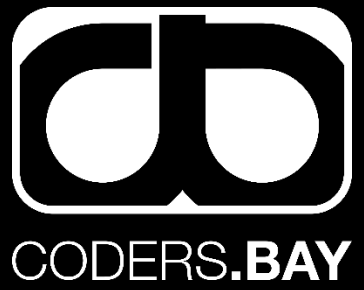


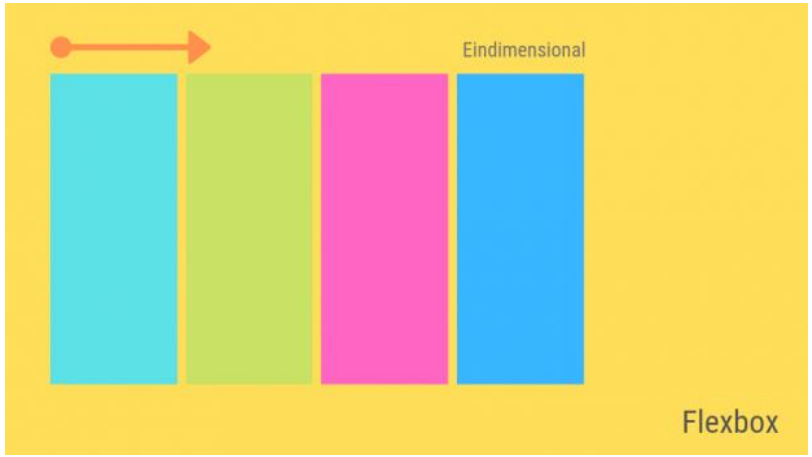
CSS Grid

Flexbox vs Grid

CODERS.BAY

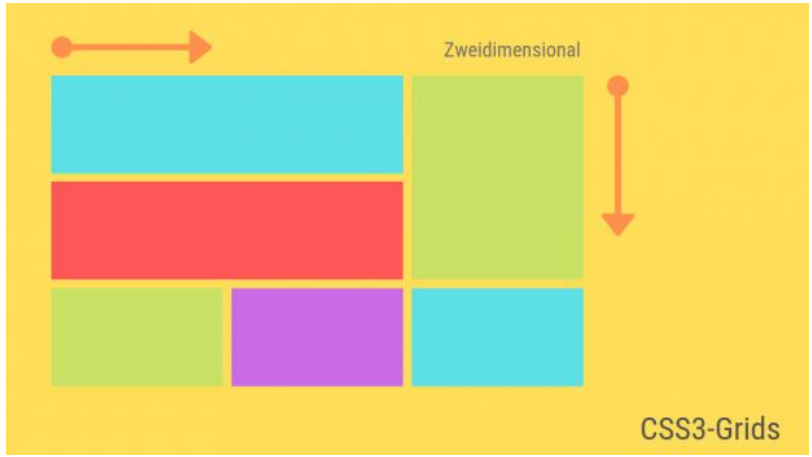
FLEXBOX

- Eindimensionales System
- Content First
- Bei kleinteiligen Inhalten flexibler
- Layout innerhalb einer Zeile



GRID

- 2 Dimensionales System
- Layout First
- Für das große Ganze
- Gut für Raster



Quelle: https://t3n.de/news/css3-unterschiede-grids-flexbox-1169472/2/

CSS GRID

CODERS.BAY

- **Das sog. »CSS Grid« ist eine wesentliche Techniken zur Gestaltung von Layouts mit CSS.**
- Es ist ein 2-Dimensionales System, das mit Spalten (columns) und Reihen (rows) arbeitet

```
<div class="grid-container">
    <header></header>
    <main></main>
    <aside><aside>
    <footer></footer>
</div>
```

- Mit `display:grid;` auf dem Container-Element, wird das Grid aktiviert.

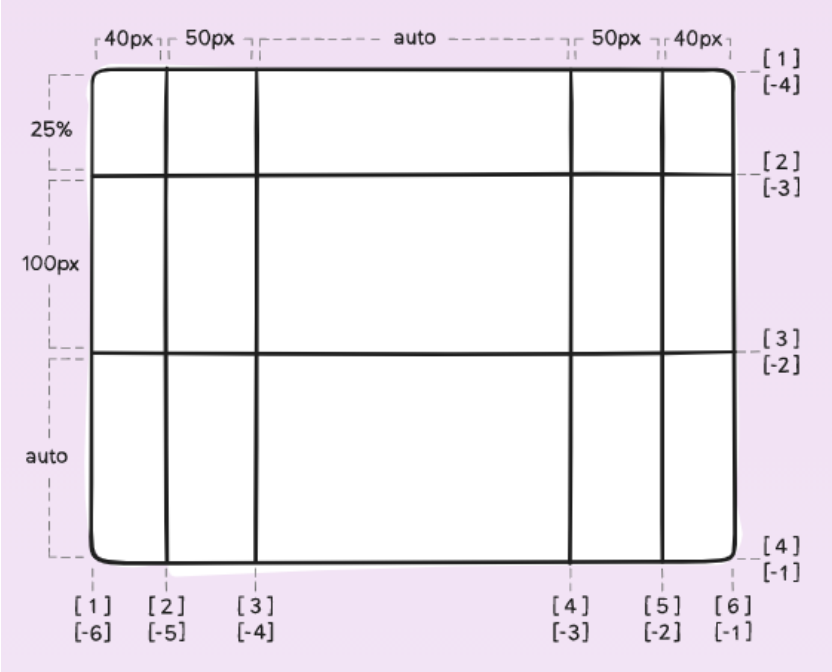
```
.grid-container { display:grid; }
```

Quelle: https://blog.kulturbanause.de/2013/07/einfuhrung-in-das-flexbox-modell-von-css/

Spalten und Reihen

- Die Spalten und Reihen geben dem Grid seine Form
- Zusätzlich können zwischen den Werten Namen für die Linien ergänzt werden
- Wenn man den Linien keine Namen geben möchte werden die Werte mit einem Leerzeichen getrennt:

```
.grid-container {
    grid-template-columns: 40px 50px auto 50px 40px;
    grid-template-rows: 25% 100px auto;
}
```

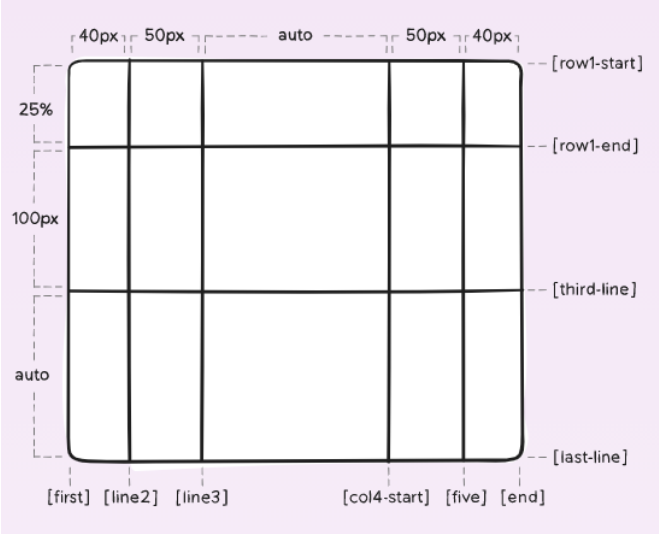


Quelle: https://blog.kulturbanause.de/2013/07/einfuhrung-in-das-flexbox-modell-von-css/

Spalten und Reihen

- Die Spalten und Reihen mit expliziten Liniennamen

```
.grid-container {
    grid-template-columns: [first] 40px [line2] 50px [line3] auto [col4-start] 50px [five] 40px [end];
    grid-template-rows: [row1-start] 25% [row1-end] 100px [third-line] auto [last-line];
}
```



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

CONTAINER

CODERS.BAY

- Man kann Linien mehrere Namen geben [row1-end row2-start]
- Wenn man gleiche Spalten erstellen möchte kann man repeat() nutzen

```
.grid-container {
    grid-template-columns: repeat(3, 20px [col-start]);
}
```

- Mit der fr Einheit kann man Fraktionen mit freiem Platz erstellen

```
.grid-container {
    grid-template-columns: 1fr 1fr 1fr;
}
```

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

CONTAINER

CODERS.BAY

- Mit column-gap und row-gap kann man die Größe der Zwischenräume angeben

```
.grid-container {
    row-gap: 10px;
    column-gap: 15px;
}
```

```
.grid-container {
    gap: <row-gap> <column-gap>
}
```

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

Beispiel


```
<div class="grid-container">
    <div></div>
    <div></div>
    <div></div>
    <div></div>
    <div></div>
</div>
```

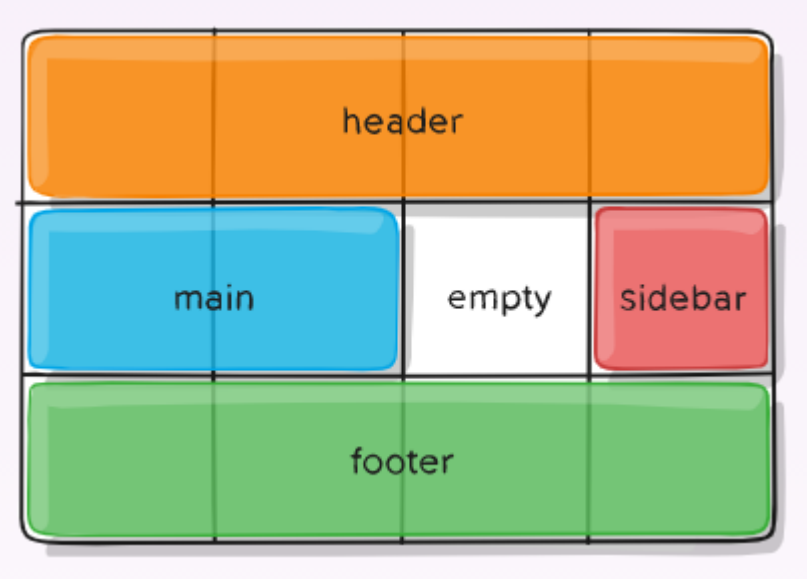
```
.grid-container {
    display: grid;
    grid-template-columns: 50px 1fr 1fr ;
    grid-template-rows: 100px;
    gap: 10px 10px;
}

.grid-container > div{
    background-color: #ccc;
}
```

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

Grid Areas

- Mittels grid-template-areas kann das Grid mittels Namen der verschiedenen Areas angegeben werden
- Mit einem . kann die Zelle freigelassen werden




```
.item-a {
    grid-area: header;
}
.item-b {
    grid-area: main;
}
.item-c {
    grid-area: sidebar;
}
.item-d {
    grid-area: footer;
}
```

```
.container {
   display: grid;
   grid-template-columns: 50px 50px 50px 50px;
   grid-template-rows: auto;
   grid-template-areas:
         "header header header header"
         "main main . sidebar"
         "footer footer footer footer";
}
```

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

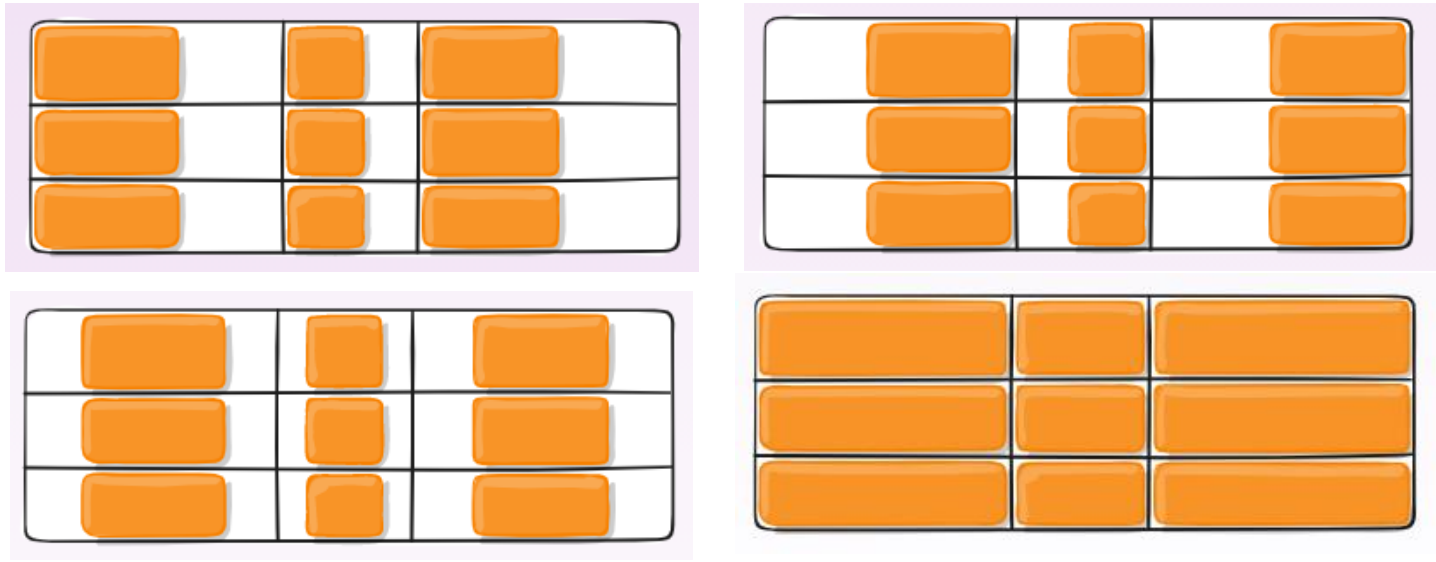
CONTAINER

CODERS.BAY

- Wie bei der Flex-box gibt es die Möglichkeit die Items mit **justify-items** horizontal auszurichten

```
.grid-container {
    justify-items: start | end | center | stretch;
}
```

Default: stretch



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

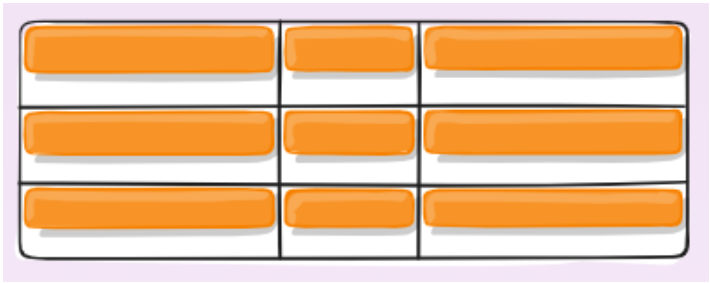
CONTAINER

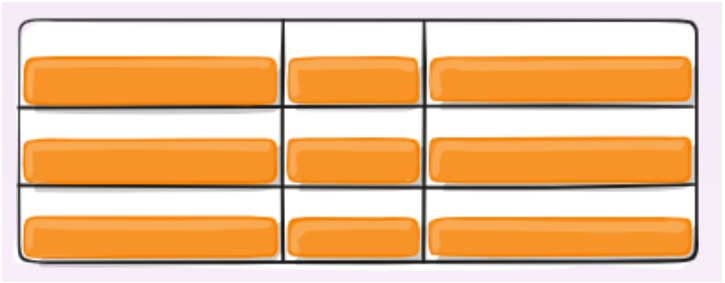
CODERS.BAY

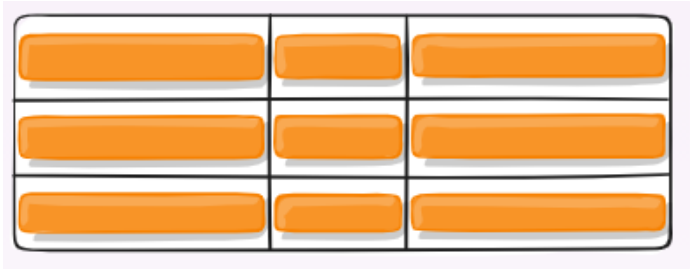
- Wie bei der Flex-box gibt es die Möglichkeit die Items mit **align-items** vertikal auszurichten

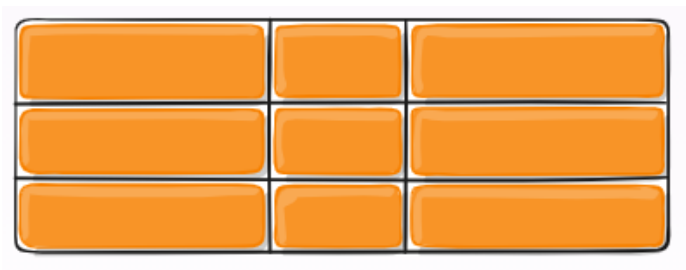
```
.grid-container {
      align-items: start | end | center | stretch;
}
```

Default: stretch









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

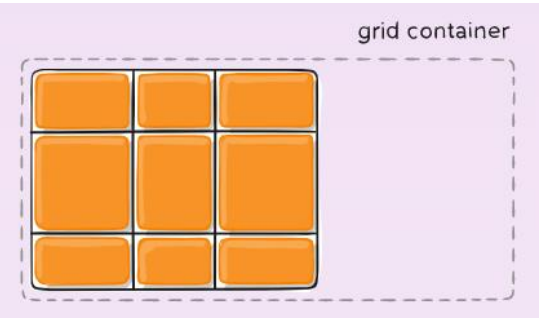
CONTAINER

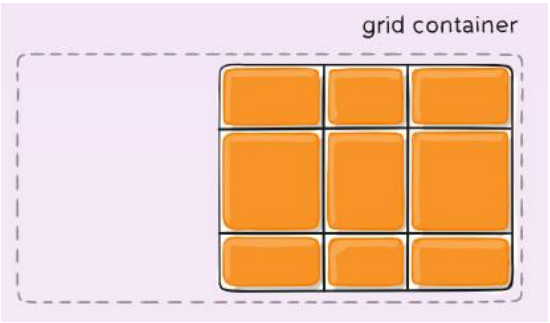
CODERS.BAY

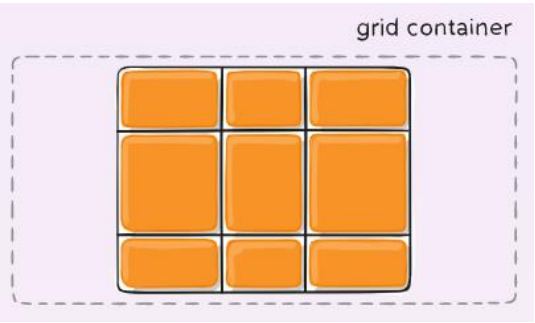
- Wie bei der Flex-box gibt es die Möglichkeit den Content mit **justify-content** horizontal auszurichten

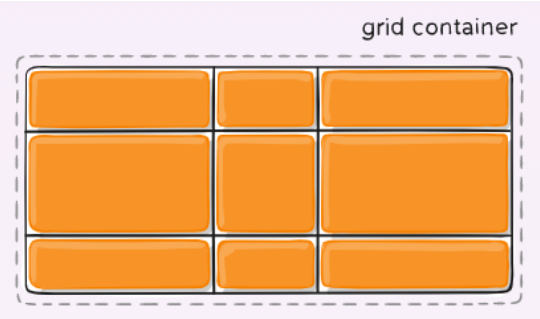
Default: stretch

```
.grid-container {
      justify-content: start | end | center | stretch | space-around | space-between | space-evenly;
}
```









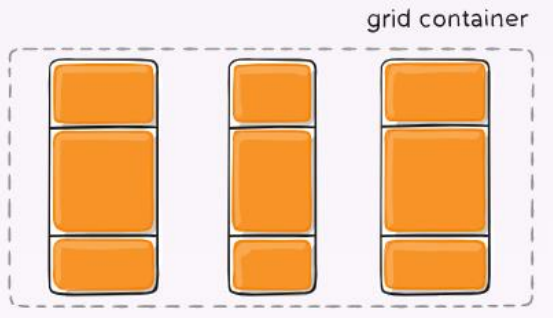
Quelle: https://css-tricks.com/snippets/css/complete-guide-grid/

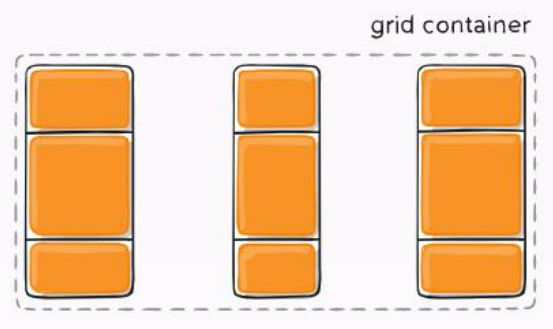
CONTAINER

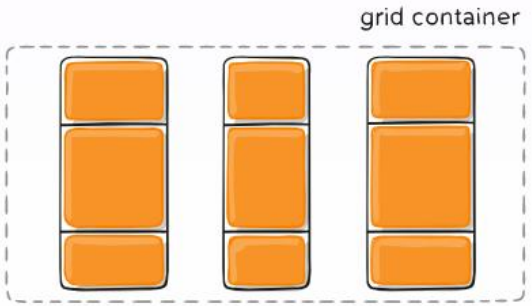
- Wie bei der Flex-box gibt es die Möglichkeit den Content mit **justify-content** horizontal auszurichten

Default: stretch

```
.grid-container {
      justify-content: start | end | center | stretch | space-around | space-between | space-evenly;
}
```







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

ITEM

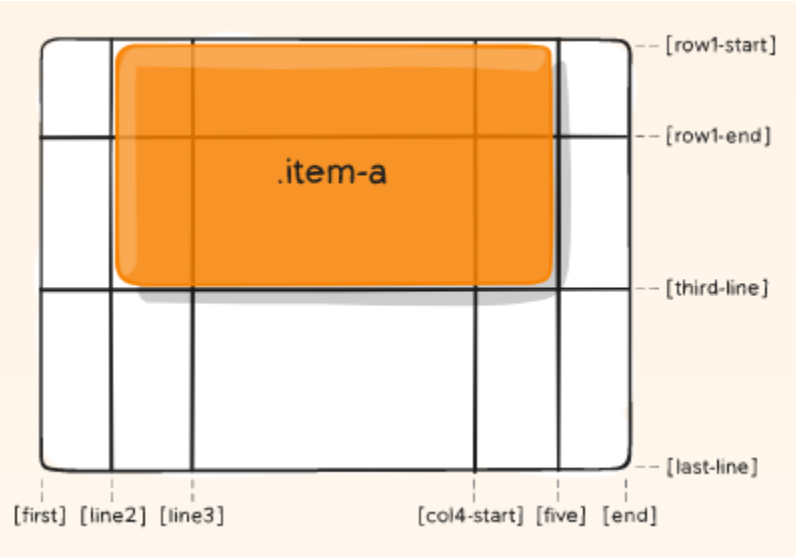
- Float, display: inline-block | table-cell, vertical-align haben keinen Effekt auf Kindelemente des Grids
- Man kann dem Kind explizit sagen wo es zu starten hat und wo es enden soll
  - Das gibt man mit den Line-Names an

```
.item {
   grid-column-start: <number> | <name> | span <number> | span <name> | auto;
   grid-column-end: <number> | <name> | span <number> | span <name> | auto;
   grid-row-start: <number> | <name> | span <number> | span <name> | auto;
   grid-row-end: <number> | <name> | span <number> | span <name> | auto;
}
```

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

Beispiel

```
.item-a {
   grid-column-start: 2;
   grid-column-end: five;
   grid-row-start: row1-start;
   grid-row-end: 3;
}
```



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

ITEM – GRID AREA

CODERS.BAY

- Grid Area kann man dazu benutzen dem Bereich einen Namen zu geben um den Bereich dann in der Template-Area für den Content benutzen zu können
- Oder man kann es für eine kurzschreibweise für grid-row-start + grid-column-start + grid-row-end + grid-column-end

```
.item {
   grid-area : header;
}
```

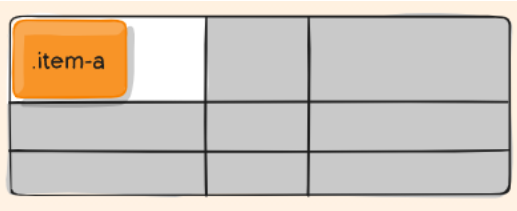
```
.item {
   grid-area : 1 / col4-start / last-line / 6;
}
```

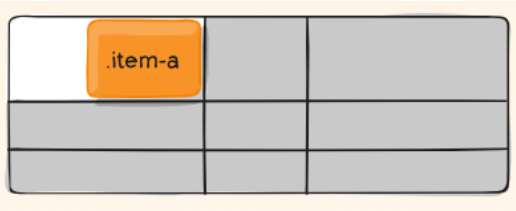
Quelle: https://css-tricks.com/snippets/css/complete-guide-grid/

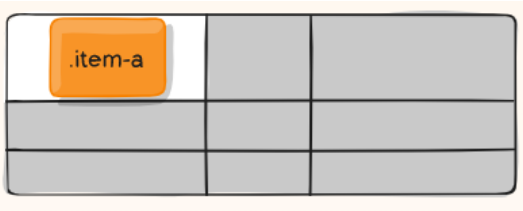
ITEM – JUSTIFY-SELF

- Positioniert das Element innerhalb der Zelle in der horizontalen Achse

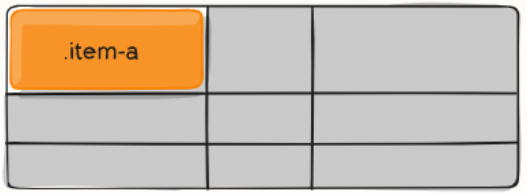
```
.item {
   justify-self: start | end | center | stretch;
}
```







Default: stretch

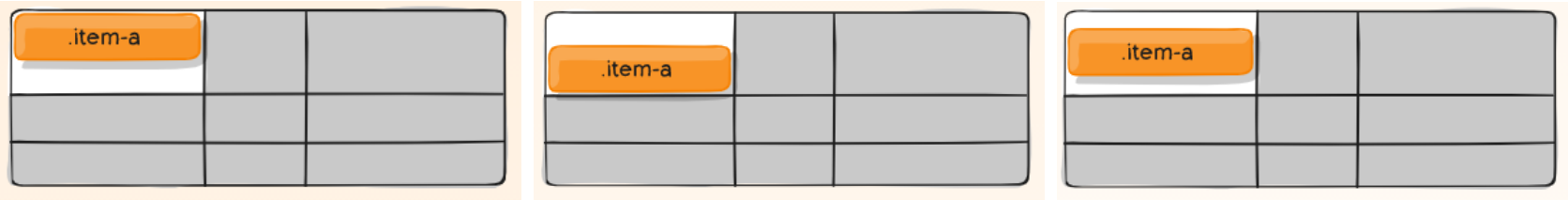


Quelle: https://css-tricks.com/snippets/css/complete-guide-grid/

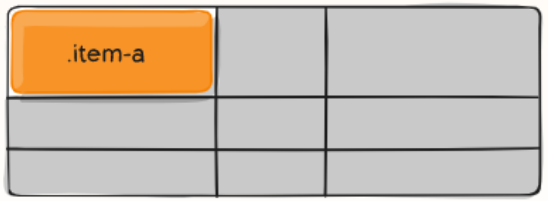
ITEM – ALIGN-SELF

- Positioniert das Element innerhalb der Zelle in der vertikalen Achse

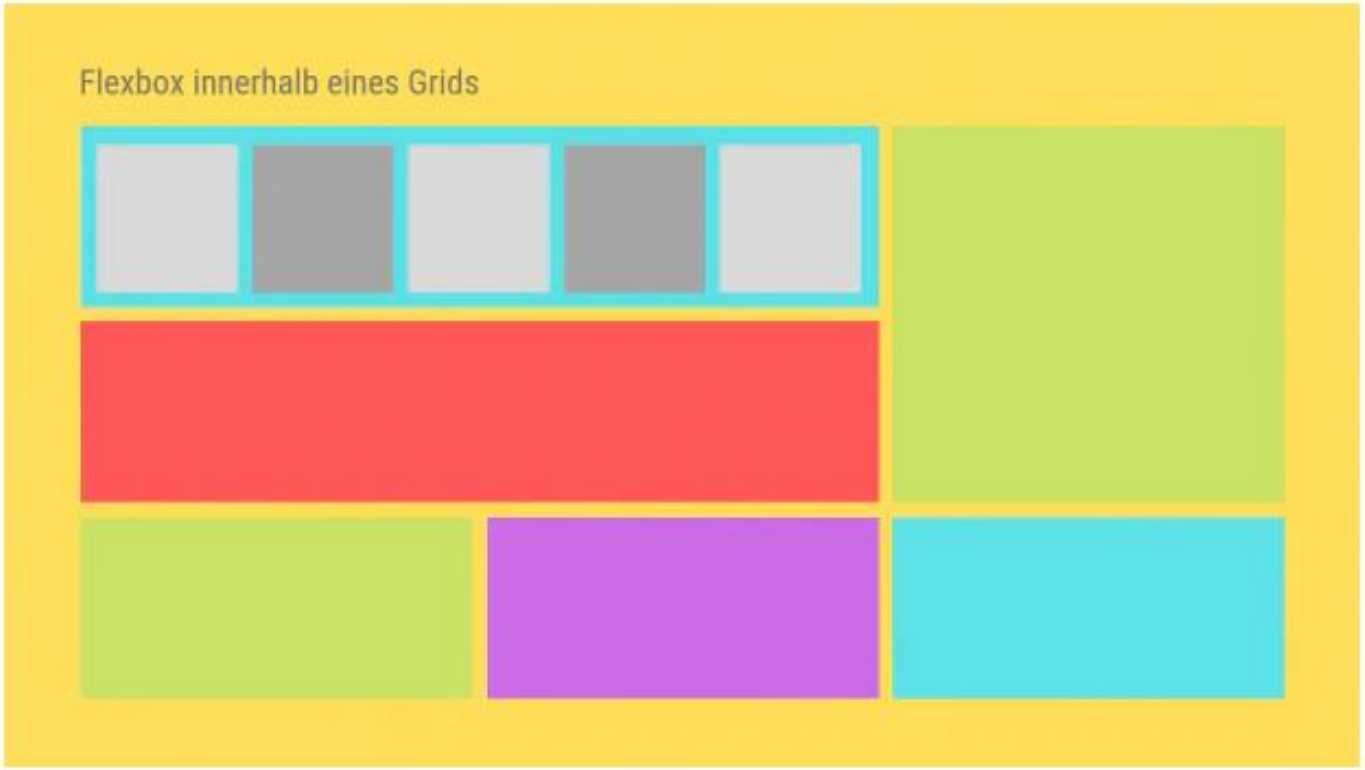
```
.item {
   align-self: start | end | center | stretch;
}
```



Default: stretch



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



Flexbox ist ein eindimensionales Konzept. (Grafik: t3n)

Ende