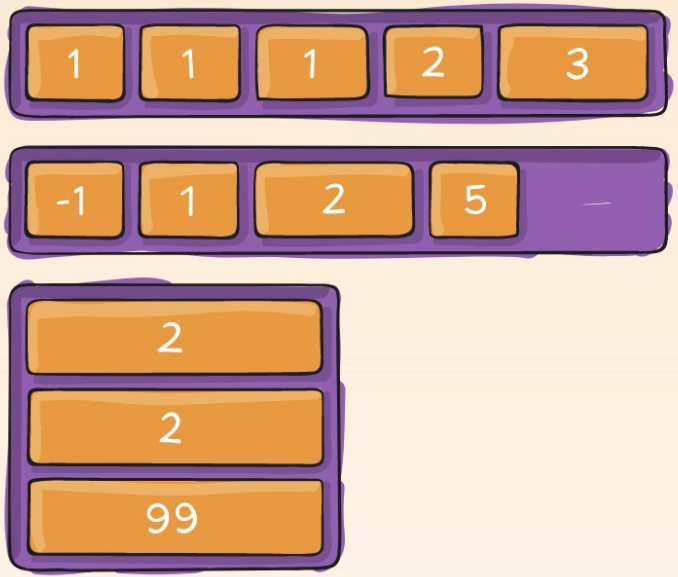
Dieser Abstand wird nur zwischen den Elementen und nicht an den Außenkanten angewendet.

Flexbox - Flex-Kinder



Flexbox - Flex-Kinder

order

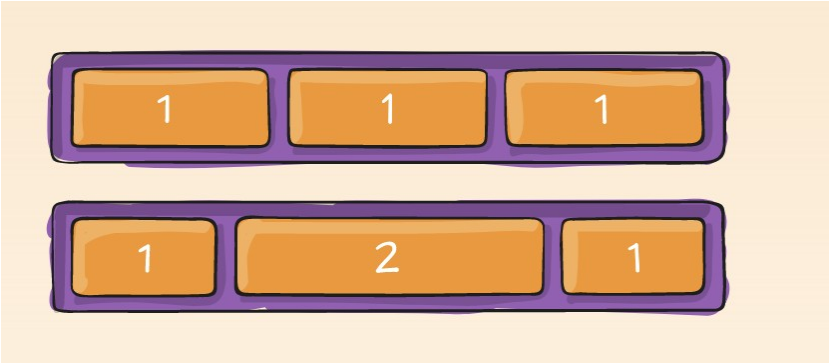



```
div {
   order: number
 }
```

Standardmäßig werden die Flex-Elemente in der Reihenfolge im DOM-Tree angeordnet.

Order steuert die Reihenfolge, in der sie im Flex-Container erscheinen.

Flexbox - Flex-Kinder



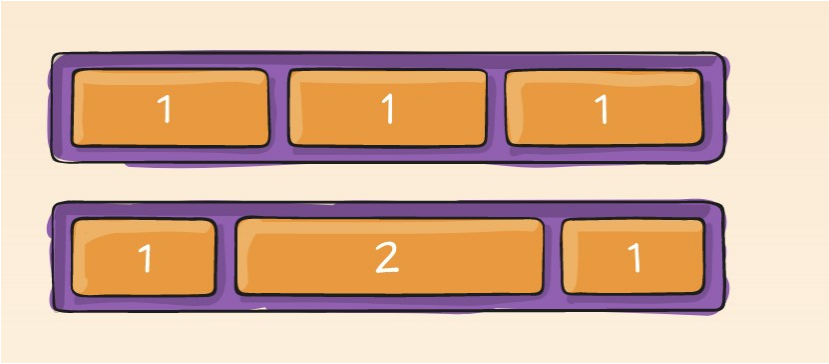

```
div {
   flex-grow: number
 }
```

flex-grow (default 0):
Definiert die Fähigkeit eines Flex-Kindes, bei Bedarf zu wachsen.

Wenn alle Flex-Kinder auf 1 eingestellt sind, wird der verbleibende Platz im Container gleichmäßig auf alle Kinder verteilt.

Wenn ein Flex-Kind auf 2 und andere auf 1 eingestellt sind, nimmt dieses Element doppelt so viel Platz in Anspruch wie eines der anderen.

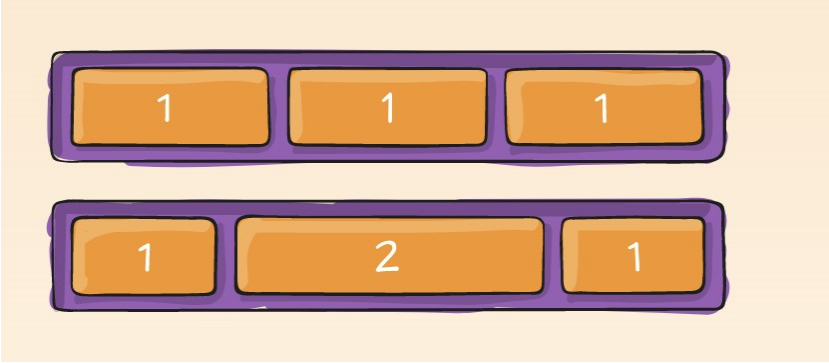
Flexbox - Flex-Kinder




```
div {
   flex-shrink: number
}
```

flex-shrink (default 1):
Definiert die Fähigkeit eines Flex-Kindes, bei Bedarf zu schrumpfen.

Flexbox - Flex-Kinder




```
div {
   flex-basis: px | % | auto
}
```

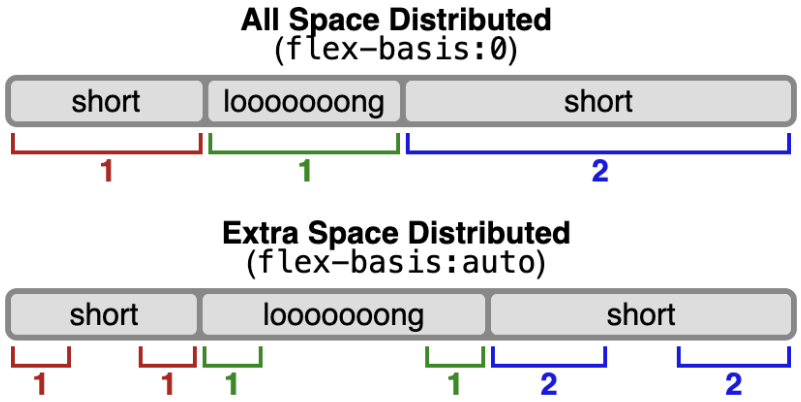
flex-basis
Wunsch-Größe eines Elements, bevor der verbleibende Platz verteilt wird. Es kann eine Länge (z.B. 100%, 0px, etc.) oder auto sein.

flex-basis: auto (default)
“Ich möchte so groß wie mein Inhalt sein”

flex-basis: 0px
“Ich möchte 0px groß sein”
Macht gemeinsam mit **flex-grow** Sinn.

flex-basis: 100%
“Ich möchte so breit wie der Container sein”
Macht gemeinsam mit **flex-wrap: wrap** Sinn.
-> Siehe Flexbox - Multiline

Flexbox - Flex-Kinder



flex-basis: 0
Der ganze Flex-Container (**ALL SPACE**) wird proportional auf die Flex-Kinder verteilt. In abhängig ihres flex-grow Wertes.

flex-basis: auto
Der extra Bereich des Flex-Containers (**EXTRA-SPACE**) wird proportional auf die Flex-Kinder verteilt. In abhängigkeit ihres flex-grow Wertes.

Flex-Algorithmus: Zuerst Basis -> dann shrink -> dann grow

Flexbox - Flex-Shorthand

Three values: flex-grow & flex-shrink & flex-basis
flex: 0 0 100%;

One value unitless number: flex-grow
flex: 1;
-> flex: 1 0 0;

Flexbox - Flex-Shorthand

flex: 0 1 auto (default)
“I want to be the width of my real content …. BUT if necessary I will shrink”
-> This prevents overflowing the line

flex: 1 0 auto
“I want to be the width of my real content …. BUT if possible I will grow.”
-> This might overflow the line (DANGEROUS !)

flex: 1 1 auto
“I want to be the width of my real content … BUT let’s see if I have to grow or shrink”

flex: 0 0 auto
“I will be the width of my real content …. YOU BASTARDS, don’t touch me !!!”
-> This might overflow the line (DANGEROUS !)

Flexbox - Flex-Shorthand

flex: 1 0 0px
flex: 1
“I will start with my absolute minimum size … BUT if possible I will grow”
-> Put this on all childs and they will grow equal in size

flex: 0 0 100%
“I will take the entire flex-line for me
-> Makes sense in ‘flex-wrap: wrap’

Flexbox - with Response Media

Hiking in Newfoundland Photo: Courtesy of Claire Volkman

Unesco World Heritage Sites
Offering UNESCO World Heritage Sites, sprawling national parks, picturesque seaside villages, a world-class food scene, and quite possibly the country's friendliest locals, you'd think Newfoundland would be swarming with tourists. However, this Canadian island that sits off the eastern coast of the country, flanking Nova Scotia and Prince Edward Island, sees a mere fraction of the visitors of British Columbia and Alberta.

1) Response media

```
img, video {
   max-width: 100%;
   height: auto;
   /*remove bottom gap from
    img/video */
   vertical-align: bottom;
}
```

2) Wrap img/video inside another tag

```
<div><img src=””></div>
```

-> img/video has “intrinsic content” :-(
-> max-width will obey his “master” the wrapper-tag :-)
-> the wrapper-tag is controlled by flexbox

3) Flex-Container

```
.intro-article {
   display: flex;
   flex-direction: row;
   align-items: center;
}
```

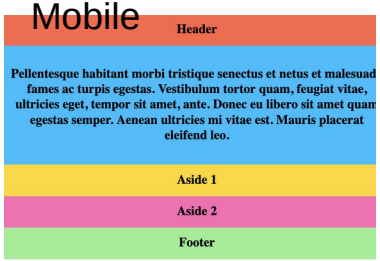
4) Flex-Kinder

```
.intro-article__content,
.intro-article__media {
   flex: 1;
}
```

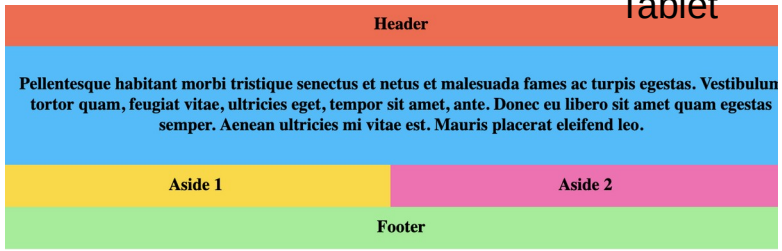
Flexbox - Multiline

Multiline

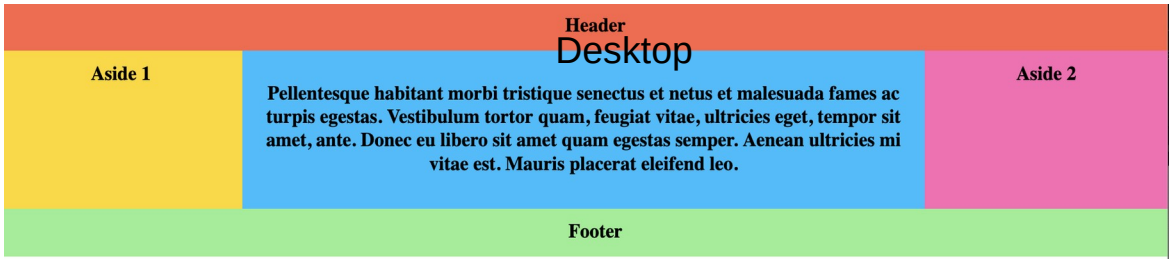
Mobile



Tablet



Desktop




```html
<div class="wrapper">
  <header class="header">Header</header>
  <article class="main">...</article>
  <aside class="aside aside-1">Aside 1</aside>
  <aside class="aside aside-2">Aside 2</aside>
  <footer class="footer">Footer</footer>
</div>
```

1) Flex-Container

```css
.wrapper {
   display: flex;
   flex-direction: row;
   flex-wrap: wrap;
}
```

2) Flex-Kinder - Mobile First

```css
.wrapper > * {
    flex: 0 0 100%;
}
```

3) Flex-Kinder - Tablet

```css
@media all and (min-width: 600px) {
   .aside { flex: 1 0 0; }
}
```

4) Flex-Kinder - Desktop

```css
@media all and (min-width: 800px){
   .main { flex: 3 0 0; }
   .header {order: 1; }
   .aside-1 { order: 2; }
   .main { order: 3; }
   .aside-2 { order: 4; }
   .footer { order: 5; }
}
```

Flexbox - Summary

Flex-Container
display: flex | inline-flex

flex-direction: row (**default**) | column | row-reverse | column-reverse
flex-wrap: nowrap (**default**) | wrap | wrap-reverse
flex-flow: row nowrap (**default**); **/* Shorthand for flex-direction & flex-wrap */**

justify-content: flex-start (**default**) | center | flex-end | space-around | space-between | space-evenly
align-items: flex-start | center | flex-end | stretch (**default**) | baseline
align-content: flex-start (**default**) | center | flex-end | stretch

Flex-Kinder
flex-grow: number
flex-shrink: number
flex-basis: auto | px | %
flex: 0 1 auto (**default**) **/* Shorthand for flex-grow & flex-shrink & flex-basis */**