avionschool

Lesson 3.1 Layouts - Grid

BATCH 4

NOVEMBER 18, 2020

CSS Grid

- Excels at dividing a page into major regions or defining the relationship in terms of size, position, and layer, between parts of a control built from HTML primitives.
- It is a 2-dimensional system, meaning it can handle both columns and rows, unlike flexbox which is largely a 1-dimensional system.

Important Terminology

Grid Container

The element on which `display: grid` is applied.

It’s the direct parent of all the grid items.

```
<div class="container">
  <div class="item item-1"> </div>
  <div class="item item-2"> </div>
  <div class="item item-3"> </div>
</div>
```

Important Terminology

Grid Item

The children (i.e. direct descendants) of the grid container.

Here the `item` elements are grid items, but `sub-item` isn't.

```
<div class="container">
  <div class="item"> </div>
  <div class="item">
    <p class="sub-item"> </p>
  </div>
  <div class="item"> </div>
</div>
```

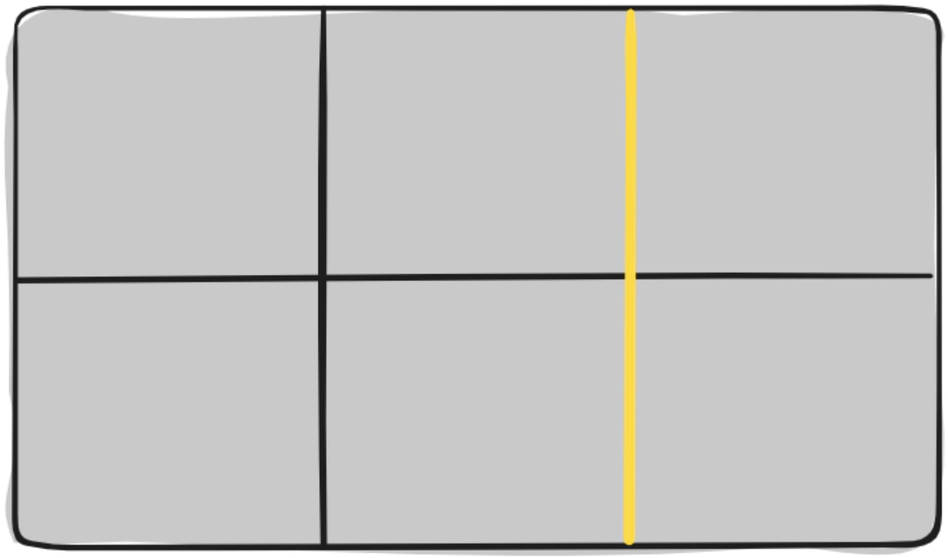
Important Terminology

Grid Line

The dividing lines that make up the structure of the grid.

They can be either vertical (“column grid lines”) or horizontal (“row grid lines”) and reside on either side of a row or column.

Here the yellow line is an example of a column grid line.



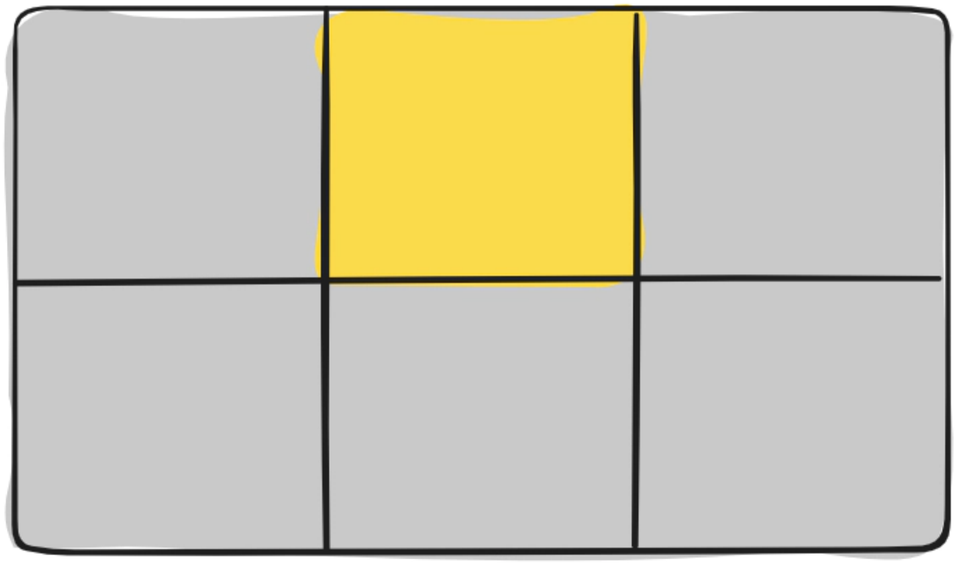
Important Terminology

Grid Cell

The space between two adjacent row and two adjacent column grid lines.

It’s a single “unit” of the grid.

Here’s the grid cell between row grid lines 1 and 2, and column grid lines 2 and 3.



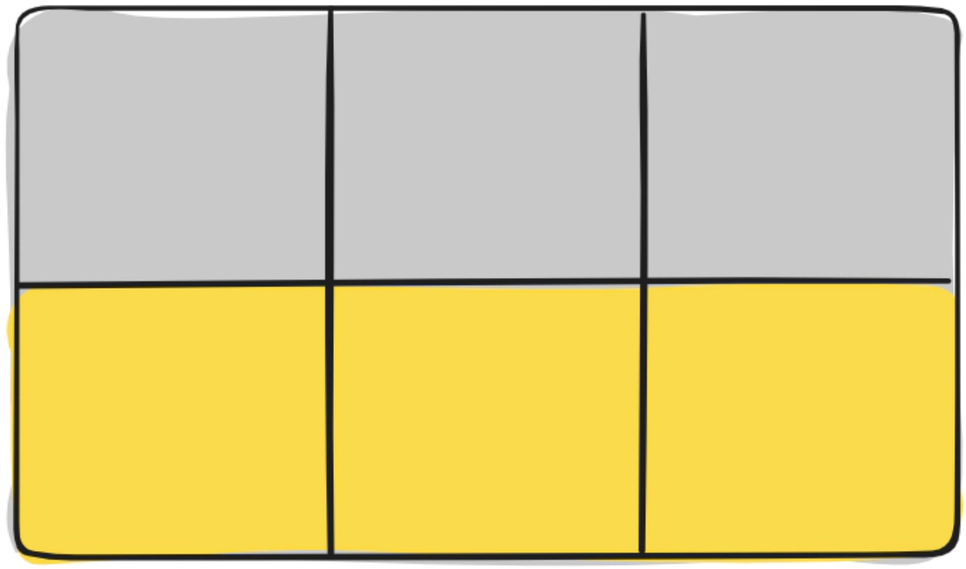
Important Terminology

Grid Track

The space between two adjacent grid lines.

You can think of them like the columns or rows of the grid.

Here’s the grid track between the second and third row grid lines.



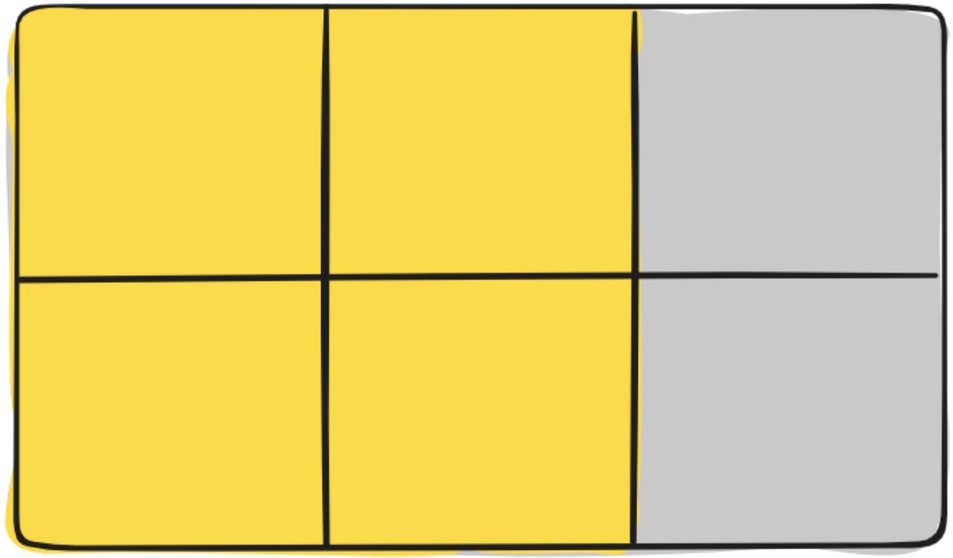
Important Terminology

Grid Area

The total space surrounded by four grid lines.

A grid area may be composed of any number of grid cells.

There’s the grid area between row grid lines 1 and 3, and column grid lines 1 and 3.



Properties for the Parent (Grid Container)

display

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

Properties for the Parent (Grid Container)

grid-template-columns, grid-template-rows

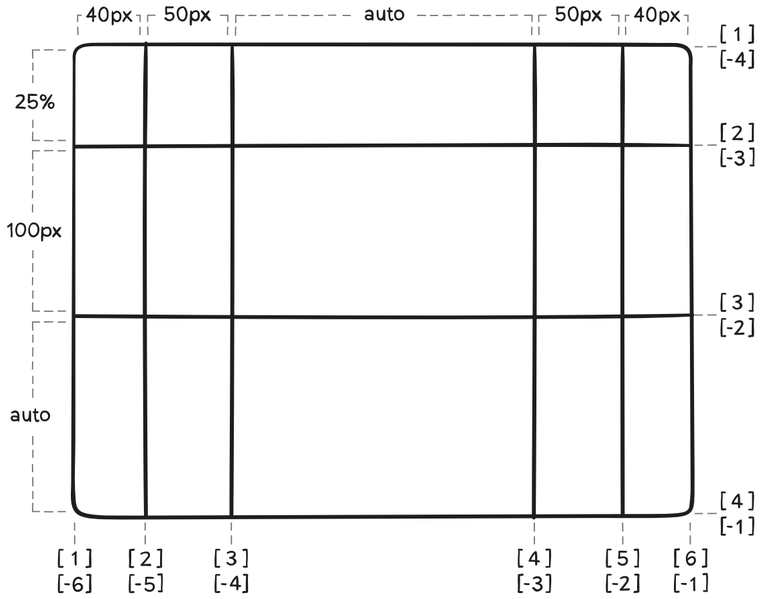
`<track-size>`

- can be a length, a percentage, or a fraction of the free space in the grid (using the `fr` unit)

`<line-name>`

- an arbitrary name of your choosing

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



Properties for the Parent (Grid Container)

grid-template-areas

`<grid-area-name>`

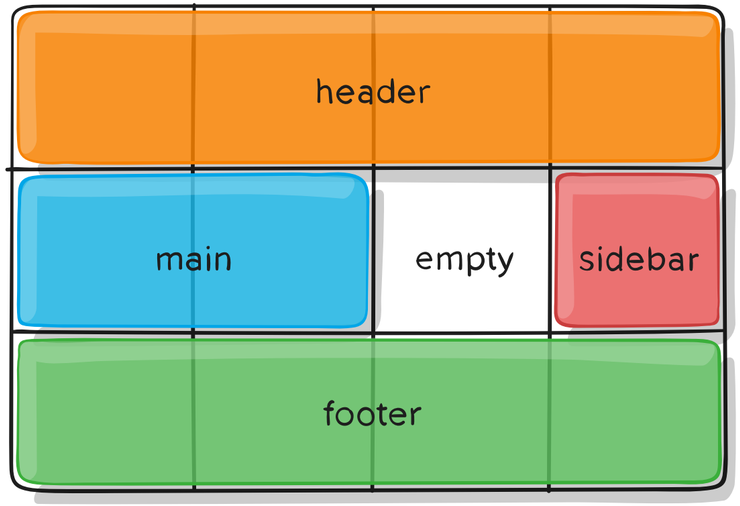
➤ the name of a grid area specified with `grid-area`

`.`

➤ a period signifies an empty grid cell

none

➤ no grid areas are defined




```css
.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";
}
```

Properties for the Parent (Grid Container)

grid-template

none

➤ sets all three properties to their initial values

`<grid-template-rows> / <grid-template-columns>`

➤ sets `grid-template-columns` and `grid-template-rows` to the specified values, respectively, and sets `grid-template-areas` to `none`

```
.container {
  grid-template:
    [row1-start] "header header header" 25px [row1-end]
    [row2-start] "footer footer footer" 25px [row2-end]
    / auto 50px auto;
}
```

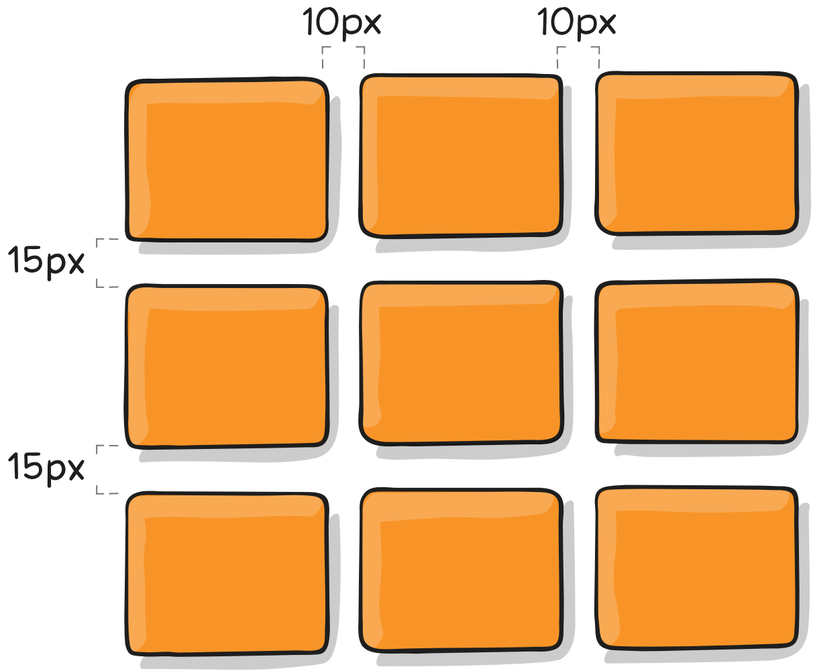
Properties for the Parent (Grid Container)

column-gap , row-gap, grid-column-gap, grid-row-gap

`<line-size>`

➤ a length value

```
.container {
  grid-template-columns: 100px 50px 100px;
  grid-template-rows: 80px auto 80px;
  column-gap: 10px;
  row-gap: 15px;
}
```



Properties for the Parent (Grid Container)

gap, grid-gap

`<grid-row-gap> <grid-column-gap>`

➤ length values

```
.container {
  grid-template-columns: 100px 50px 100px;
  grid-template-rows: 80px auto 80px;
  gap: 15px 10px;
}
```

Properties for the Parent (Grid Container)

justify-items

start

- aligns items to be flush with the start edge of their cell

end

- aligns items to be flush with the end edge of their cell

center

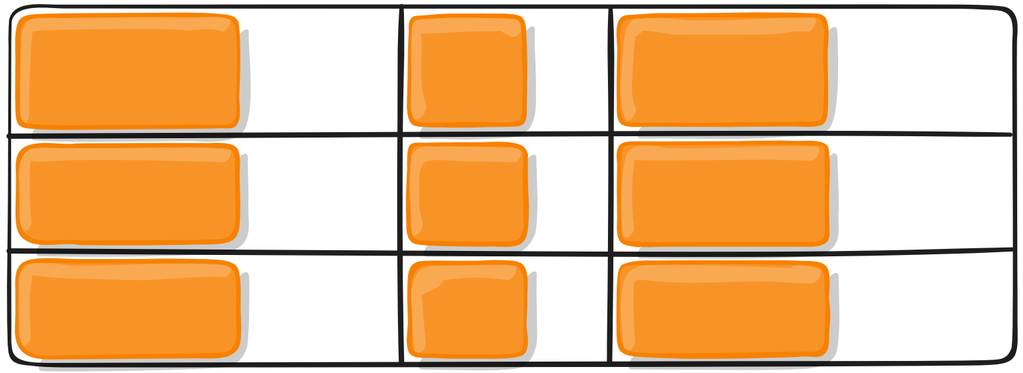
- aligns items in the center of their cell

stretch

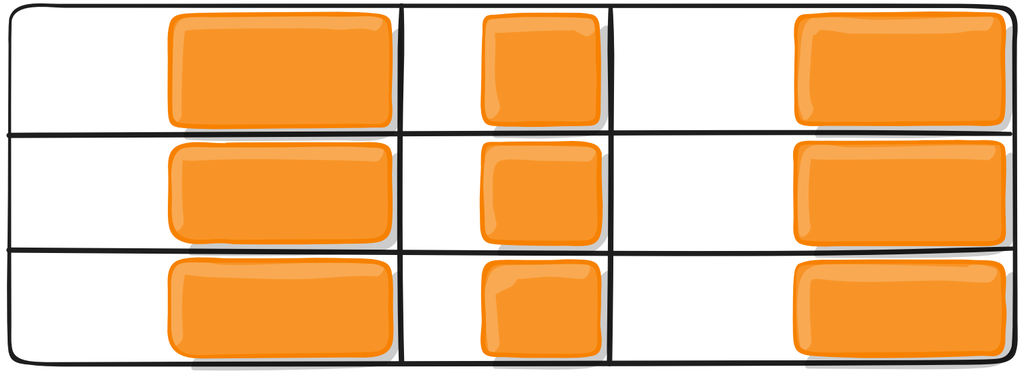
- fills the whole width of the cell (this is the default)

justify-items

```css
.container {
  justify-items: end;
}
```

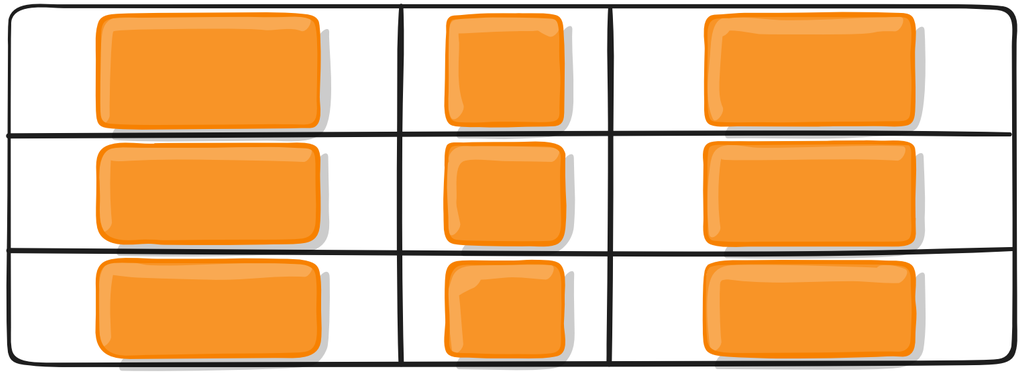


```css
.container {
  justify-items: end;
}
```

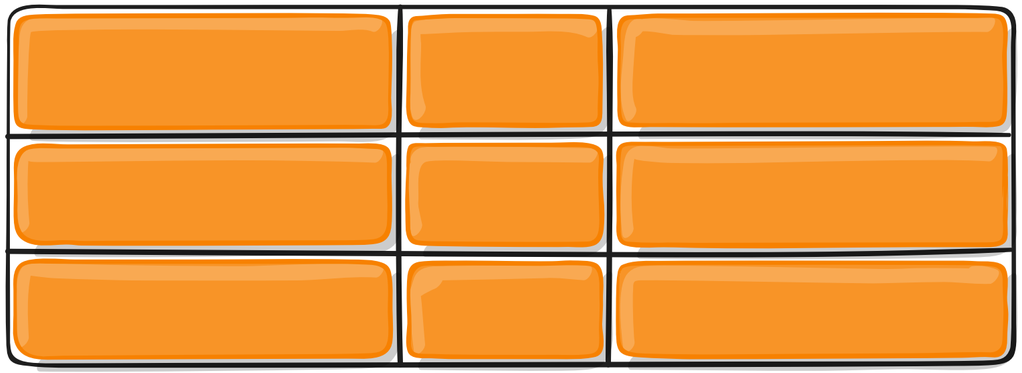


justify-items

```css
.container {
  justify-items: center;
}
```



```css
.container {
  justify-items: stretch;
}
```



Properties for the Parent (Grid Container)

align-items

start

- aligns items to be flush with the start edge of their cell

end

- aligns items to be flush with the end edge of their cell

center

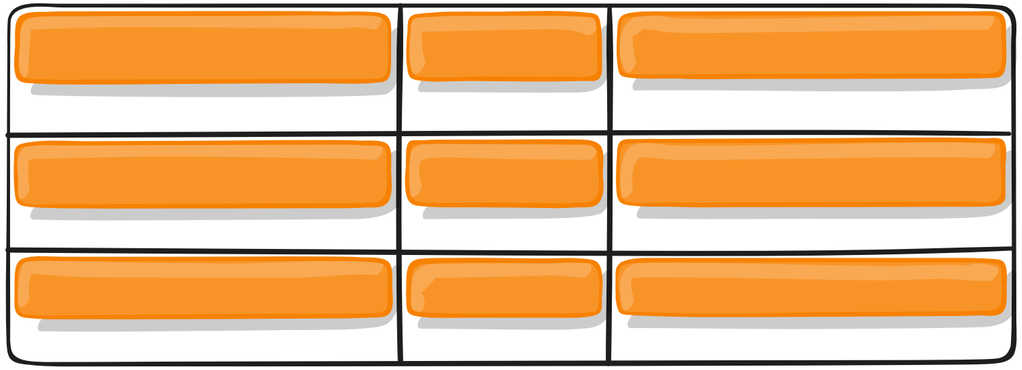
- aligns items in the center of their cell

stretch

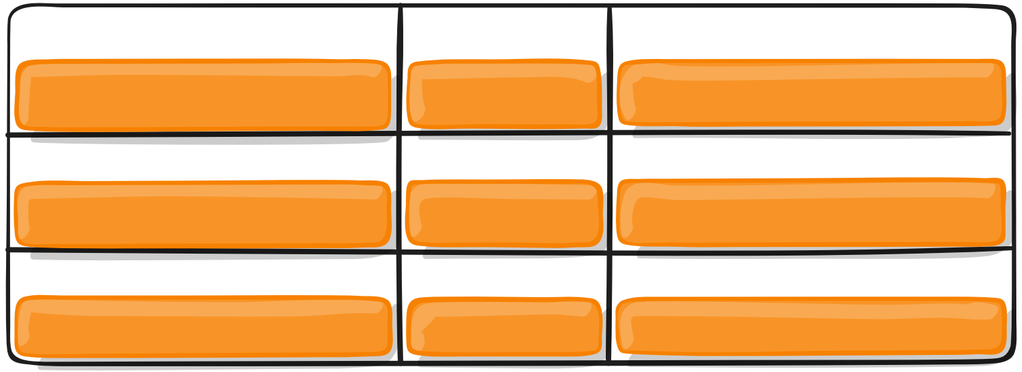
- fills the whole height of the cell (this is the default)

align-items

```css
.container {
  align-items: start;
}
```

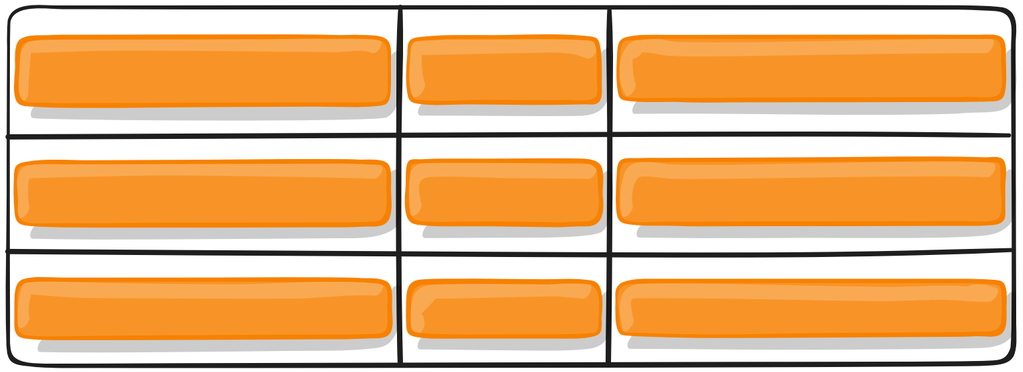


```css
.container {
  align-items: end;
}
```

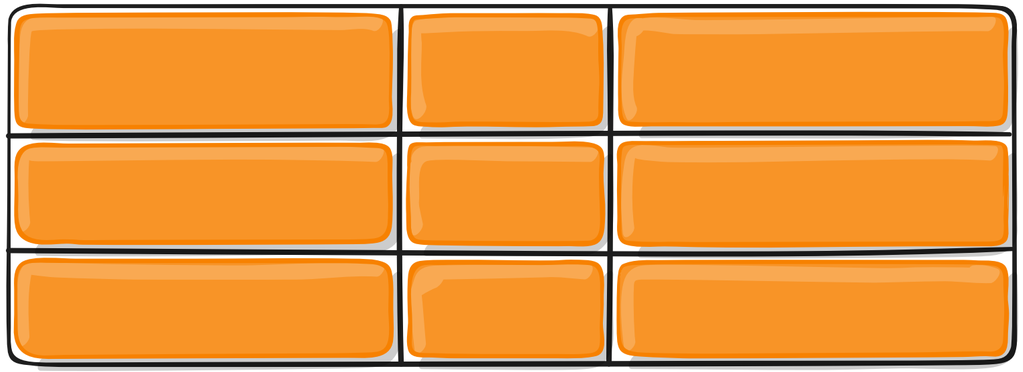


align-items

```css
.container {
  align-items: center;
}
```



```css
.container {
  align-items: stretch;
}
```



Properties for the Parent (Grid Container)

place-items

`<align-items> / <justify-items>`

- The first value sets align-items, the second value justify-items
- If the second value is omitted, the first value is assigned to both properties.

Properties for the Parent (Grid Container)

justify-content

start

➤ aligns the grid to be flush with the start edge of the grid container

end

➤ aligns the grid to be flush with the end edge of the grid container

center

➤ aligns the grid in the center of the grid container

stretch

➤ resizes the grid items to allow the grid to fill the full width of the grid container

space-around

➤ places an even amount of space between each grid item, with half-sized spaces on the far ends

space-between

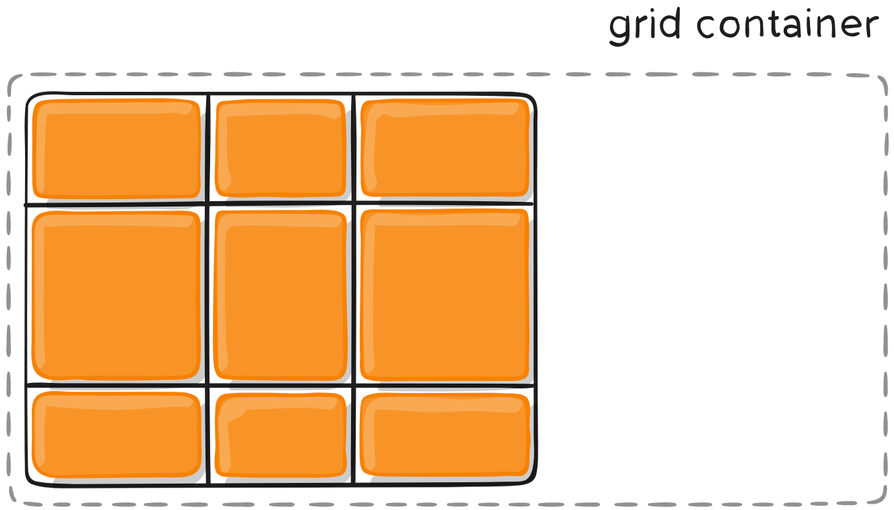
➤ places an even amount of space between each grid item, with no space at the far ends

space-evenly

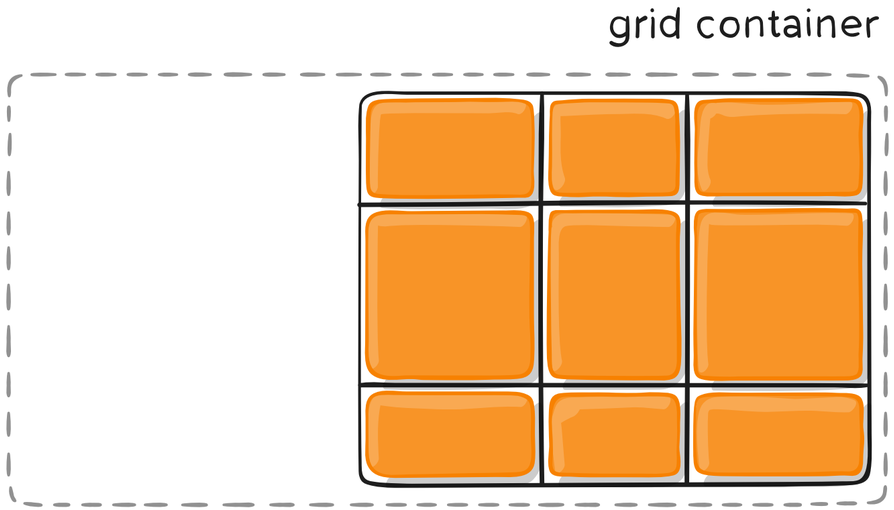
➤ places an even amount of space between each grid item, including the far ends

justify-content

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

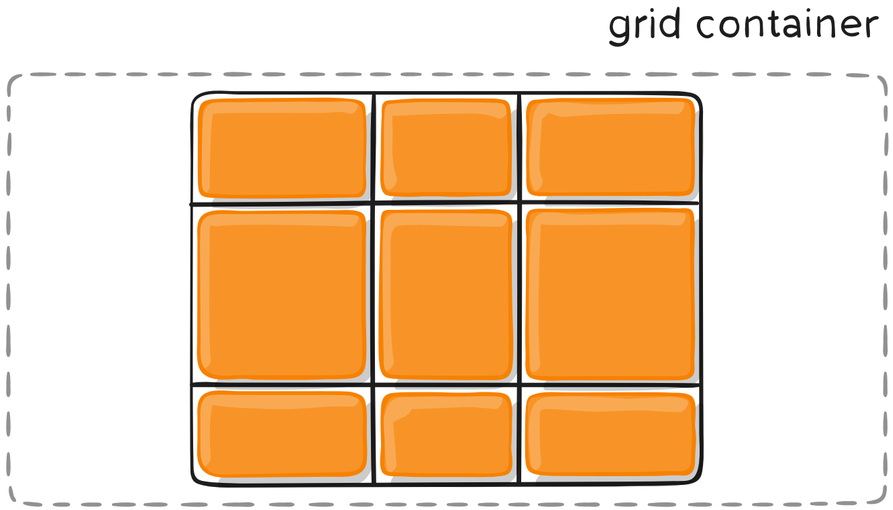



```css
.container {
  justify-content: end;
}
```

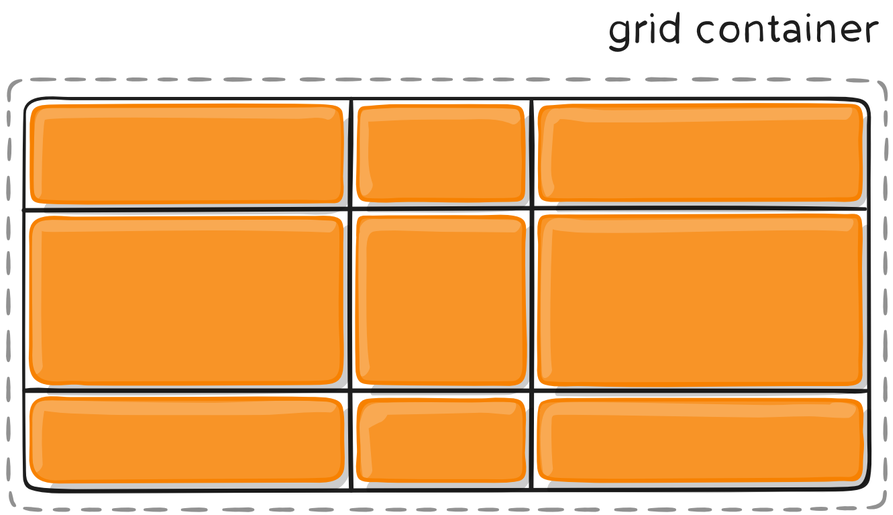


justify-content

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

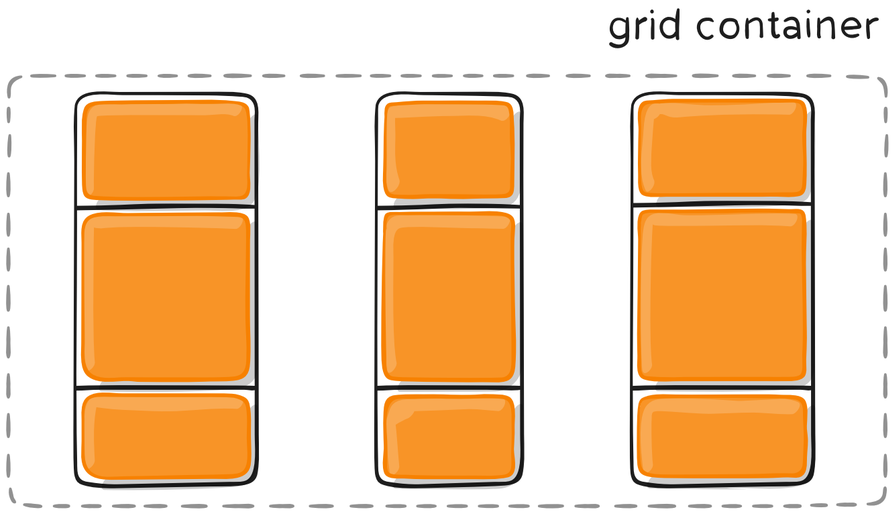



```css
.container {
  justify-content: stretch;
}
```

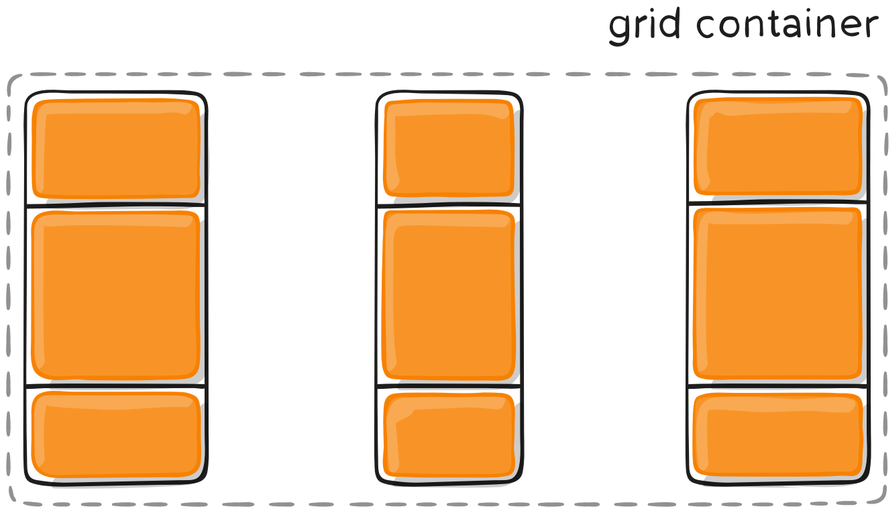


justify-content

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

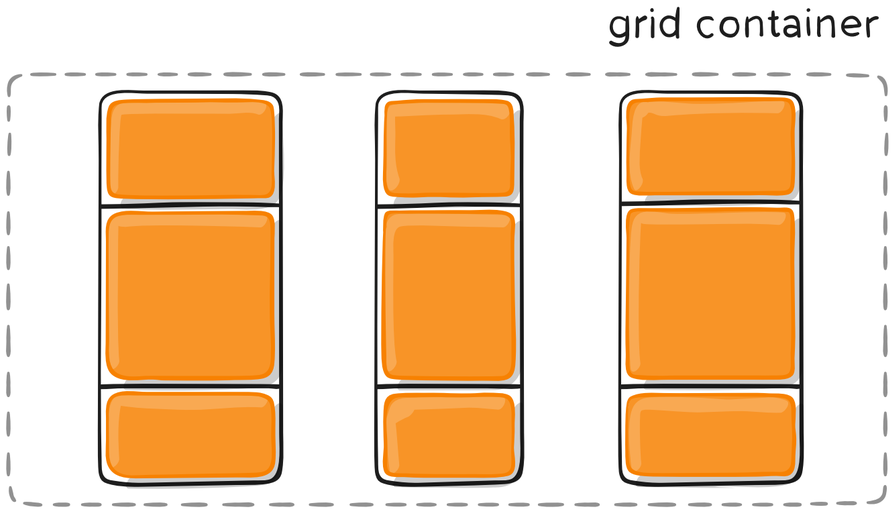



```
.container {
  justify-content: space-between;
}
```



justify-content

```css
.container {
  justify-content: space-evenly;
}
```



Properties for the Parent (Grid Container)

align-content

start

➢ aligns the grid to be flush with the start edge of the grid container

end

➢ aligns the grid to be flush with the end edge of the grid container

center

➢ aligns the grid in the center of the grid container

stretch

➢ resizes the grid items to allow the grid to fill the full height of the grid container

space-around

➢ places an even amount of space between each grid item, with half-sized spaces on the far ends

space-between

➢ places an even amount of space between each grid item, with no space at the far ends

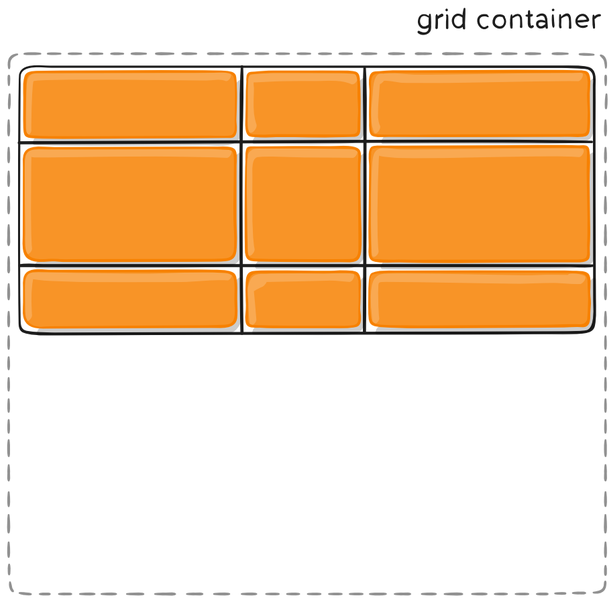
space-evenly

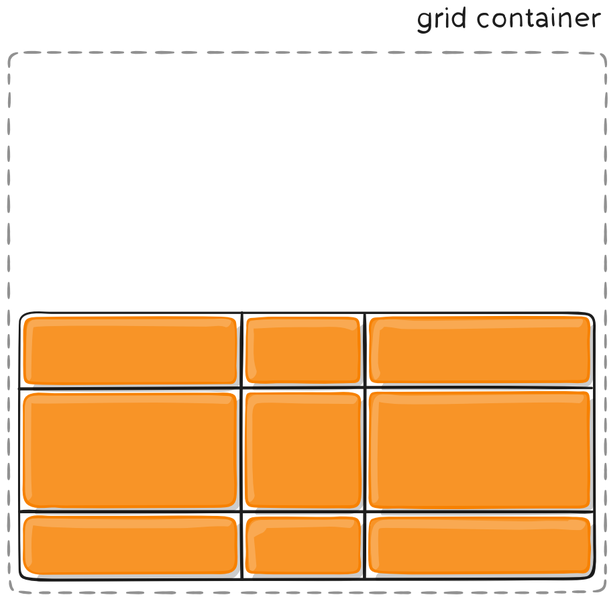
➢ places an even amount of space between each grid item, including the far ends

align-content

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

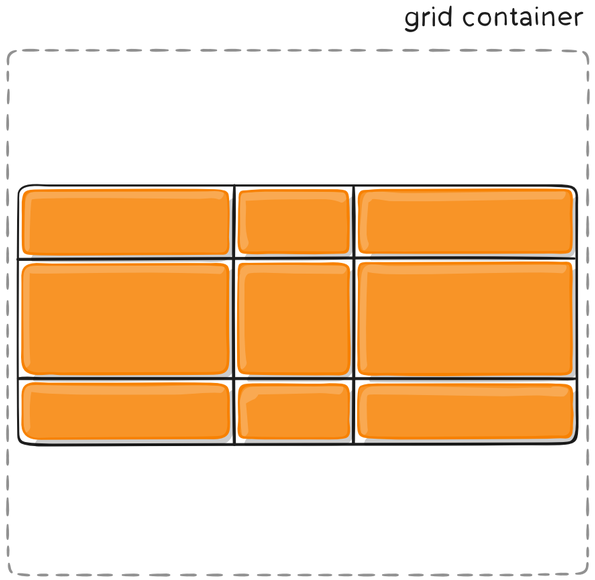
```css
.container {
  align-content: end;
}
```



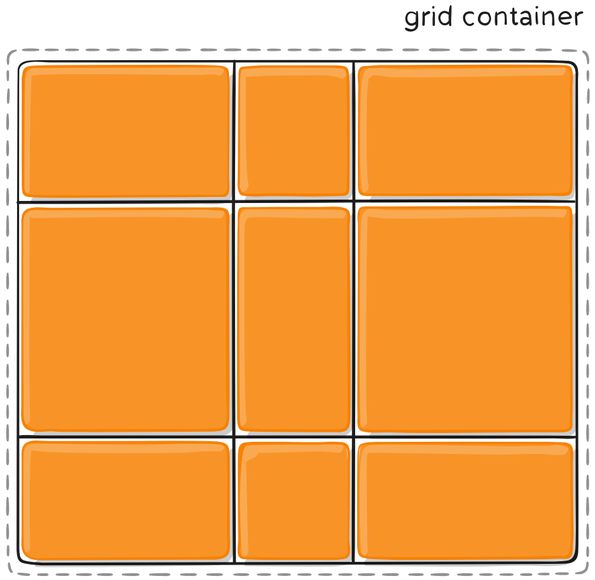


align-content

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

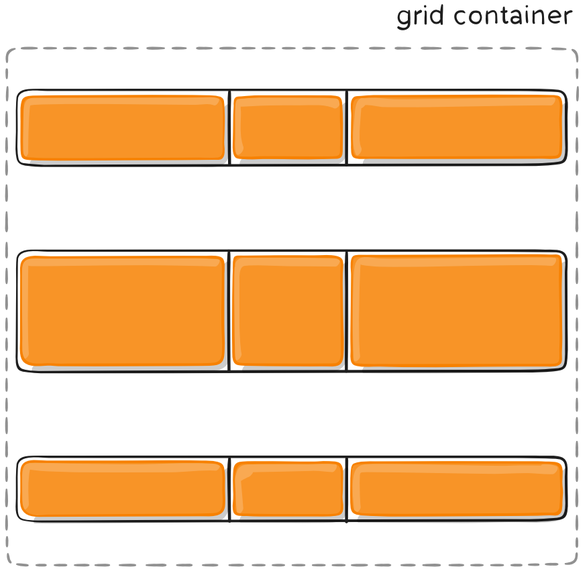



```css
.container {
  align-content: stretch;
}
```

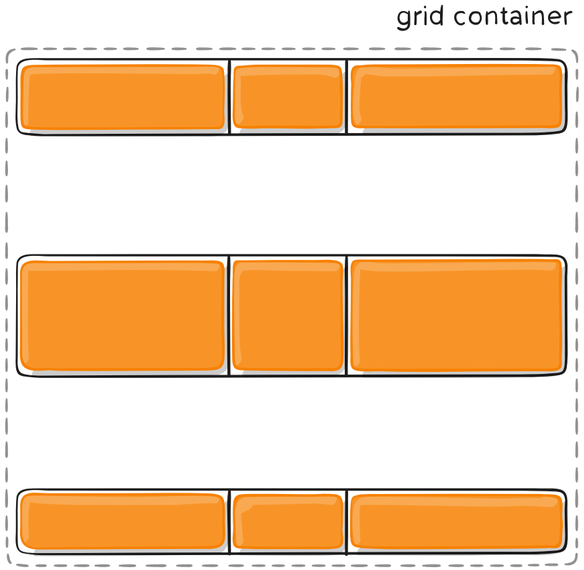


align-content

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

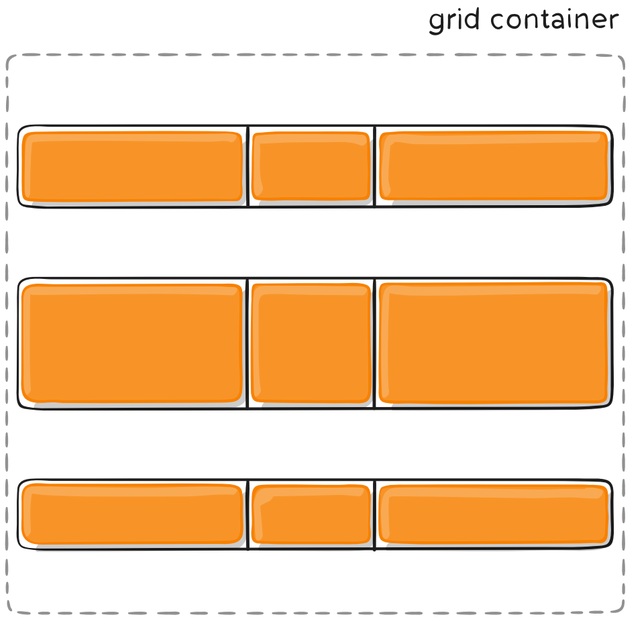



```css
.container {
  align-content: space-between;
}
```



align-content

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



Properties for the Parent (Grid Container)

place-content

`<align-content> / <justify-content>`

- The first value sets `align-content`, the second value `justify-content`.
- f the second value is omitted, the first value is assigned to both properties.

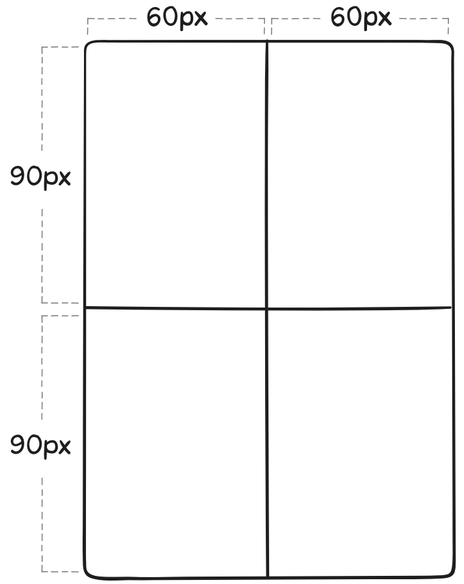
Properties for the Parent (Grid Container)

grid-auto-columns, grid-auto-rows

`<track-size>`

- can be a length, a percentage, or a fraction of the free space in the grid (using the `fr` unit)

```
.container {
  grid-template-columns: 60px 60px;
  grid-template-rows: 90px 90px;
}
```



Properties for the Parent (Grid Container)

grid-auto-flow

row

➤ tells the auto-placement algorithm to fill in each row in turn, adding new rows as necessary (default)

column

➤ tells the auto-placement algorithm to fill in each column in turn, adding new columns as necessary

dense

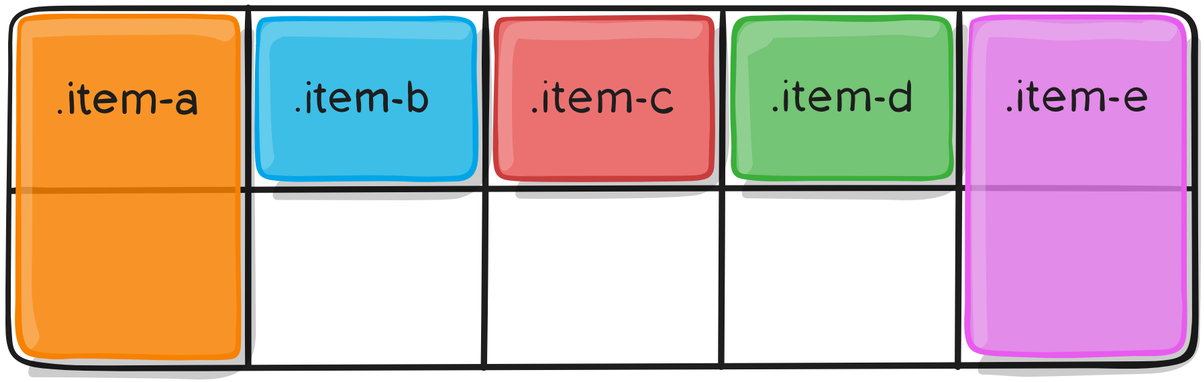
➤ tells the auto-placement algorithm to attempt to fill in holes earlier in the grid if smaller items come up later

grid-auto-flow

```html
<section class="container">
  <div class="item-a">item-a</div>
  <div class="item-b">item-b</div>
  <div class="item-c">item-c</div>
  <div class="item-d">item-d</div>
  <div class="item-e">item-e</div>
</section>
```

```css
.container {
  display: grid;
  grid-template-columns: 60px 60px 60px 60px 60px;
  grid-template-rows: 30px 30px;
  grid-auto-flow: row;
}
```

```css
.item-a {
  grid-column: 1;
  grid-row: 1 / 3;
}
.item-e {
  grid-column: 5;
  grid-row: 1 / 3;
}
```



Properties for the Children(Grid Items)

grid-column-start, grid-column-end, grid-row-start, grid-row-end

`<line>`

- can be a number to refer to a numbered grid line, or a name to refer to a named grid line

`span <number>`

- the item will span across the provided number of grid tracks

`span <name>`

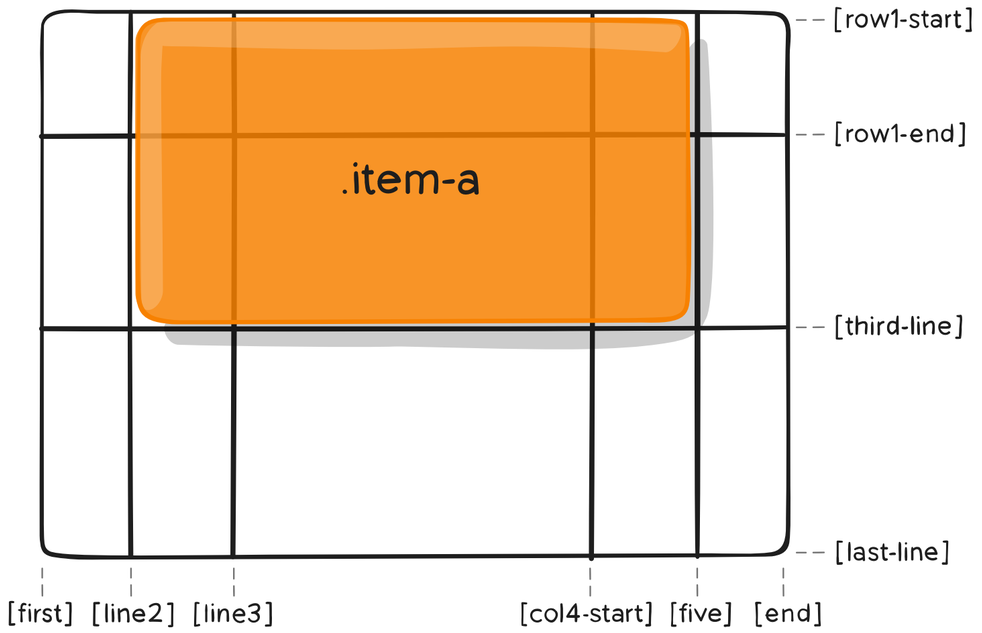
- the item will span across until it hits the next line with the provided name

auto

- indicates auto-placement, an automatic span, or a default span of one

grid-column-start, grid-column-end, grid-row-start, grid-row-end

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



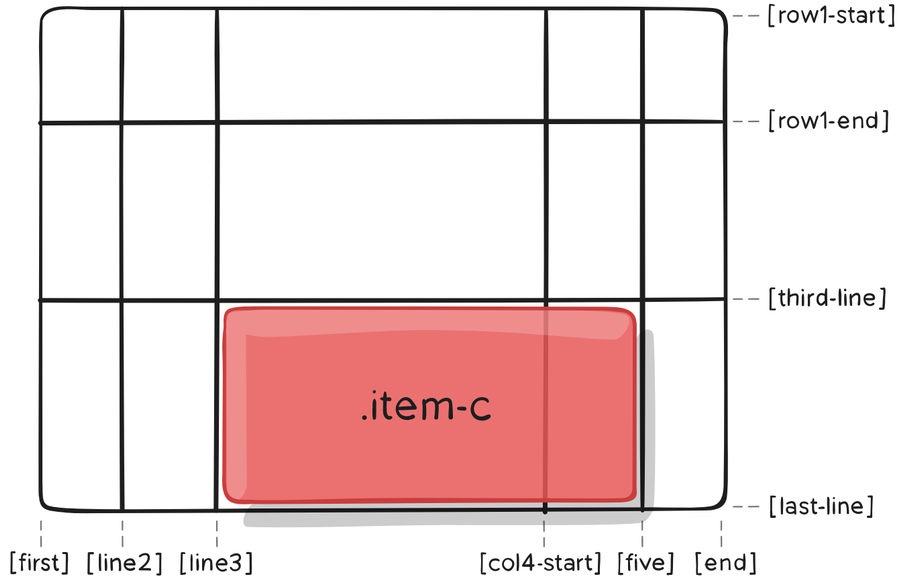
Properties for the Children(Grid Items)

grid-column, grid-row

`<start-line> / <end-line>`

- each one accepts all the same values as the longhand version, including span

```css
.item-c {
  grid-column: 3 / span 2;
  grid-row: third-line / 4;
}
```



Properties for the Children(Grid Items)

grid-area

`<name>`

- a name of your choosing

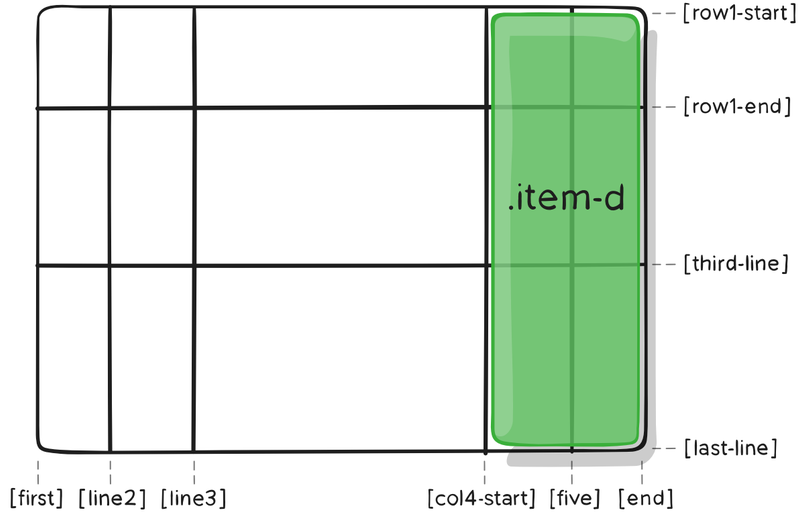
`<row-start> / <column-start> / <row-end> / <column-end>`

- can be numbers or named lines

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

As the short-shorthand for grid-row-start + grid-column-start + grid-row-end + grid-column-end

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



Properties for the Children(Grid Items)

justify-self

start

- aligns the grid item to be flush with the start edge of the cell

end

- aligns the grid item to be flush with the end edge of the cell

center

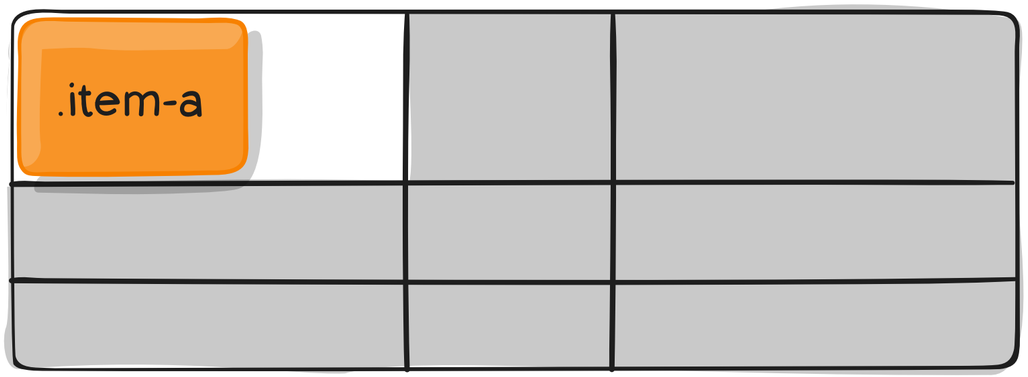
- aligns the grid item in the center of the cell

stretch

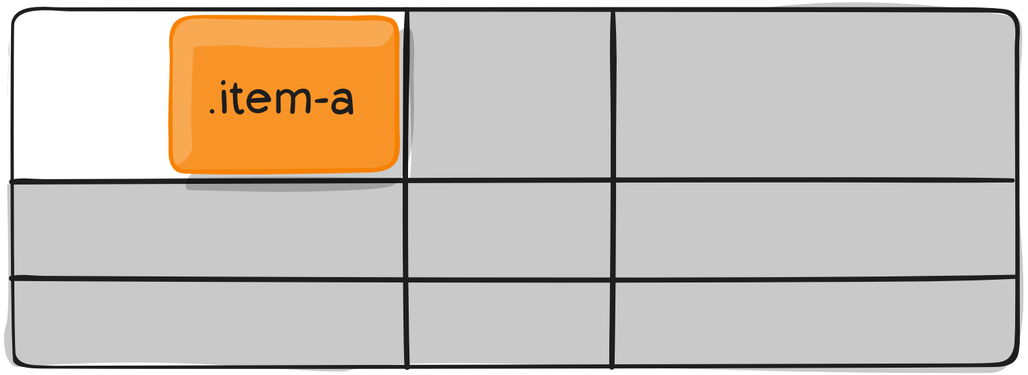
- fills the whole width of the cell (this is the default)

justify-self

```css
.item-a {
  justify-self: start;
}
```

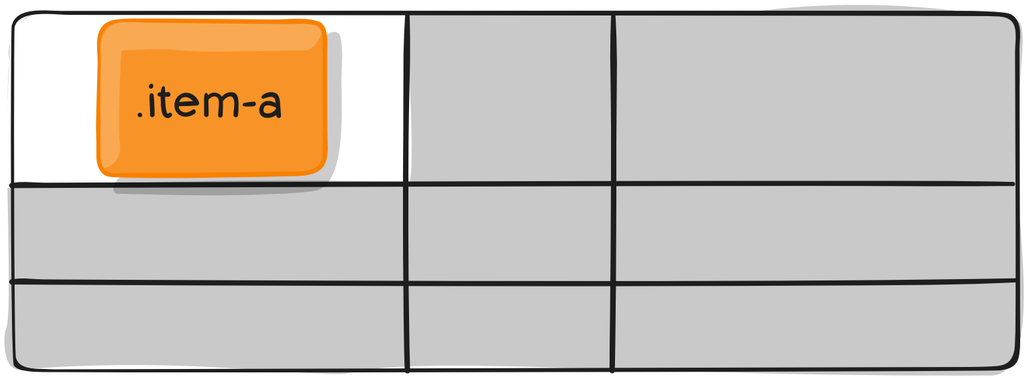



```css
.item-a {
  justify-self: end;
}
```

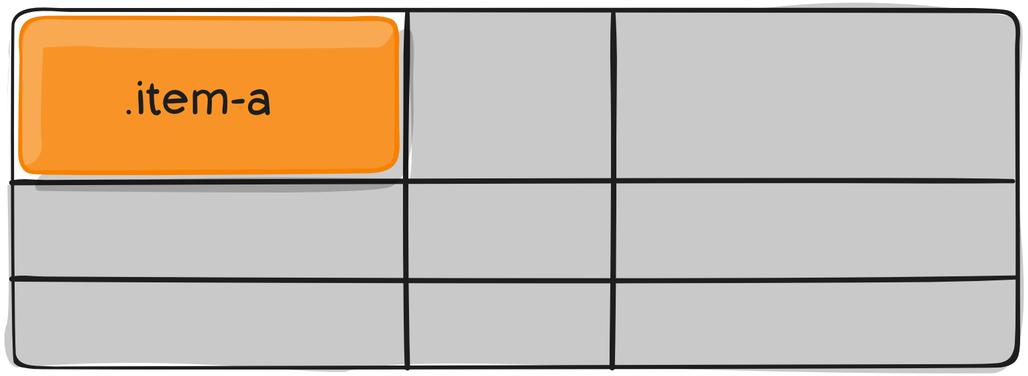


justify-self

```css
.item-a {
  justify-self: center;
}
```




```css
.item-a {
  justify-self: stretch;
}
```



Properties for the Children(Grid Items)

align-self

start

- aligns the grid item to be flush with the start edge of the cell

end

- aligns the grid item to be flush with the end edge of the cell

center

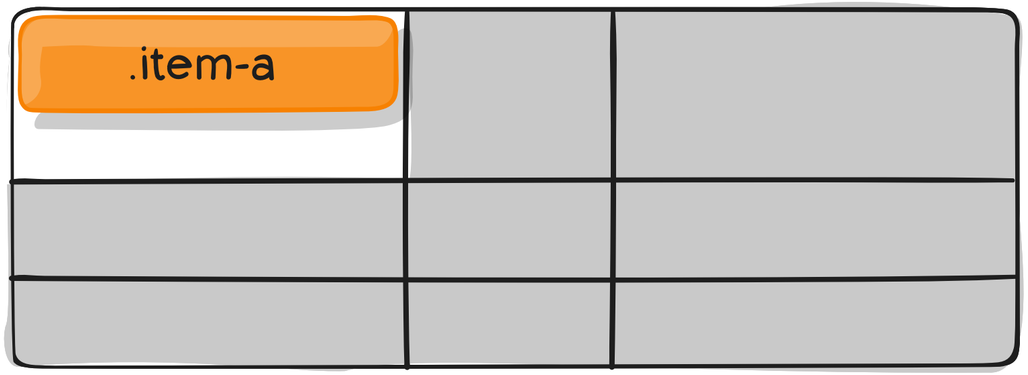
- aligns the grid item in the center of the cell

stretch

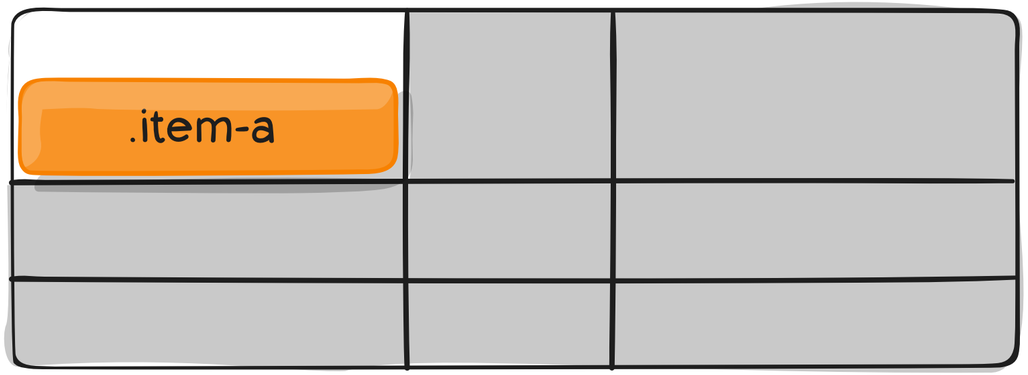
- fills the whole height of the cell (this is the default)

align-self

```css
.item-a {
  align-self: start;
}
```

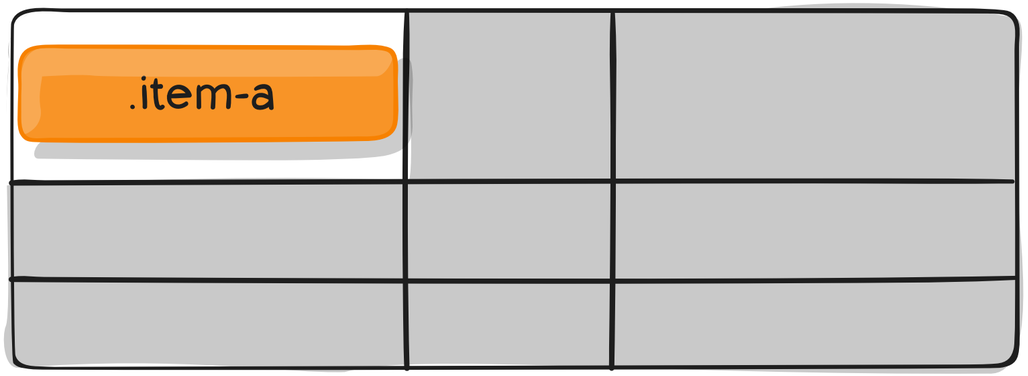



```css
.item-a {
  align-self: end;
}
```

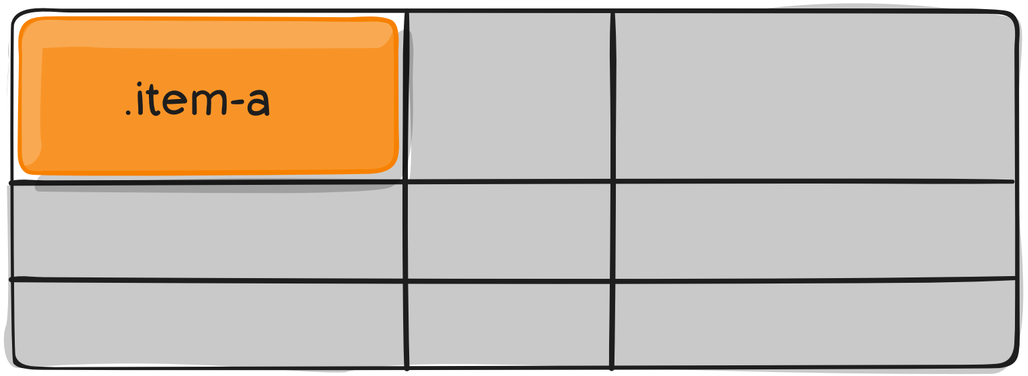


align-self

```css
.item-a {
  align-self: center;
}
```




```css
.item-a {
  align-self: stretch;
}
```



Properties for the Children(Grid Items)

place-self

auto

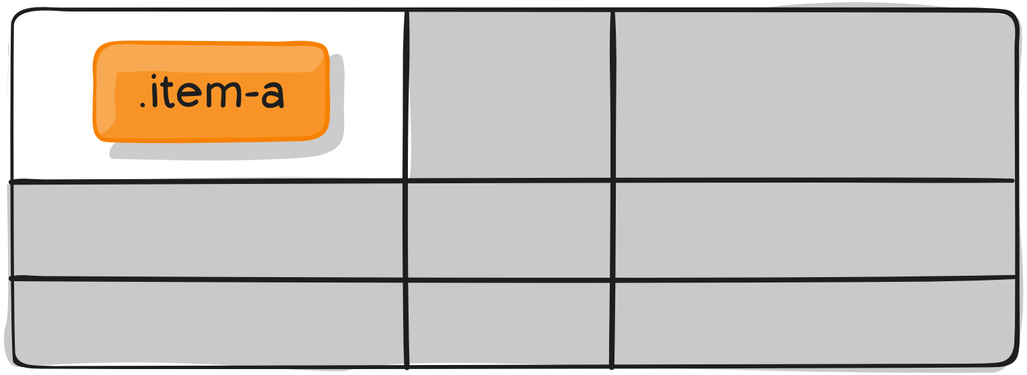
➤ The “default” alignment for the layout mode.

`<align-self> / <justify-self>`

➤ The first value sets `align-self`, the second value `justify-self`.

➤ If the second value is omitted, the first value is assigned to both properties

```css
.item-a {
  place-self: center;
}
```



CSS Grid

grid

none

- sets all sub-properties to their initial values.

`<grid-template>`

- works the same as the `grid-template` shorthand.

`<grid-template-rows> / [ auto-flow && dense? ] <grid-auto-columns>?`

- sets `grid-template-rows` to the specified value.
- If the `auto-flow` keyword is to the right of the slash, it sets `grid-auto-flow` to `column`.
- If the `dense` keyword is specified additionally, the auto-placement algorithm uses a “dense” packing algorithm.
- If `grid-auto-columns` is omitted, it is set to `auto`.

`[ auto-flow && dense? ] <grid-auto-rows>? / <grid-template-columns>`

- sets `grid-template-columns` to the specified value.
- If the `auto-flow` keyword is to the left of the slash, it sets `grid-auto-flow` to `row`.
- If the `dense` keyword is specified additionally, the auto-placement algorithm uses a “dense” packing algorithm.
- If `grid-auto-rows` is omitted, it is set to `auto`.

CSS Grid

```css
.container {
  grid: 100px 300px / 3fr 1fr;
}

.container {
  grid-template-rows: 100px 300px;
  grid-template-columns: 3fr 1fr;
}
```

=

```css
.container {
  grid: auto-flow / 200px 1fr;
}

.container {
  grid-auto-flow: row;
  grid-template-columns: 200px 1fr;
}
```

Activity

Make a responsive chessboard that is currently in a middle game using grid!