UI BASICS

LAYOUT - Grid

CONTENT

Grid Container

display: grid | inline-grid;

grid-template-columns / grid-template-rows

- Explicit
- minmax()
- repeat()

grid-column-gap / column-gap
grid-row-gap / row-gap
grid-gap / gap

grid-template-areas

- Area names

justify-content
align-content
place-content

- Tracks

justify-items
align-items
place-items

- All Items

Grid Items

grid-column-start / grid-column-end
grid-row-start / grid-row-end
----
grid-column
grid-row

- Lines

grid-area

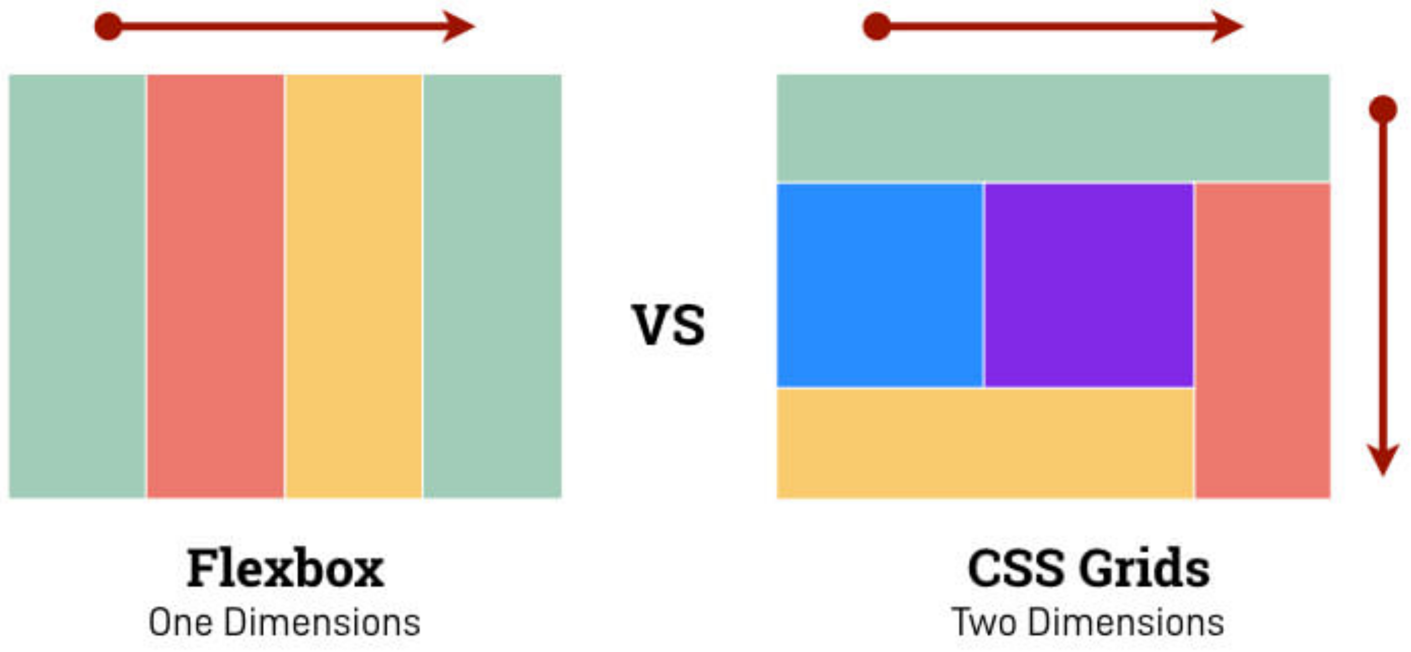
- Area names

justify-self
align-self
place-self

- Individual Item

CSS Grid Layout

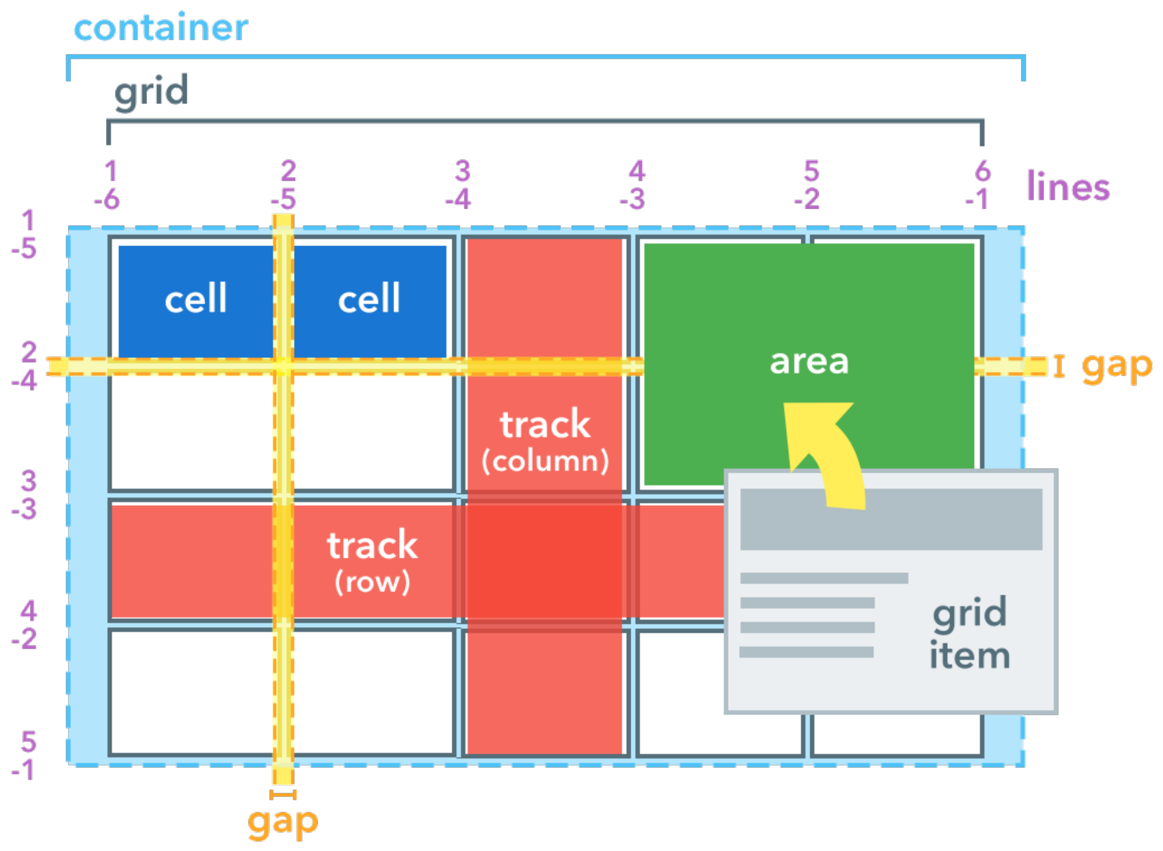
The **CSS Grid Layout** offers **a grid-based layout system**, **with rows and columns,** to create easier and design web pages.



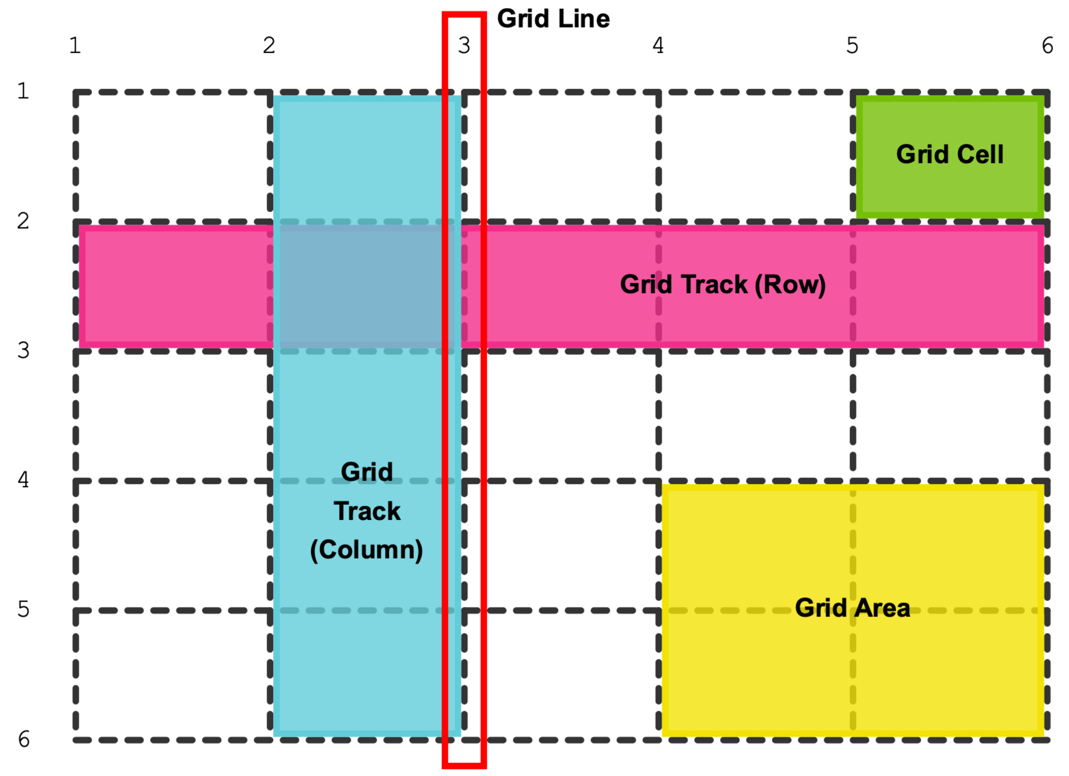
Content-first

Layout-first

CSS Grid Terminology



- Grid ***container***
- Grid ***item***
- Grid ***line***
- Grid ***cell***
- Grid ***track***
- Grid ***area***
- Grid ***gap***



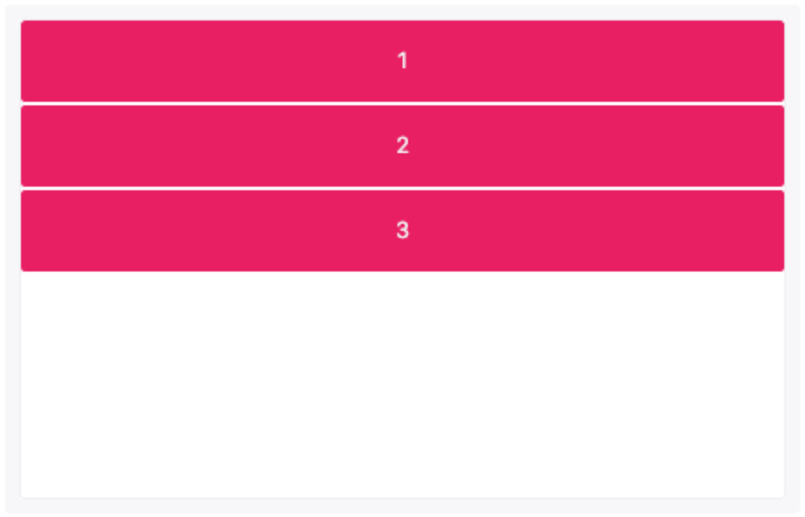
Grid Container - display

Defines the element **as a grid container** and establishes a new grid formatting context for its contents.

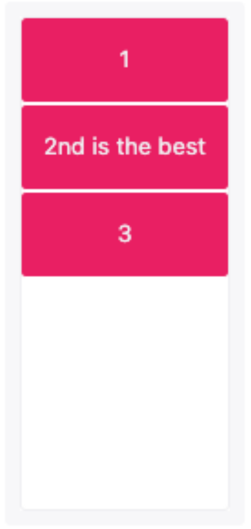
```
.container {
  display: grid | inline-grid;
}
```

- **grid** – generates a block-level grid
- **inline-grid** – generates an inline-level grid

display: grid;



display: inline-grid;



CONTENT

Grid Container

grid-template-columns / grid-template-rows

- Explicit
- minmax()
- repeat()

Grid Items

grid-column-start / grid-column-end
grid-row-start / grid-row-end
----
grid-column
grid-row

- Lines

LAYOUT

Grid Container - grid-template-columns grid-template-rows

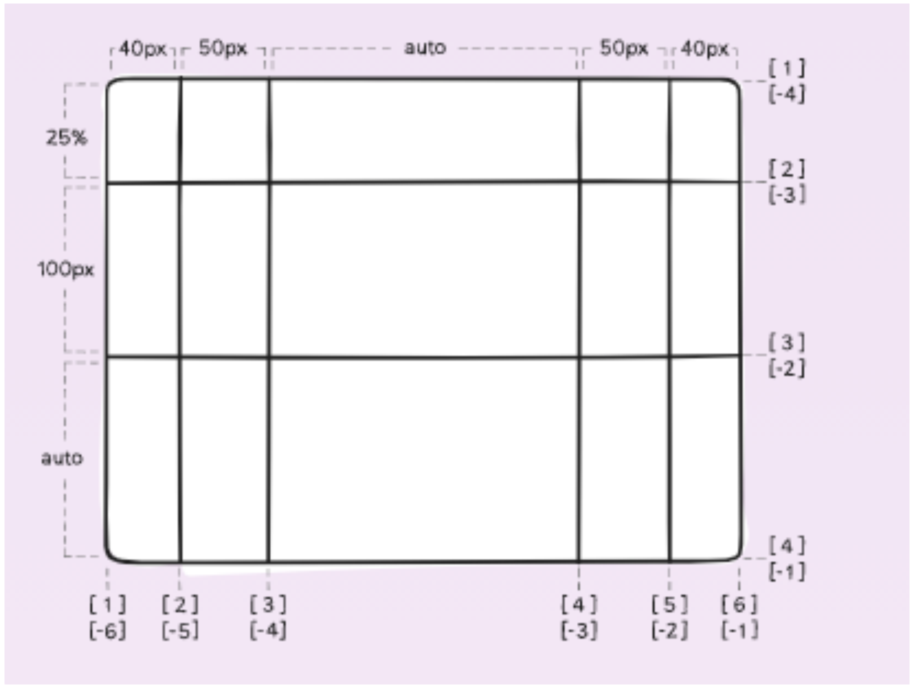
- Explicit Grid Tracks
- Min and Max Grid Tracks
- Repeating Grid Tracks

- Defines **the columns** and **rows** of the grid **with a space-separated list of values**.
- The values represent the **track size**, and the space between them represents the **grid line**.

$\underbrace{\quad\quad\quad}$

- length (px, rem, em)
- %
- fraction (fr) / auto

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



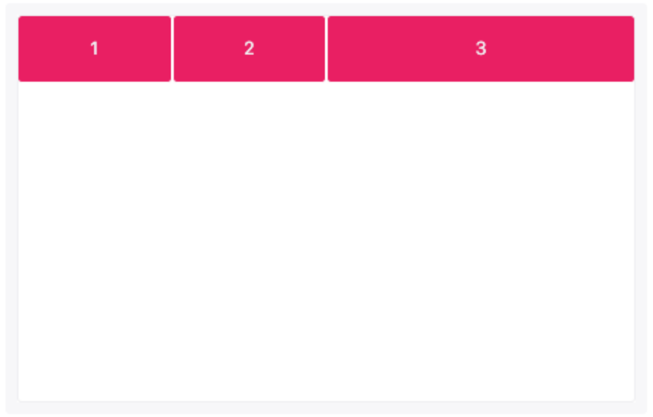
fr : The **fr** unit helps create flexible grid tracks.

LAYOUT

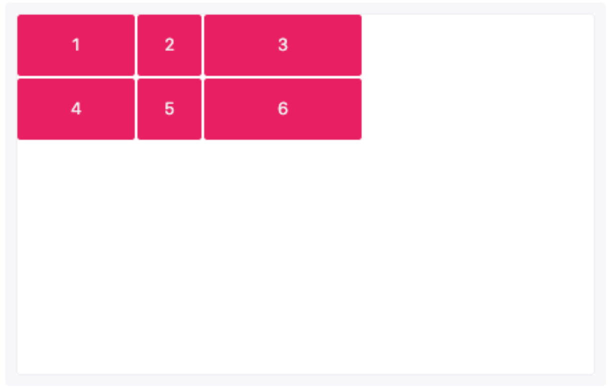
Grid Container - grid-template-columns grid-template-rows

- Explicit Grid Tracks
- Min and Max Grid Tracks
- Repeating Grid Tracks

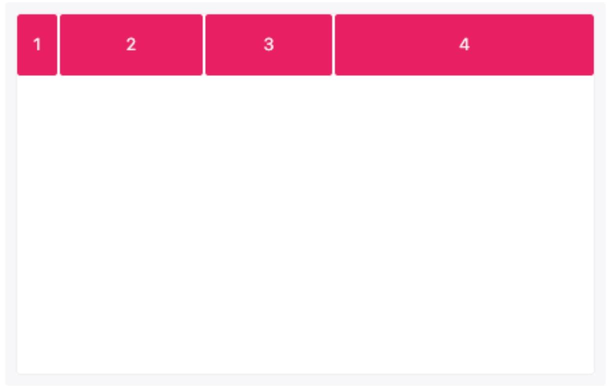
`grid-template-columns: 1fr 1fr 2fr`



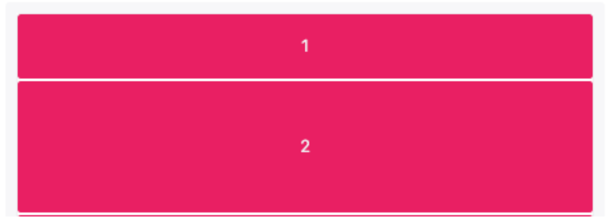
`grid-template-columns: 90px 50px 120px`



`grid-template-columns: 3rem 25% 1fr 2fr`



`grid-template-rows: 50px 100px`



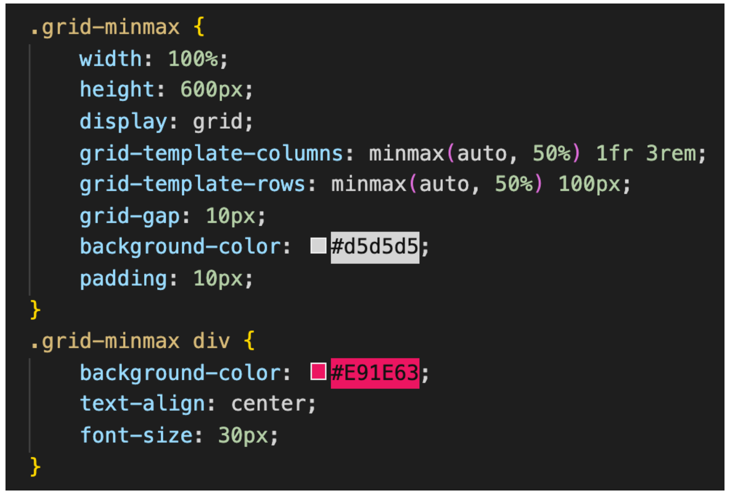
- **fr** is calculated **based on the remaining space** when combined with other length values.
- In this example, 3rem and 25% would be subtracted from the available space before the size of fr is calculated:

1fr = (width of grid - 3rem - 25% of width of grid) / 3

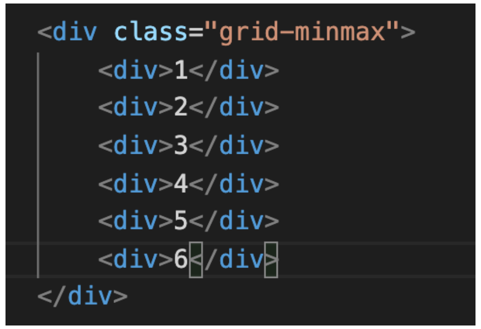
Grid Container - grid-template-columns grid-template-rows

- Explicit Grid Tracks
- Min and Max Grid Tracks
- Repeating Grid Tracks

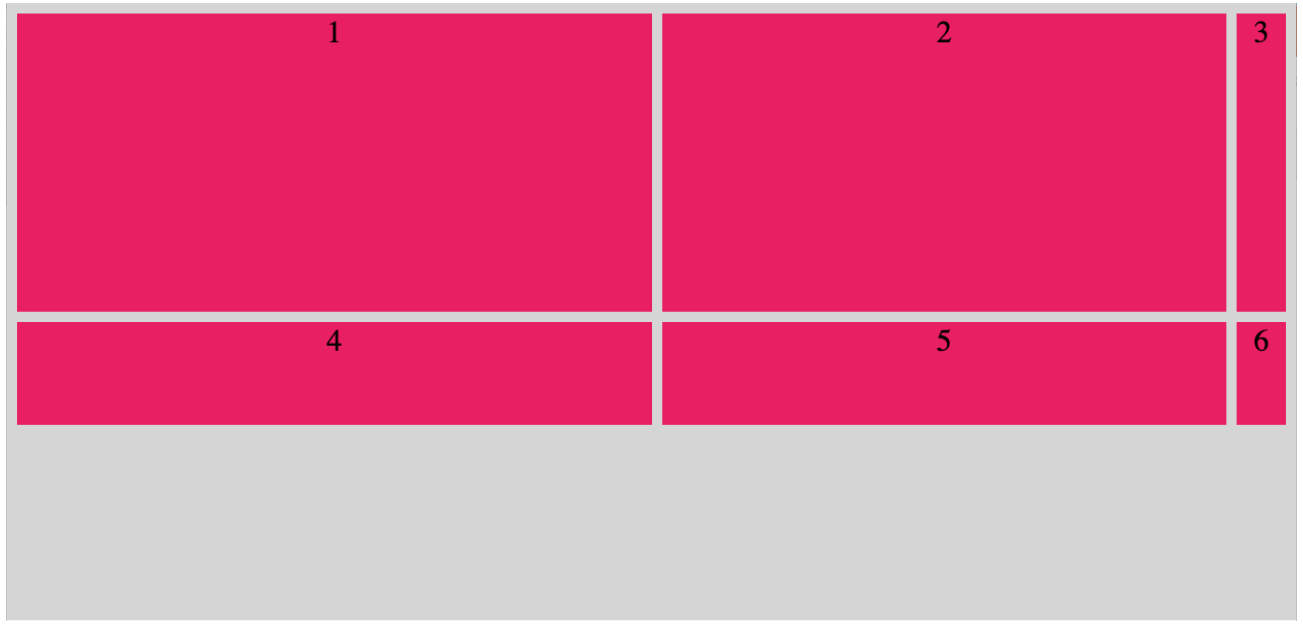
The **minmax()** function accepts 2 arguments: the first is the minimum size of the track and the second the maximum size. Alongside length values, the values can also be **auto**, which allows the track to grow/stretch based on the size of the content.



```css
.grid-minmax {
    width: 100%;
    height: 600px;
    display: grid;
    grid-template-columns: minmax(auto, 50%) 1fr 3rem;
    grid-template-rows: minmax(auto, 50%) 100px;
    grid-gap: 10px;
    background-color: #d5d5d5;
    padding: 10px;
}
.grid-minmax div {
    background-color: #E91E63;
    text-align: center;
    font-size: 30px;
}
```



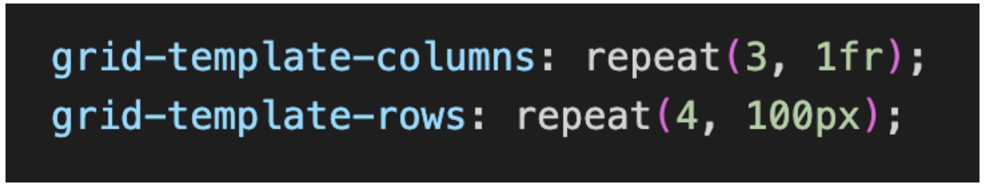
```html
<div class="grid-minmax">
    <div>1</div>
    <div>2</div>
    <div>3</div>
    <div>4</div>
    <div>5</div>
    <div>6</div>
</div>
```



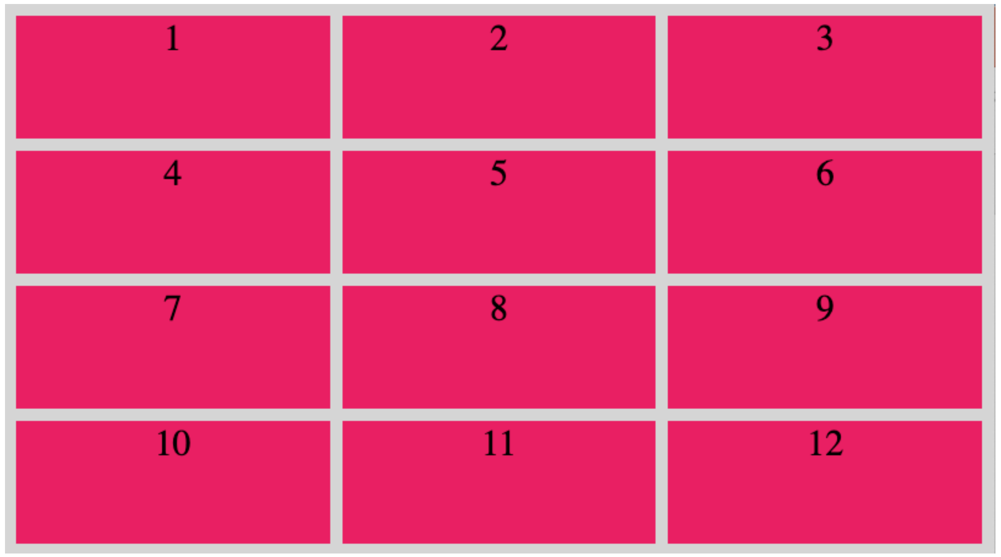
Grid Container - grid-template-columns grid-template-rows

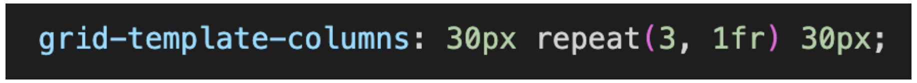
- Explicit Grid Tracks
- Min and Max Grid Tracks
- Repeating Grid Tracks

The **repeat()** notation accepts 2 arguments: the first represents the number of times the defined tracks should repeat, and the second is the track definition.

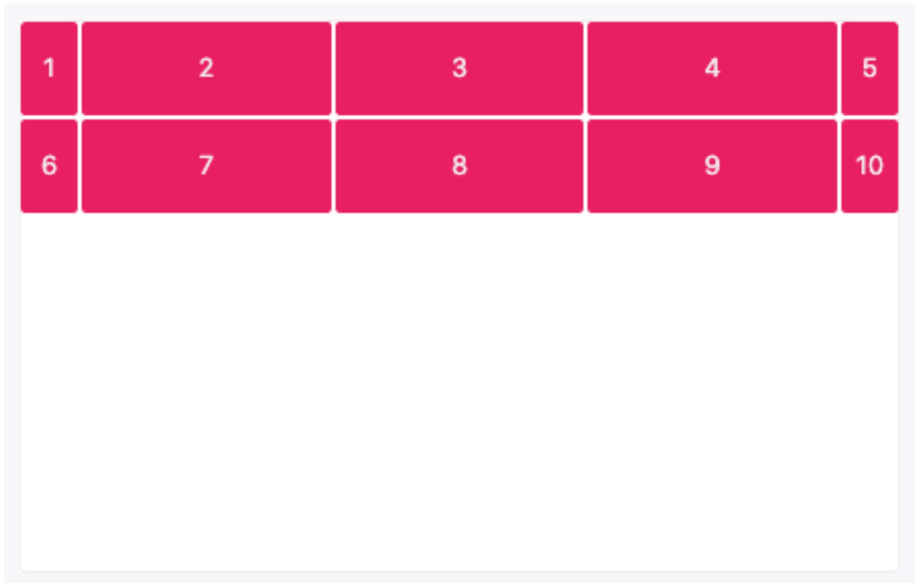


```
grid-template-columns: repeat(3, 1fr);
grid-template-rows: repeat(4, 100px);
```





```
grid-template-columns: 30px repeat(3, 1fr) 30px;
```



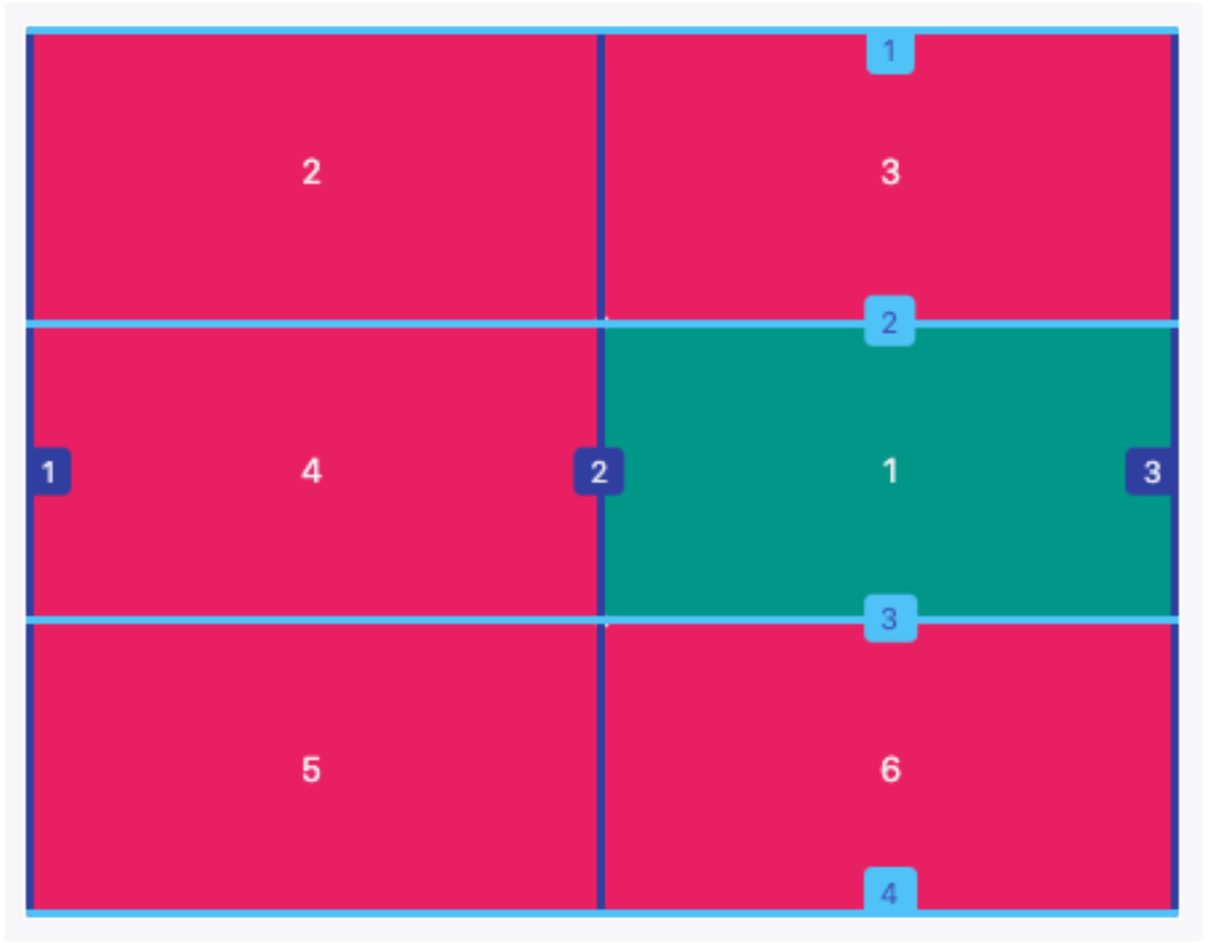
LAYOUT

Grid Items - grid-row-start
grid-row-end
grid-column-start
grid-column-end

grid-row
grid-column

grid-area

```
grid-row-start:    2;
grid-row-end:      3;
grid-column-start: 2;
grid-column-end:   3;
```



- This 2-column by 3-row grid results in 3 column lines and 4 row lines.
- **Item 1** was repositioned by row and column line numbers.

LAYOUT

Grid Items -

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

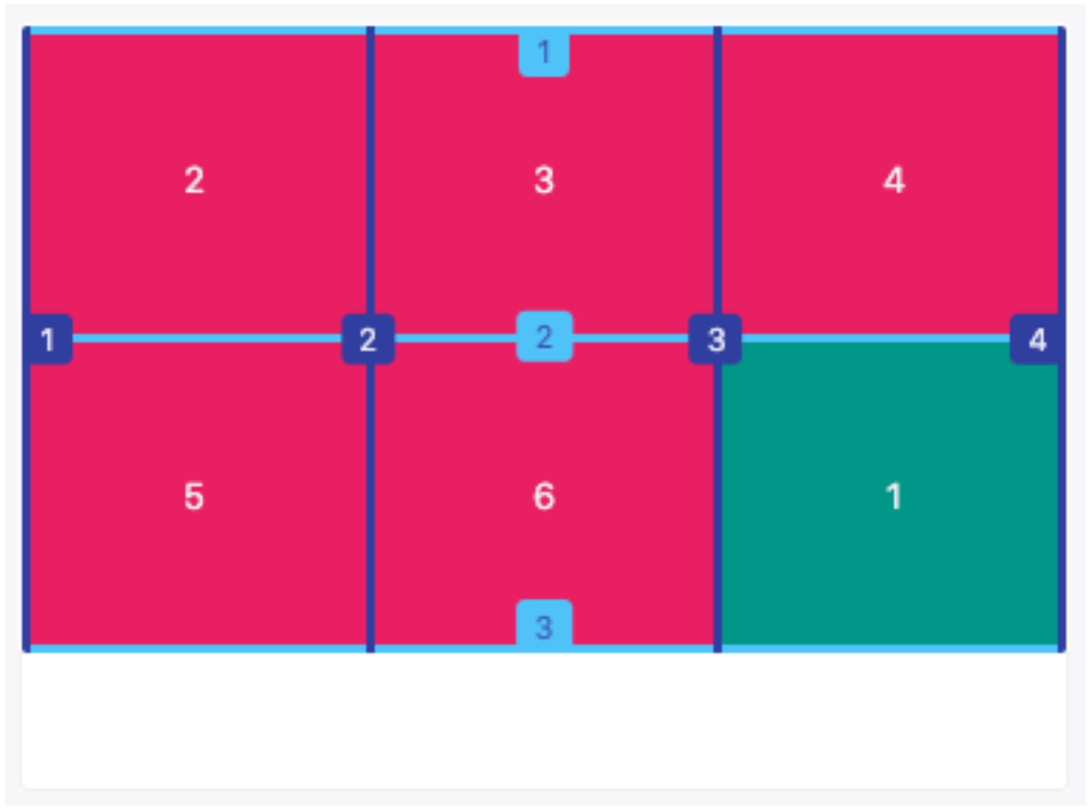
grid-row
grid-column

grid-area

grid-row : *grid-row-start / grid-row-end;*

grid-column : *grid-column-start / grid-column-end;*

```
grid-row: 2 / 3;
grid-column: 3 / 4;
```



LAYOUT

Grid Items -

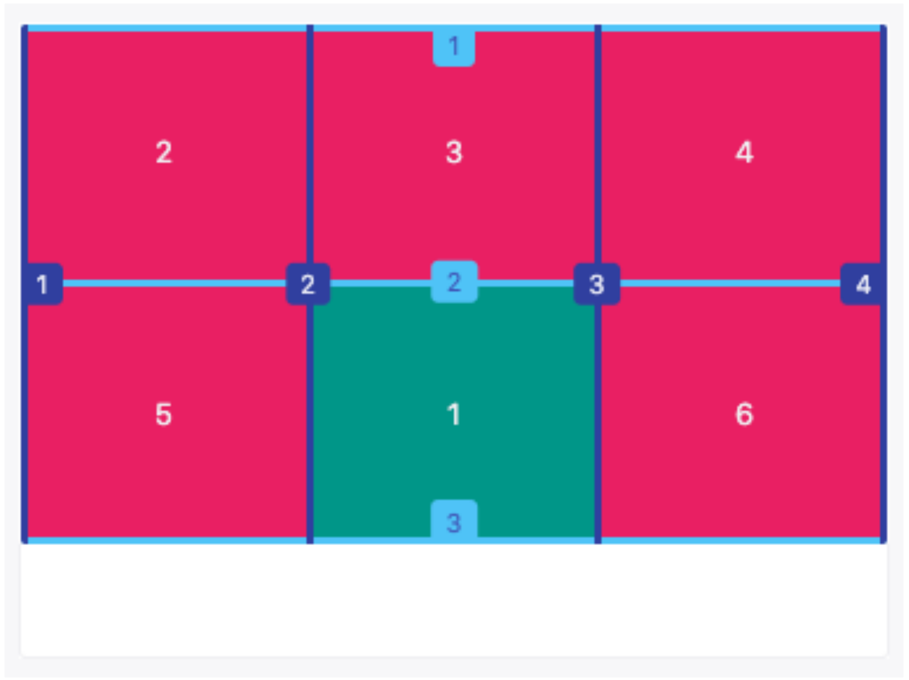
grid-row-start
grid-row-end
grid-column-start
grid-column-end

grid-row
grid-column

grid-area

grid-area *: grid-row-start / grid-column-start / grid-row-end / grid-column-end;*

grid-area: 2 / 2 / 3 / 3;



CONTENT

Grid Container

Grid Items

grid-column-gap / column-gap
grid-row-gap / row-gap
grid-gap / gap

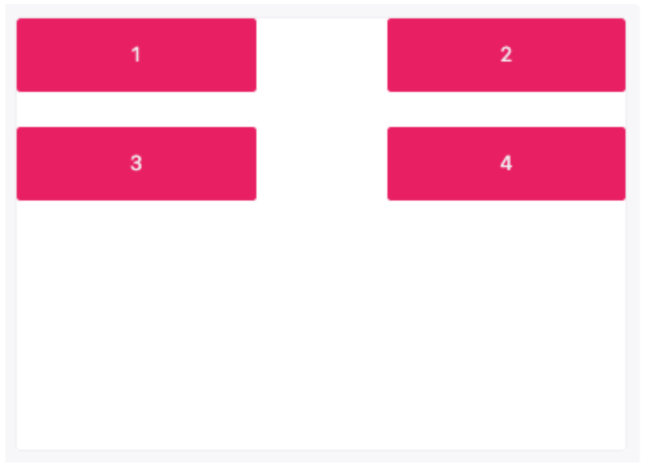
LAYOUT

Grid Container - grid-row-gap // row-gap
grid-column-gap // column-gap
grid-gap. // gap

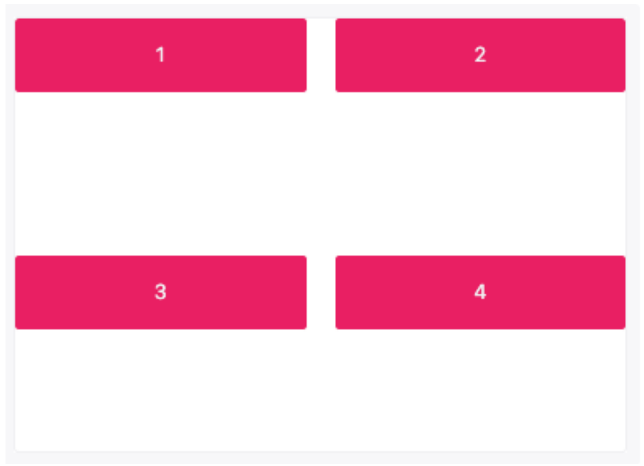
- The **grid-column-gap** and **grid-row-gap** properties create **gutters** between columns and rows.
- Grid gaps are only created in between columns and rows, and not along the edge of the grid container

grid-gap: grid-row-gap grid-column-gap;

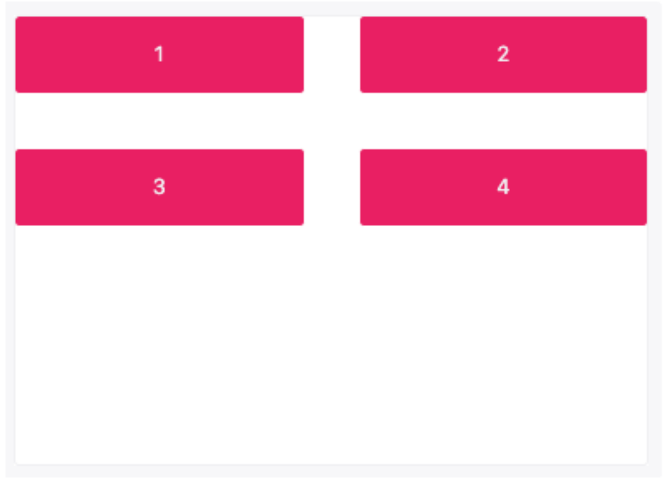
```
grid-row-gap:     20px;
grid-column-gap:  5rem;
```



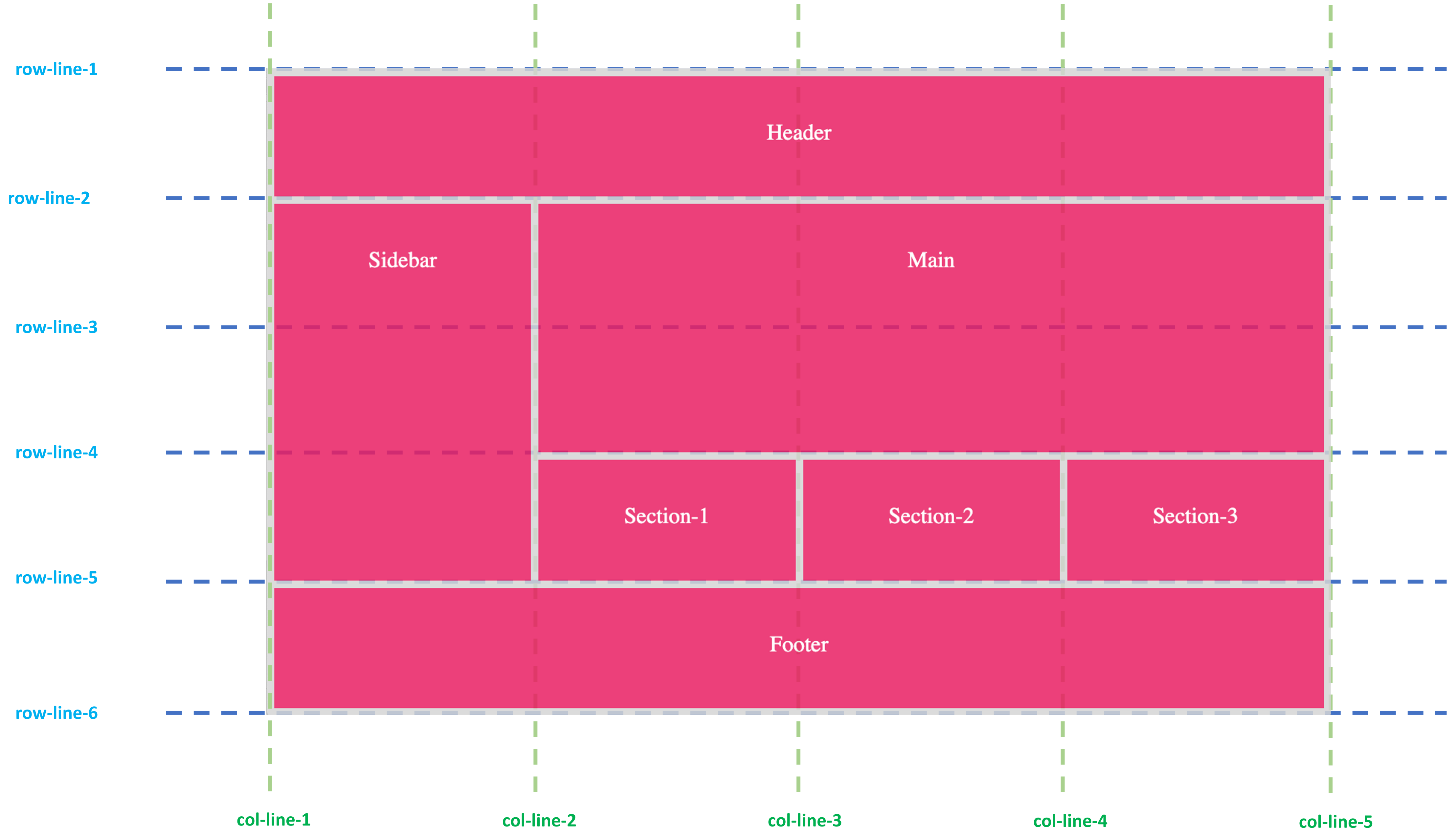
```
grid-gap: 100px 1em
```



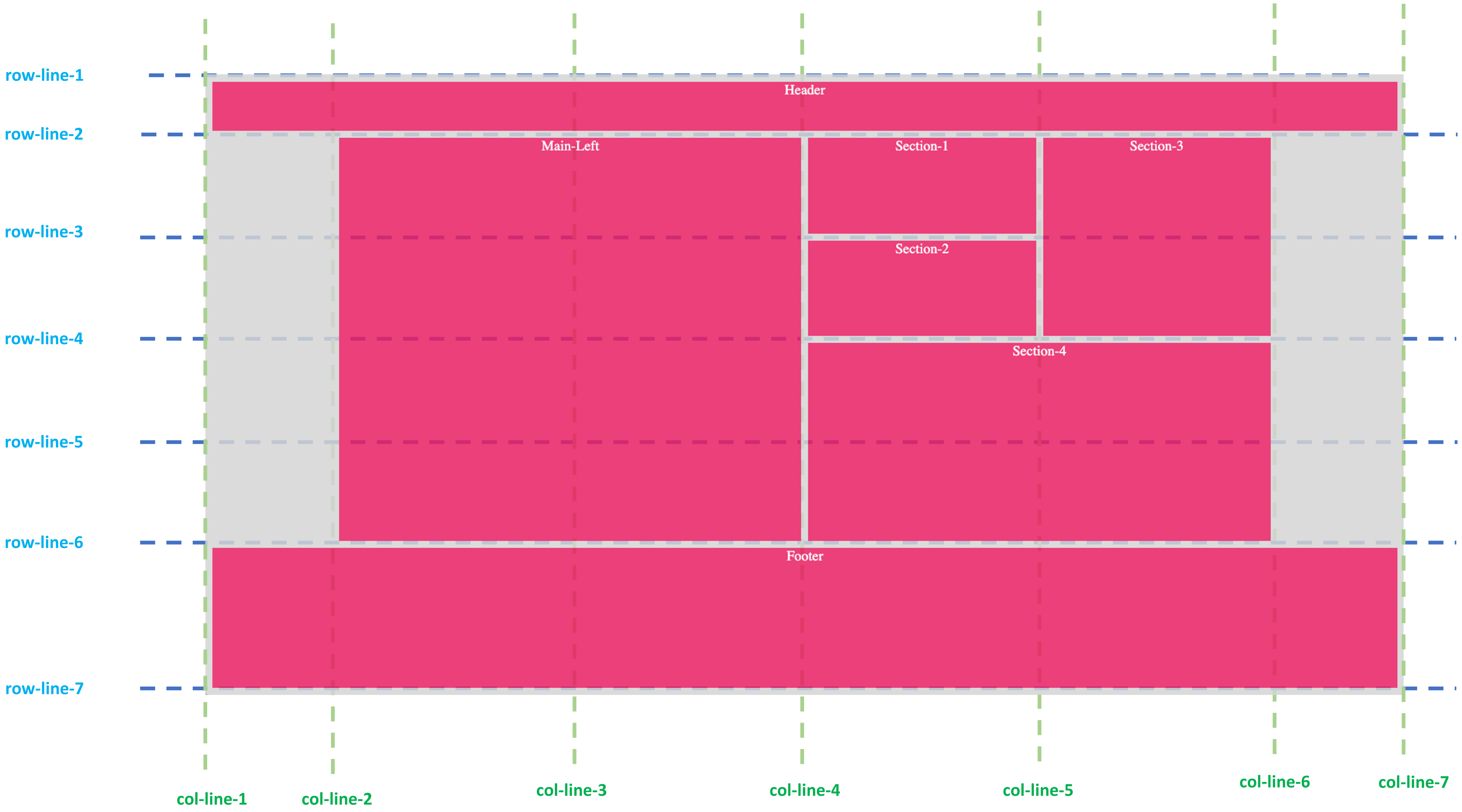
```
grid-gap: 2rem
```



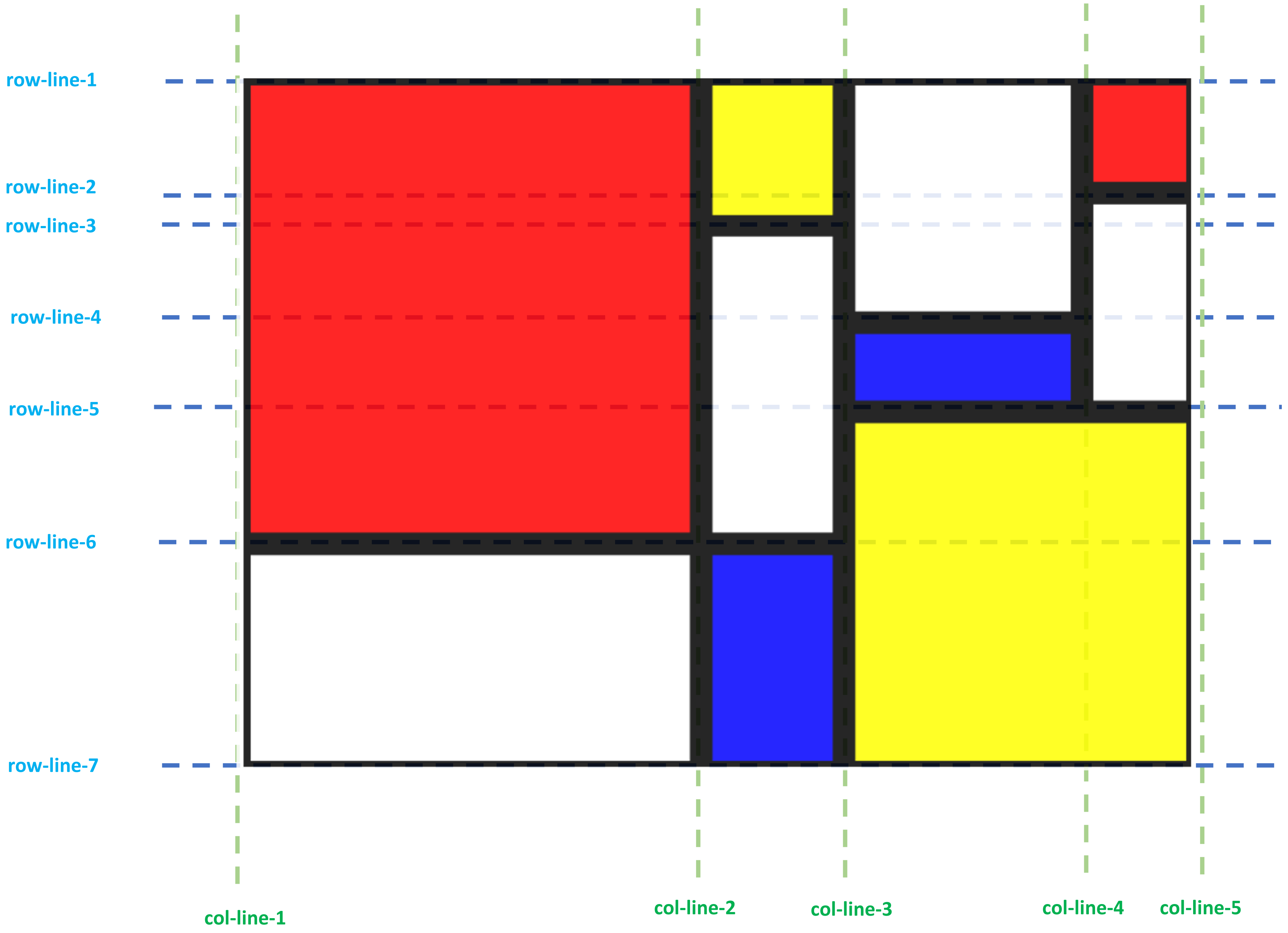
LAYOUT


row-line-1
row-line-2
row-line-3
row-line-4
row-line-5
row-line-6
Header
Sidebar
Main
Section-1
Section-2
Section-3
Footer
col-line-1
col-line-2
col-line-3
col-line-4
col-line-5

LAYOUT


row-line-1
row-line-2
row-line-3
row-line-4
row-line-5
row-line-6
row-line-7
Header
Main-Left
Section-1
Section-3
Section-2
Section-4
Footer
col-line-1
col-line-2
col-line-3
col-line-4
col-line-5
col-line-6
col-line-7

LAYOUT


row-line-1
row-line-2
row-line-3
row-line-4
row-line-5
row-line-6
row-line-7
col-line-1
col-line-2
col-line-3
col-line-4
col-line-5

CONTENT

Grid Container

grid-template-areas

- Area names

Grid Items

grid-area

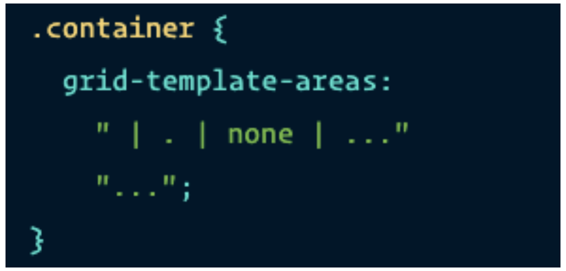
- Area names

LAYOUT

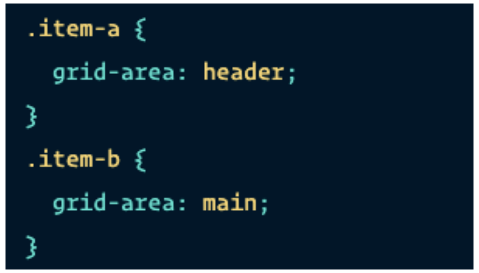
Grid Container - grid-template-areas

Grid Items - grid-area

Defines a grid template **by referencing the names of the grid areas** which are specified with the grid-area property.



```
.container {
  grid-template-areas:
    " | . | none | ..."
    "...";
}
```



```
.item-a {
  grid-area: header;
}
.item-b {
  grid-area: main;
}
```

- **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

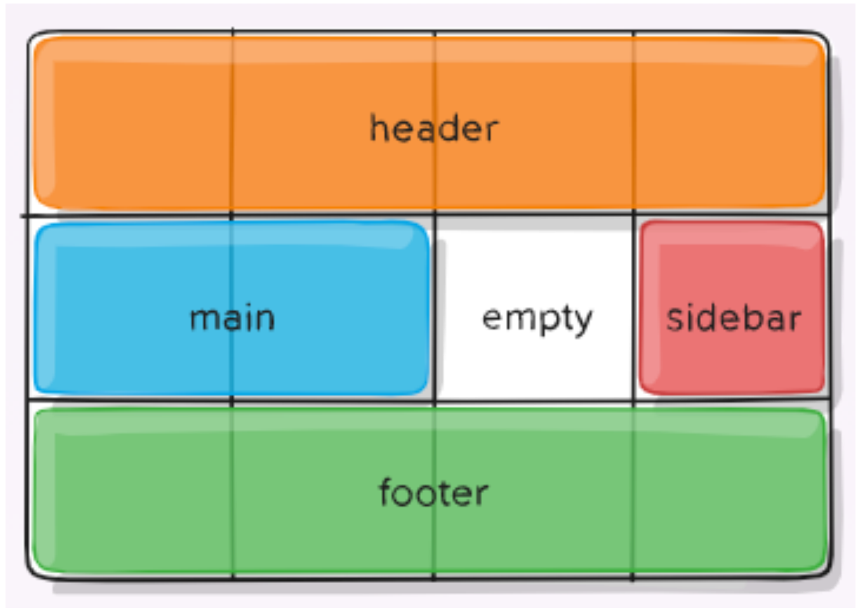
LAYOUT

Grid Container - grid-template-areas

```
.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";
}
```

Grid Items - grid-area

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



CONTENT

Grid Container

Grid Items

justify-content
align-content
place-content

- Tracks

LAYOUT

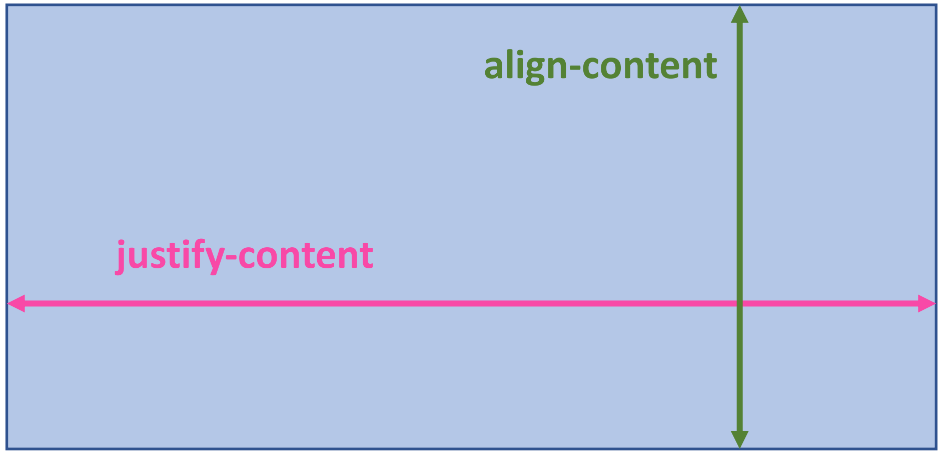
Aligning Grid Tracks

Grid Container – justify-content
align-content
place-content

- **Grid tracks** can be aligned **relative** to the grid container along the **row and column axes**.
- **align-content** aligns tracks along the **vertical axis**
- **justify-content** aligns tracks along the **horizontal axis**

- They support the following properties:

  ✓ flex-start – by default
  ✓ flex-end
  ✓ center
  ✓ space-around
  ✓ space-between
  ✓ space-evenly

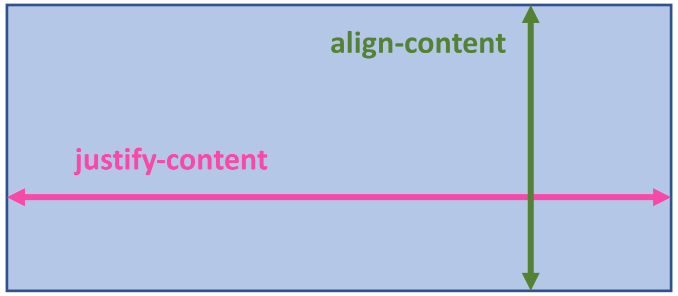


LAYOUT

Aligning Grid Tracks

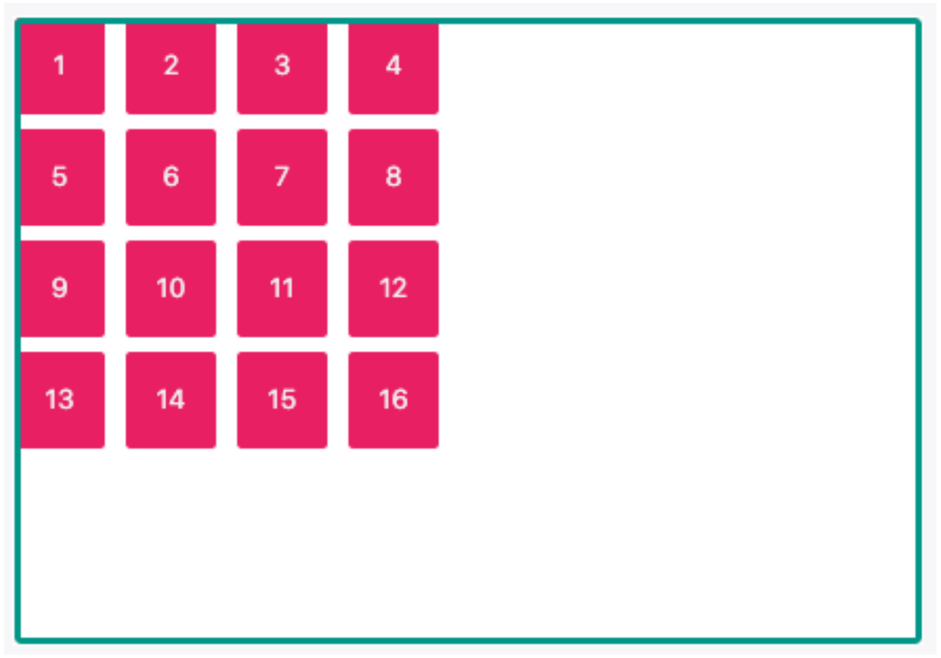
Grid Container – **justify-content**
align-content
place-content

justify-content: flex-start;



start aligns **column tracks** along and **at the start of the row axis**—it is the **default value**.

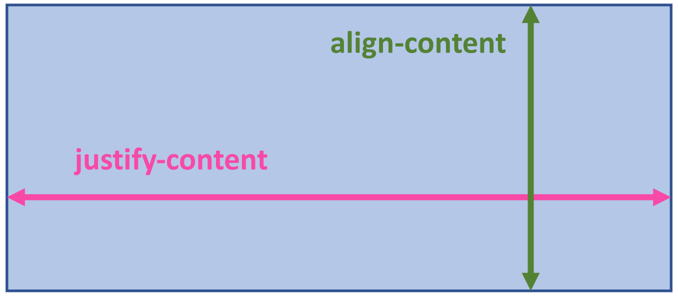
```
.grid {
  width: 100%;
  height: 300px;
  grid-template-columns: repeat(4, 45px);
  grid-template-rows: repeat(4, 45px);
  grid-gap: 0.5em;
  justify-content: start;
}
```



LAYOUT

Aligning Grid Tracks

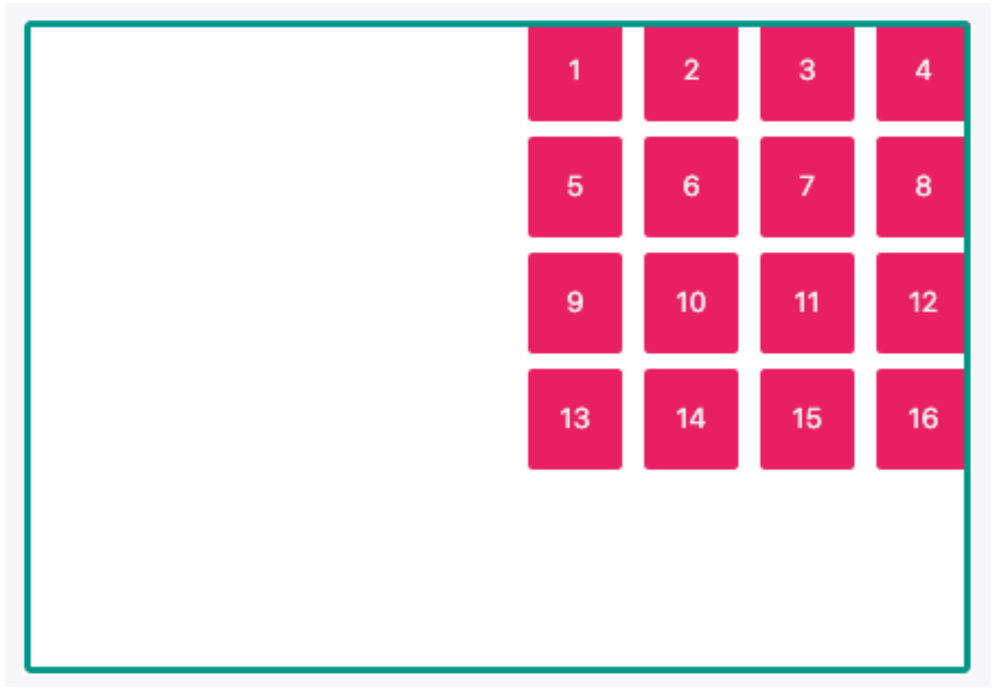
Grid Container – justify-content
align-content
place-content



justify-content: flex-end;

Columns are aligned **at the end of the row axis.**

```
justify-content: end;
```

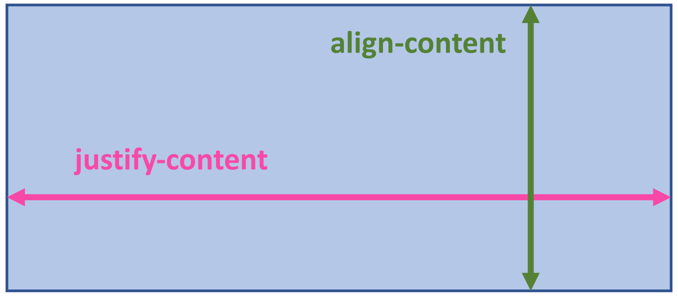


LAYOUT

Aligning Grid Tracks

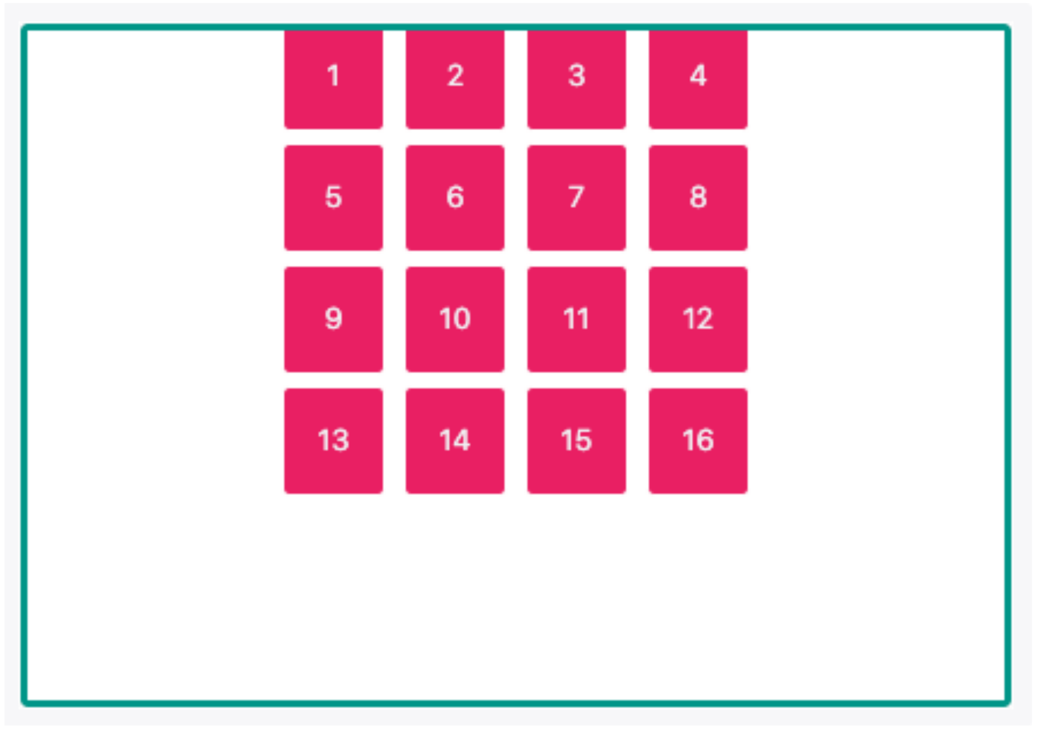
Grid Container – justify-content
align-content
place-content

justify-content: center;



Columns are aligned **at the center of the row axis**.

```
justify-content: center;
```

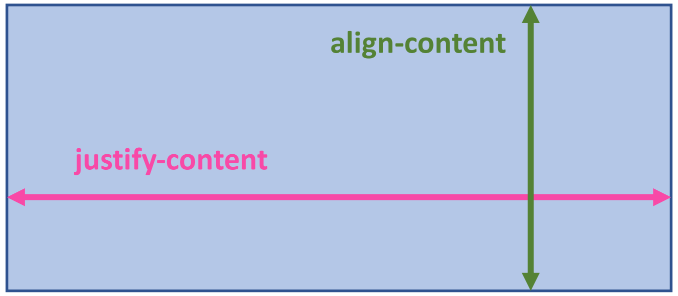


LAYOUT

Aligning Grid Tracks

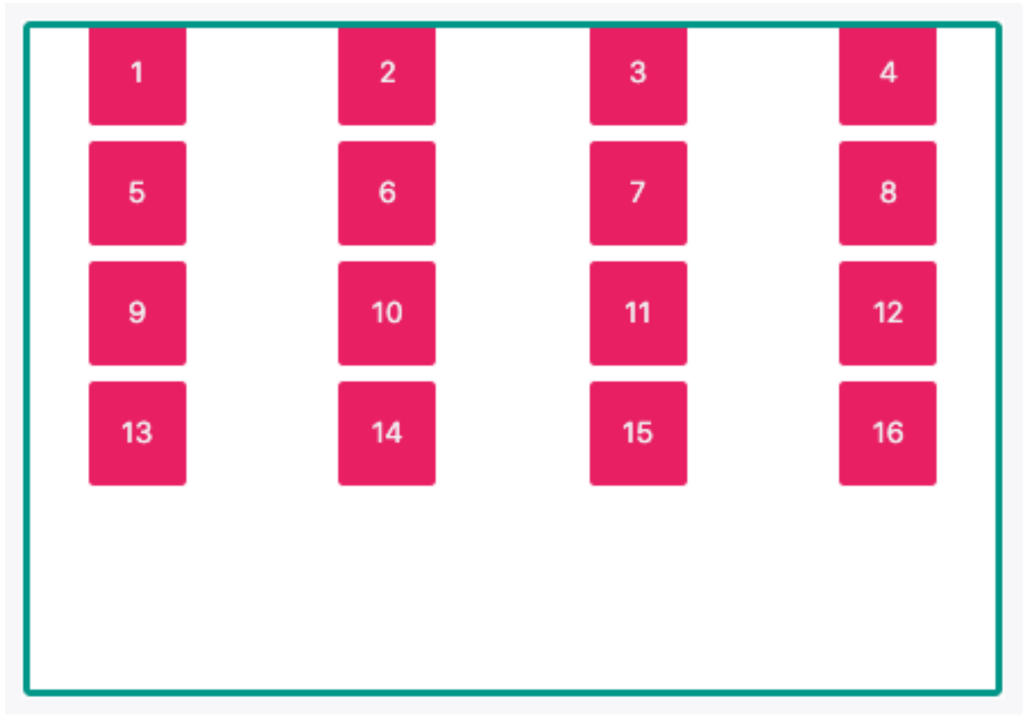
Grid Container – justify-content
align-content
place-content

justify-content: space-around;



The remaining space of the grid container is distributed and applied **to the start and end of each column track.**

```
justify-content: space-around;
```

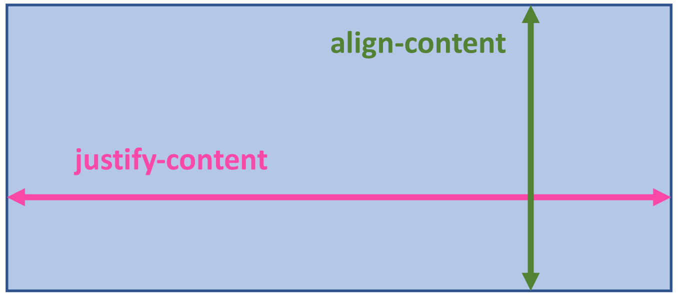


LAYOUT

Aligning Grid Tracks

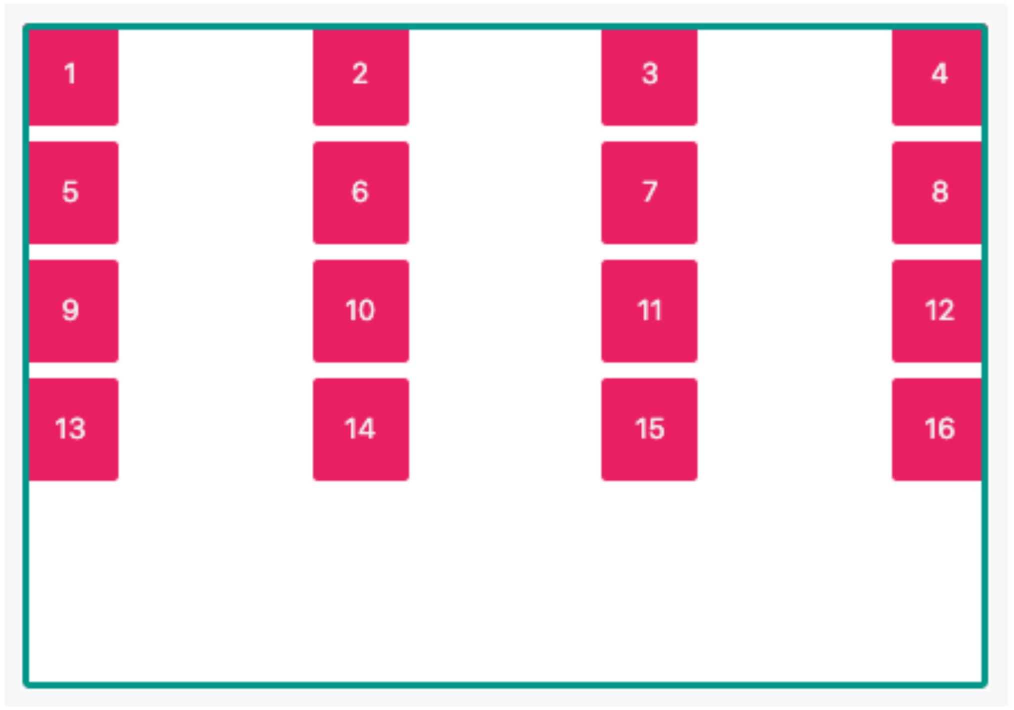
Grid Container – justify-content
align-content
place-content

justify-content: space-between;



The remaining space is distributed **between the column tracks.**

```
justify-content: space-between;
```

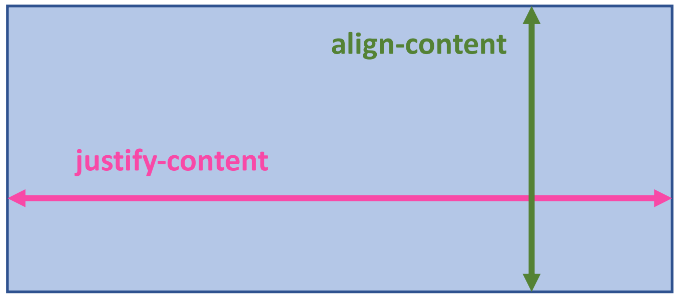


LAYOUT

Aligning Grid Tracks

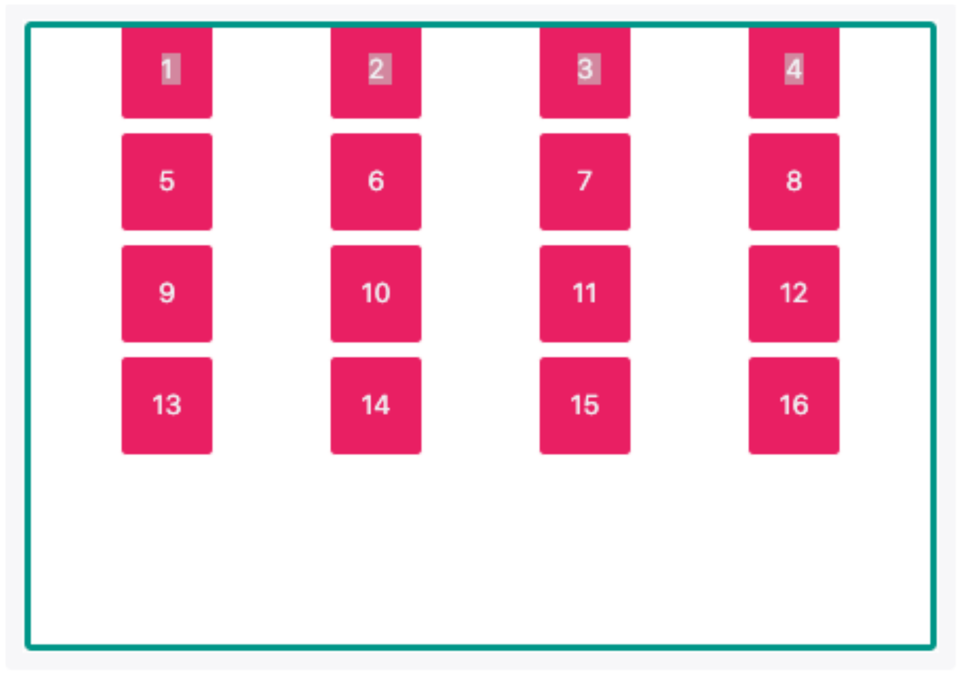
Grid Container – justify-content
align-content
place-content

justify-content: space-evenly;



The remaining space is distributed where **the space between the columns are equal to the space at the start and end of the row track.**

```
justify-content: space-evenly;
```

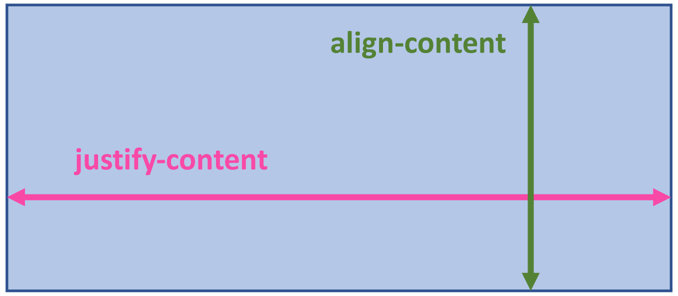


LAYOUT

Aligning Grid Tracks

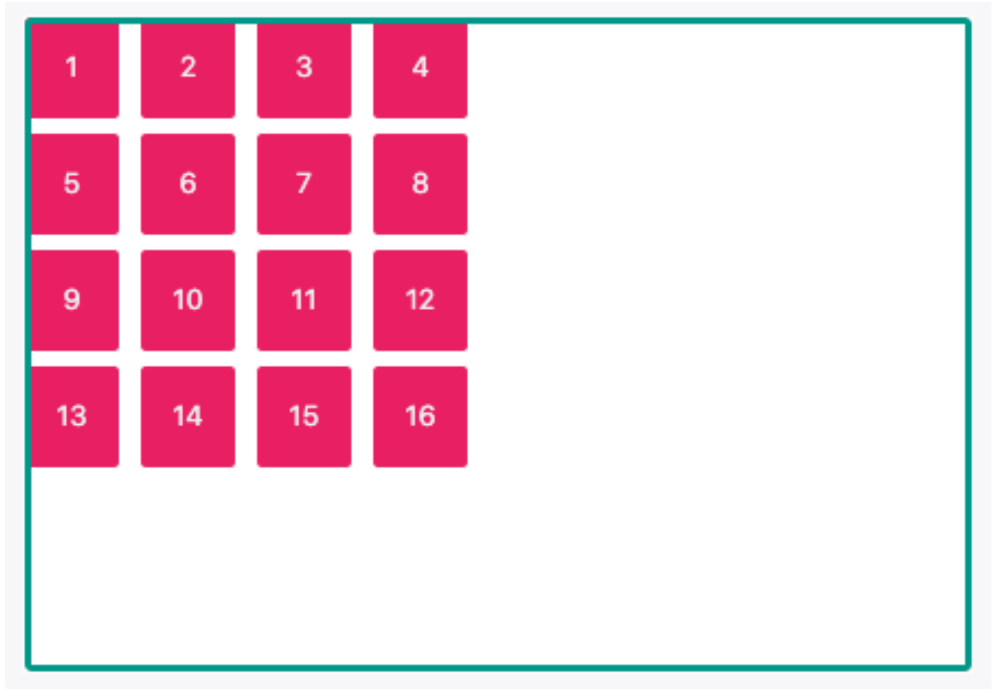
Grid Container – justify-content
align-content
place-content

align-content: flex-start;



start aligns **rows** at the **start of the column axis** and is the default value.

```
align-content: start;
```



LAYOUT

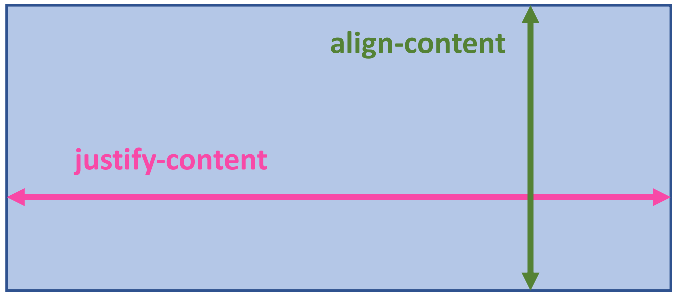
Aligning Grid Tracks

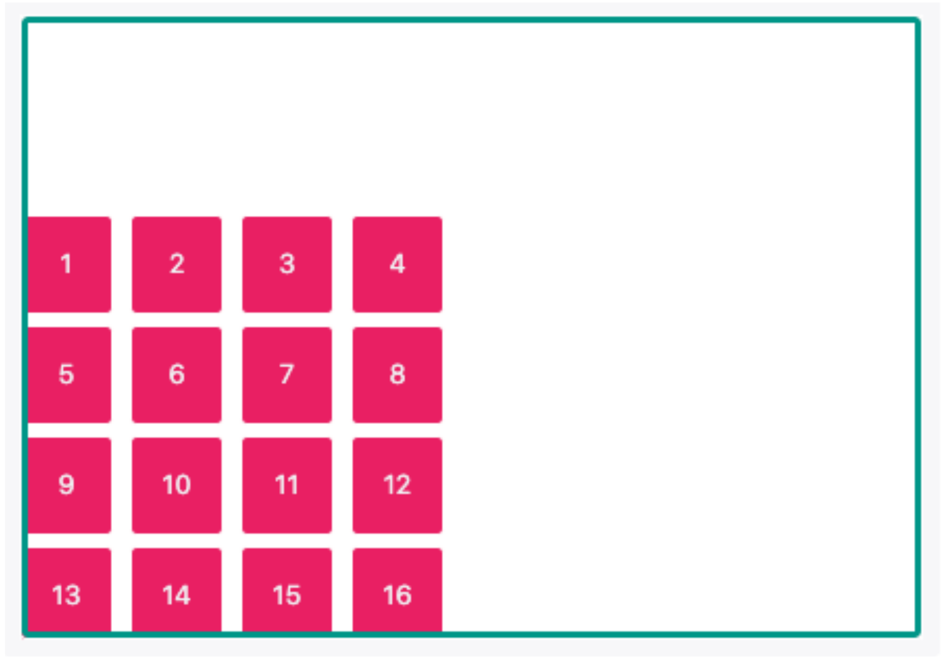
Grid Container – justify-content
align-content
place-content

align-content: flex-end;

Rows are aligned **at the end of the column axis.**

```
align-content: end;
```





LAYOUT

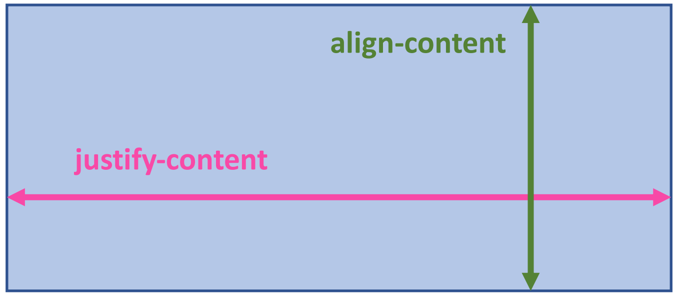
Aligning Grid Tracks

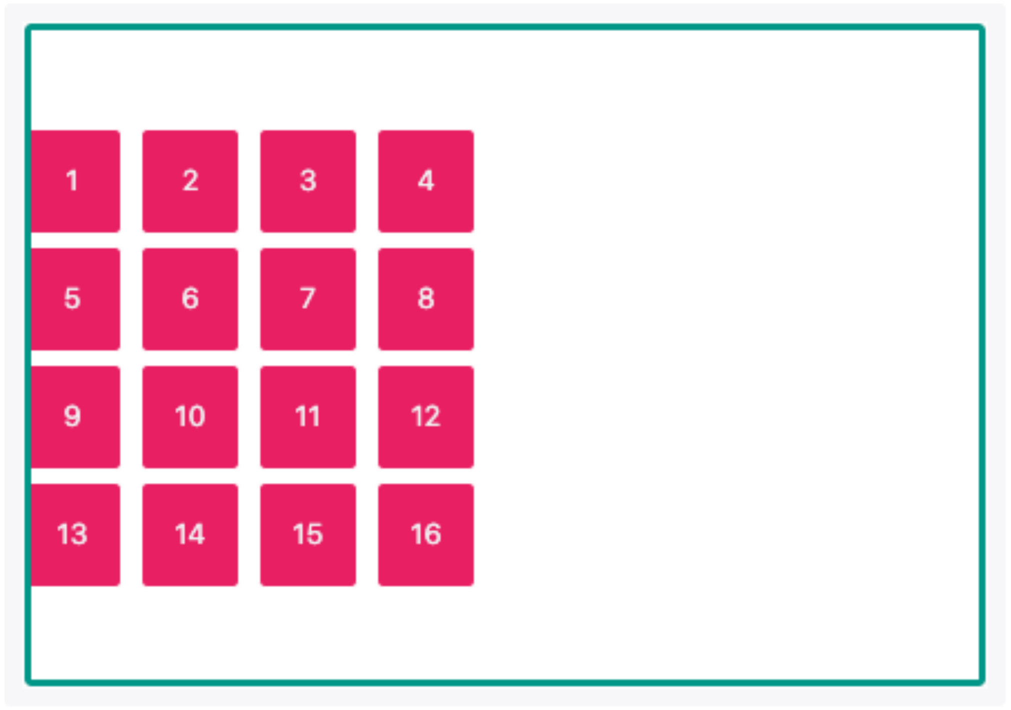
Grid Container – justify-content
align-content
place-content

align-content: center;

Rows are aligned **at the center of the column axis.**

```
align-content: center;
```

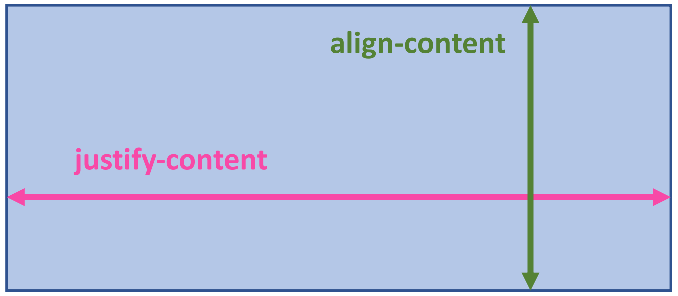




LAYOUT

Aligning Grid Tracks

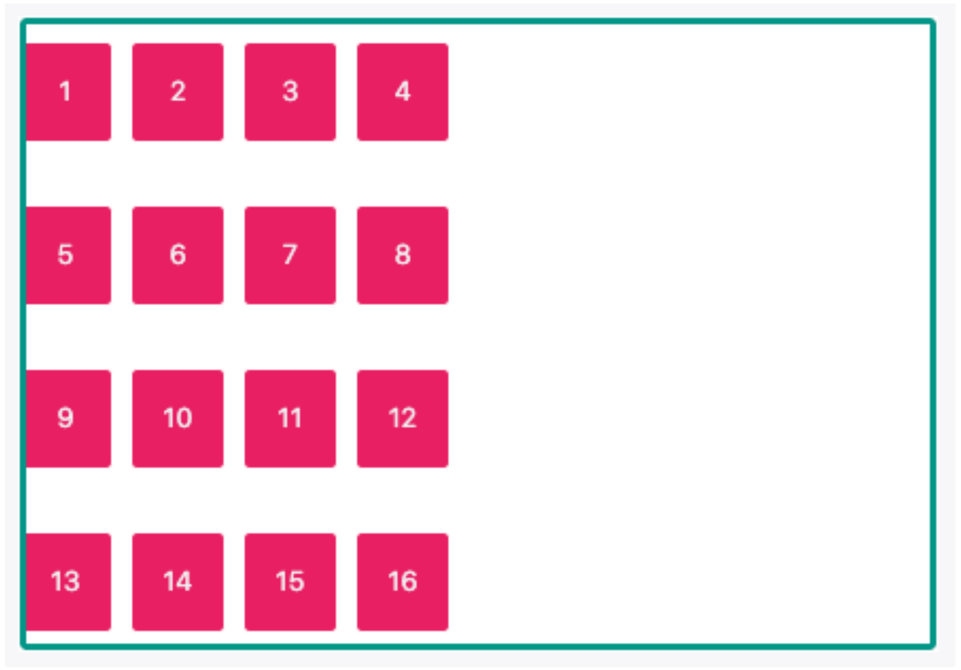
Grid Container – justify-content
align-content
place-content



align-content: space-around;

The remaining space of the grid container is distributed and applied **to the start and end of each row track.**

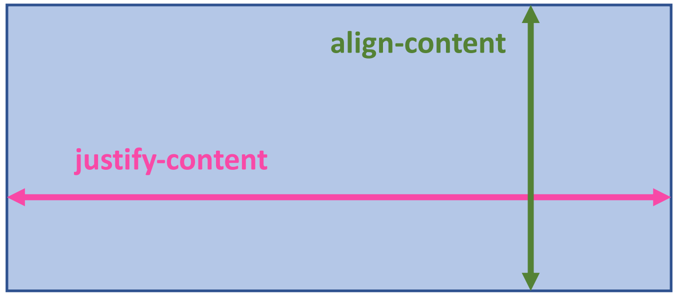
```
align-content: space-around;
```



Aligning Grid Tracks

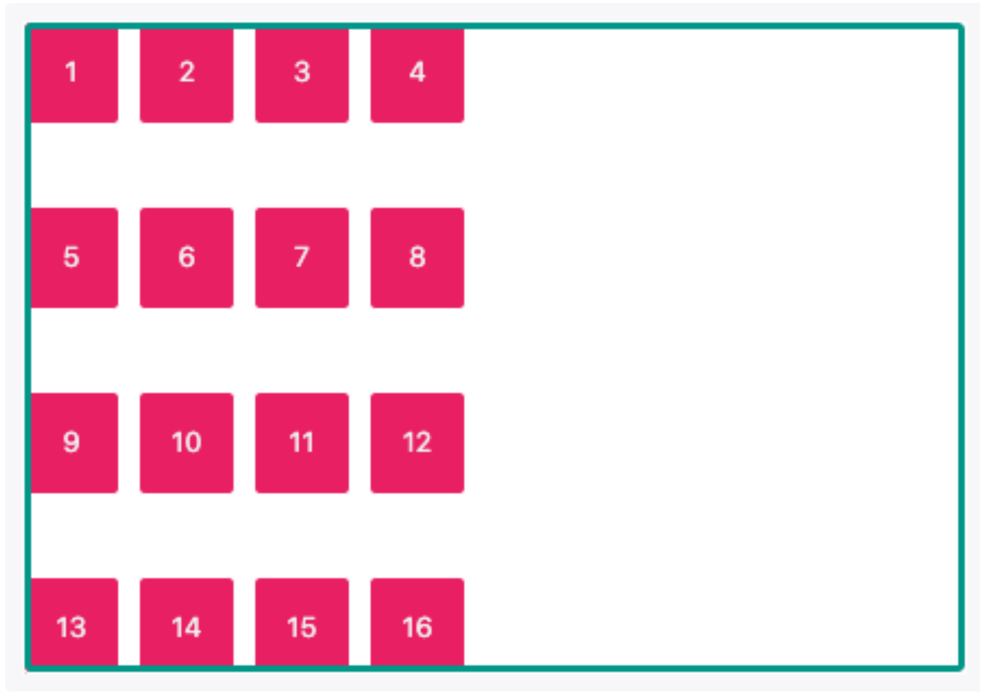
Grid Container – justify-content
align-content
place-content

align-content: space-between;



The remaining space is distributed **between the row tracks.**

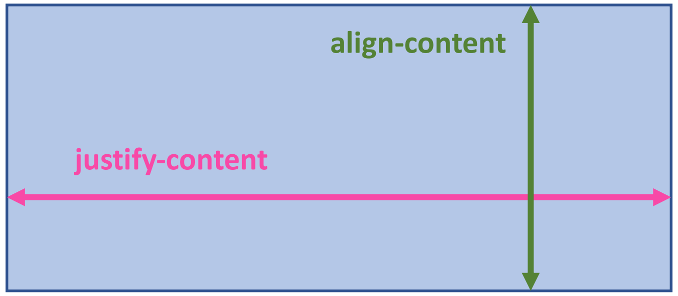
```
align-content: space-between;
```



LAYOUT

Aligning Grid Tracks

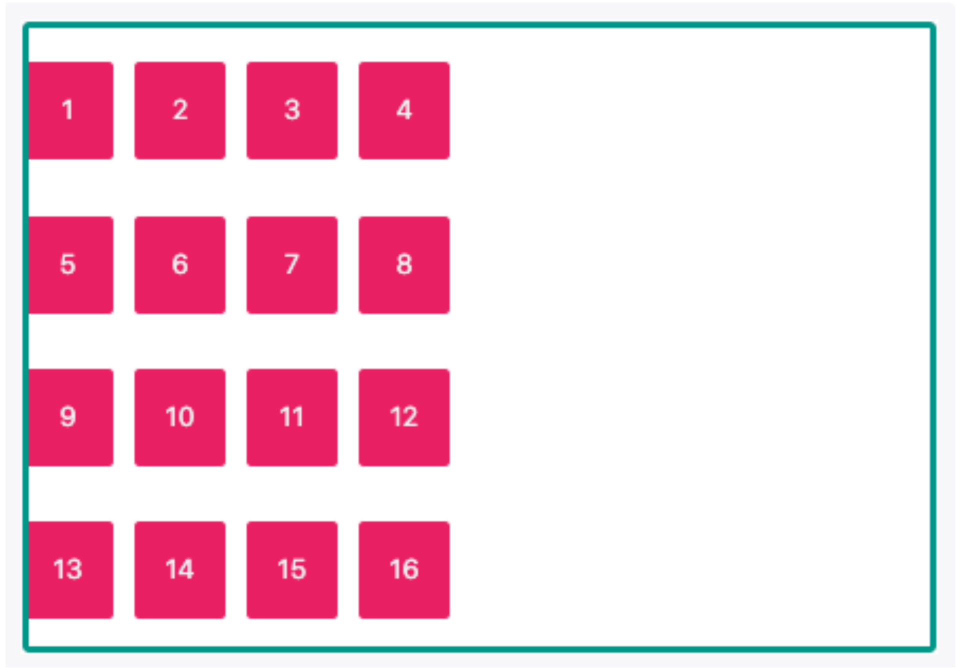
Grid Container – justify-content
align-content
place-content



align-content: space-evenly;

The remaining space is distributed where **the space between the rows are equal to the space at the start and end of the column track.**

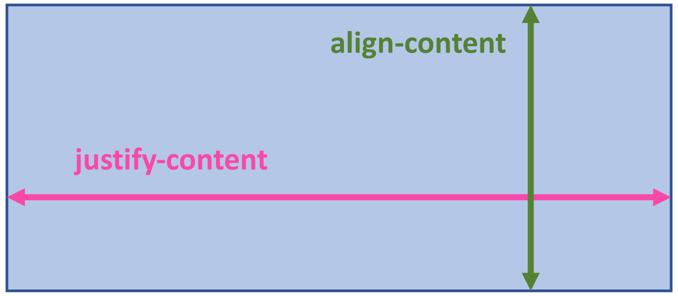
```
align-content: space-evenly;
```



LAYOUT

Aligning Grid Tracks

Grid Container – justify-content
align-content
place-content

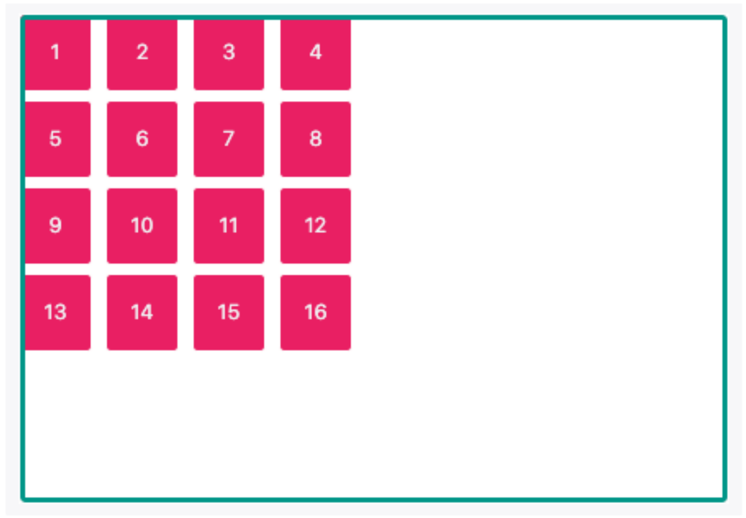


place-content sets both the **align-content** and **justify-content** properties in a single declaration.

- **<align-content> <justify-content>** – The first value sets align-content, the second value justify-content.
- If the second value is omitted, the first value is assigned **to both properties.**

```
place-content: flex-start  flex-start;
```

```
place-content: flex-start;
```



CONTENT

Grid Container

justify-items
align-items
place-items

- All Items

Grid Items

justify-self
align-self
place-self

- Individual Item

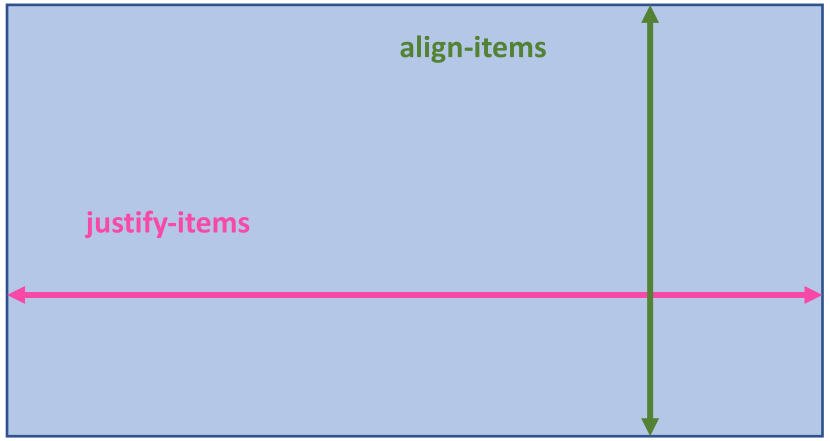
Aligning Grid Items

Grid Container – **justify-items**
align-items
place-items

- **Grid items** can be aligned **relative** to the grid container along the **row and column axes**.
- **align-items** aligns items along the **vertical axis**
- **justify-content** aligns items along the **horizontal axis**

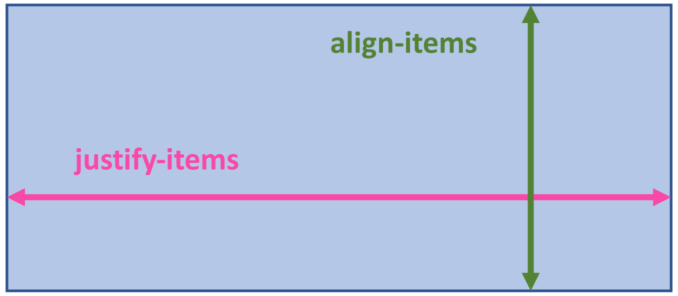
- They support the following properties:

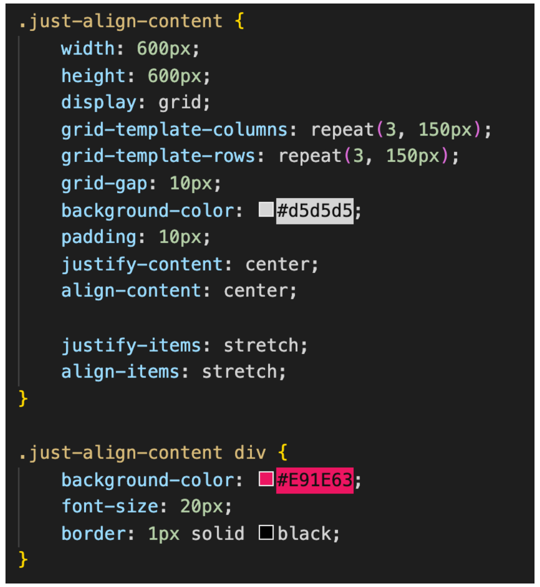
  ✓ flex-start / start
  ✓ flex-end / end
  ✓ center
  ✓ Stretch – by default

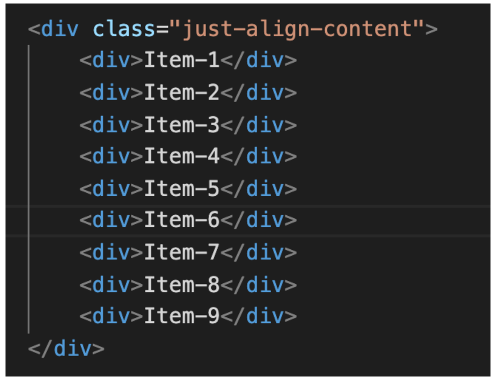


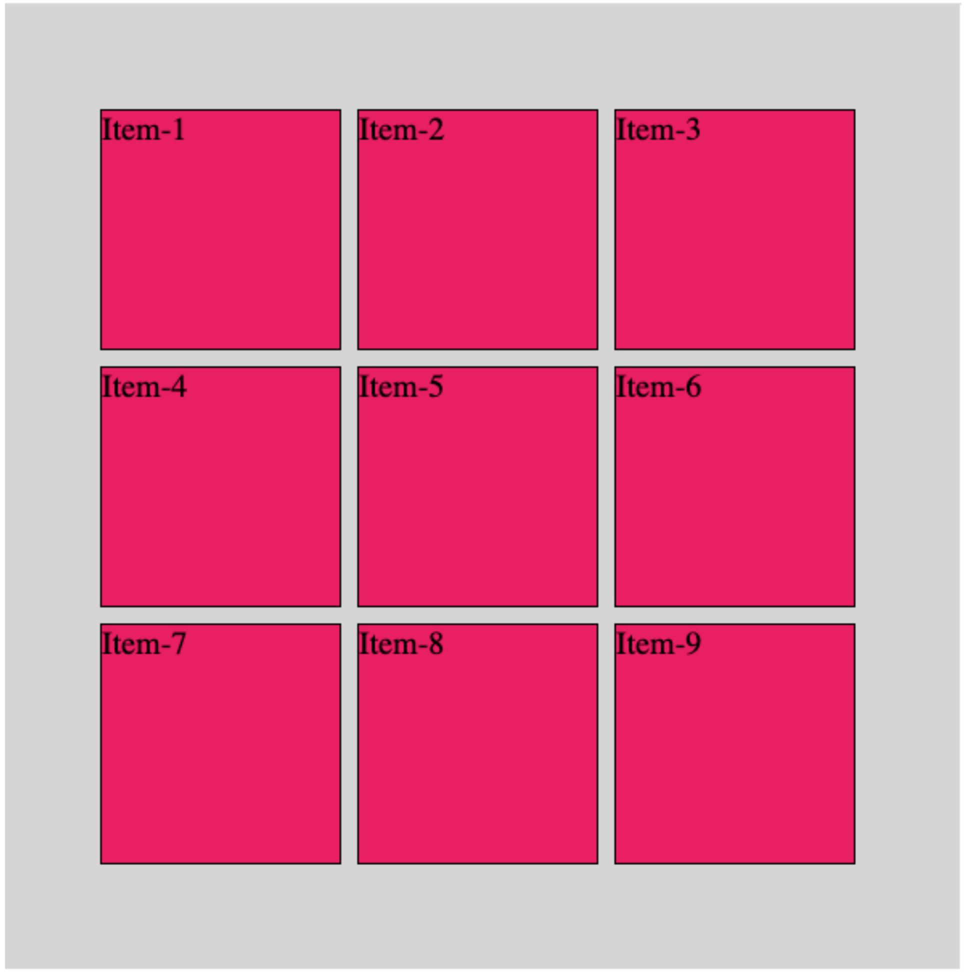
Aligning Grid Items

Grid Container – justify-items
align-items
place-items




```
.just-align-content {
    width: 600px;
    height: 600px;
    display: grid;
    grid-template-columns: repeat(3, 150px);
    grid-template-rows: repeat(3, 150px);
    grid-gap: 10px;
    background-color: #d5d5d5;
    padding: 10px;
    justify-content: center;
    align-content: center;

    justify-items: stretch;
    align-items: stretch;
}

.just-align-content div {
    background-color: #E91E63;
    font-size: 20px;
    border: 1px solid black;
}
```


```
<div class="just-align-content">
    <div>Item-1</div>
    <div>Item-2</div>
    <div>Item-3</div>
    <div>Item-4</div>
    <div>Item-5</div>
    <div>Item-6</div>
    <div>Item-7</div>
    <div>Item-8</div>
    <div>Item-9</div>
</div>
```



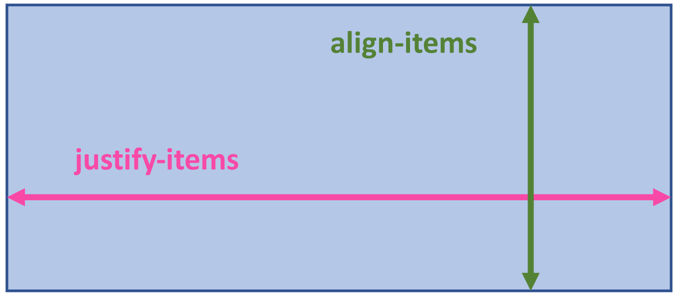
LAYOUT

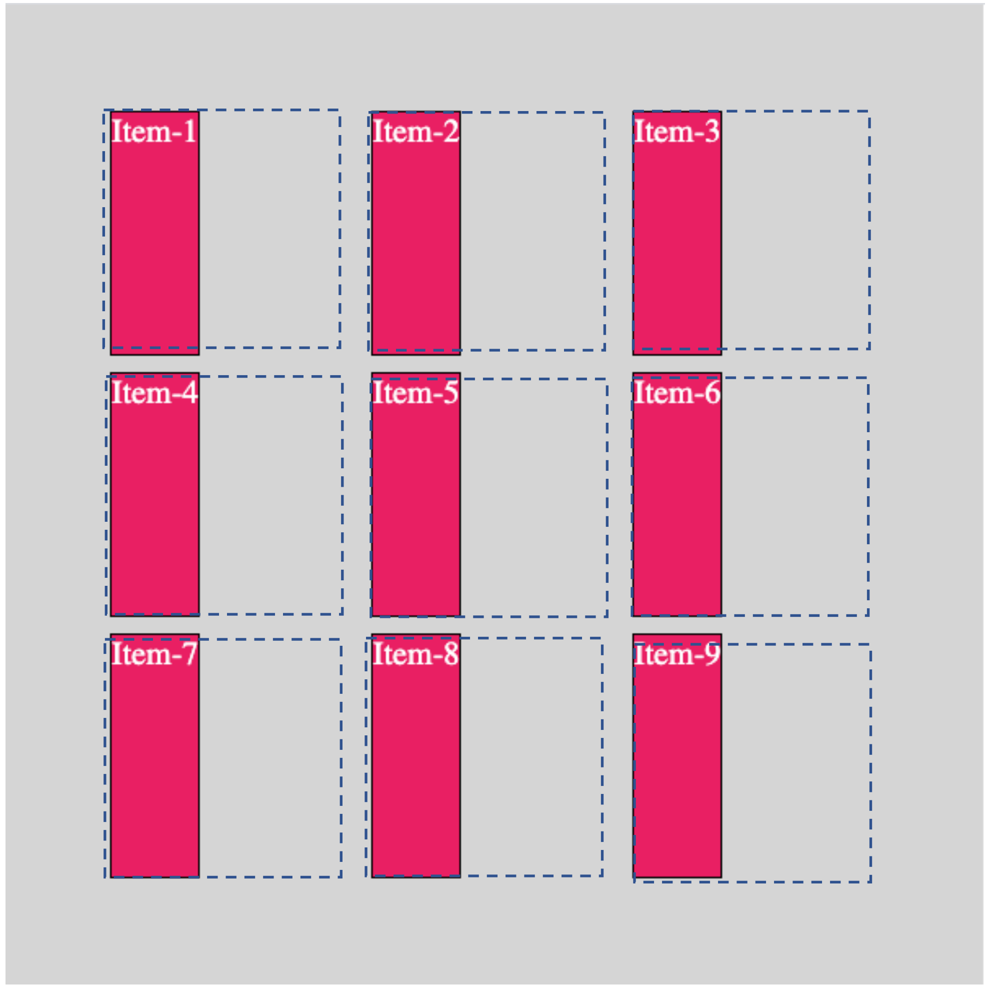
Aligning Grid Items

Grid Container – **justify-items**
align-items
place-items

justify-items: flex-start / start;

```
justify-items: flex-start;
align-items: stretch;
```



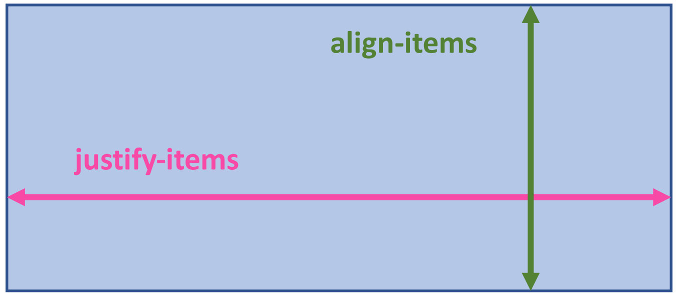


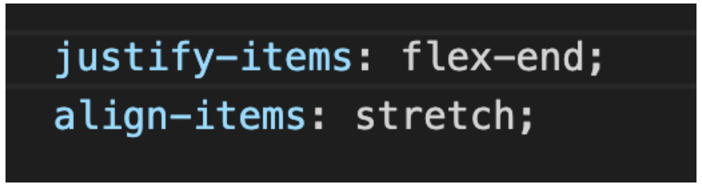
LAYOUT

Aligning Grid Items

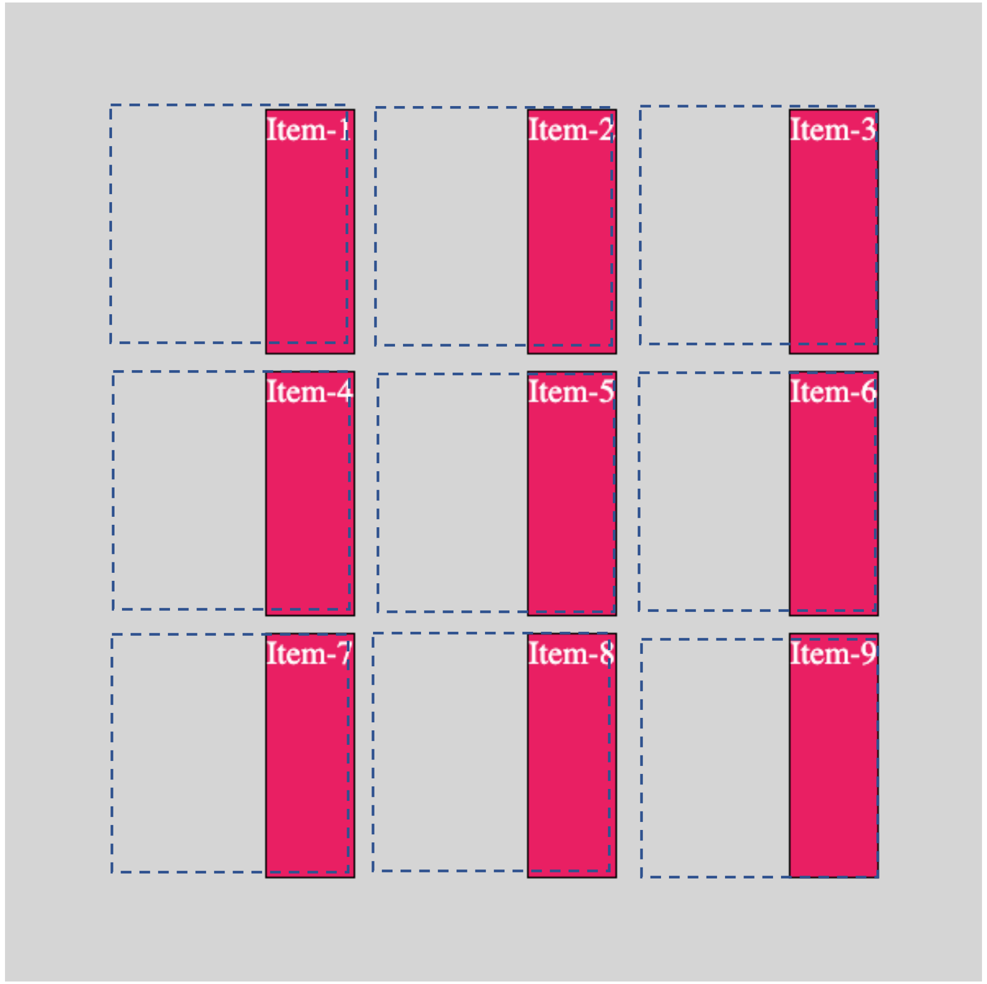
Grid Container – justify-items
align-items
place-items

justify-items: flex-end / end;





```
justify-items: flex-end;
align-items: stretch;
```



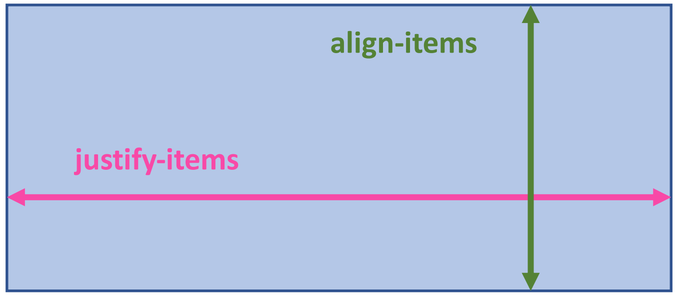
LAYOUT

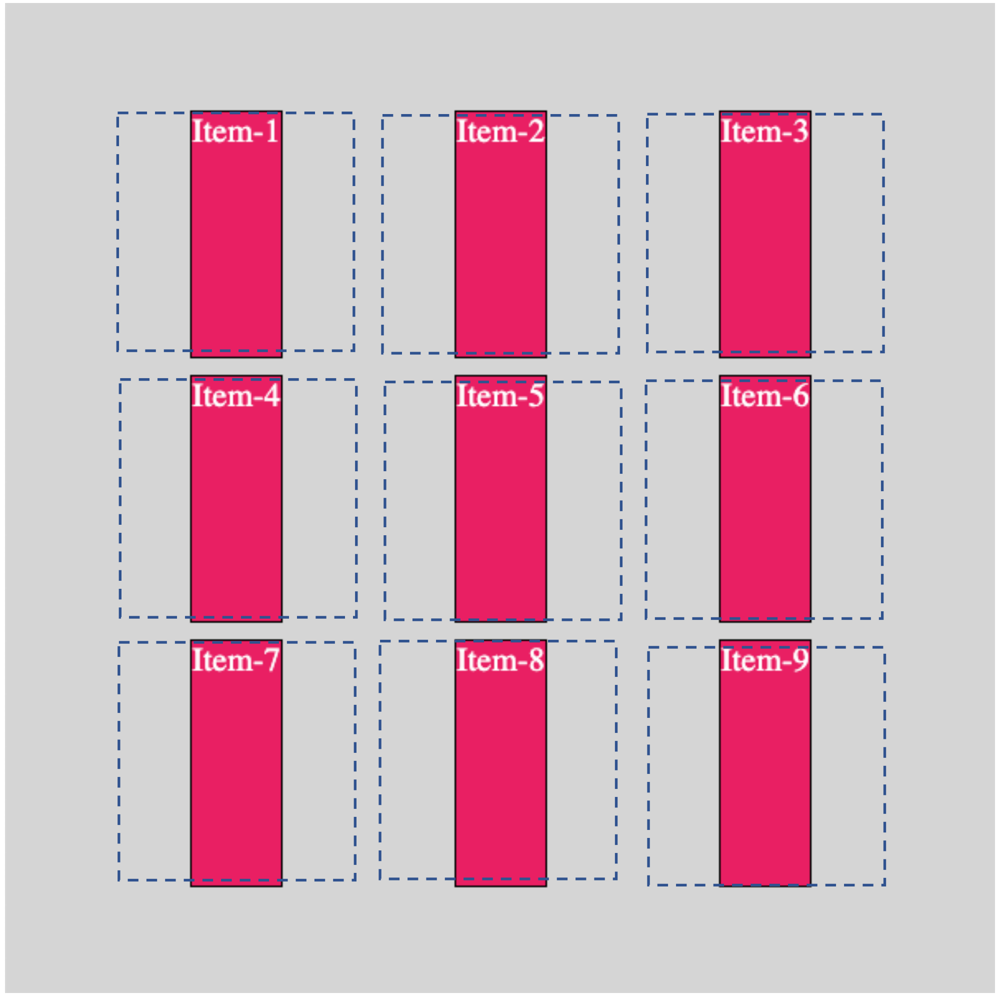
Aligning Grid Items

Grid Container – justify-items
align-items
place-items

justify-items: center;

```
justify-items: center;
align-items: stretch;
```

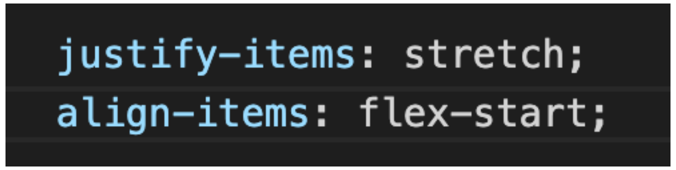




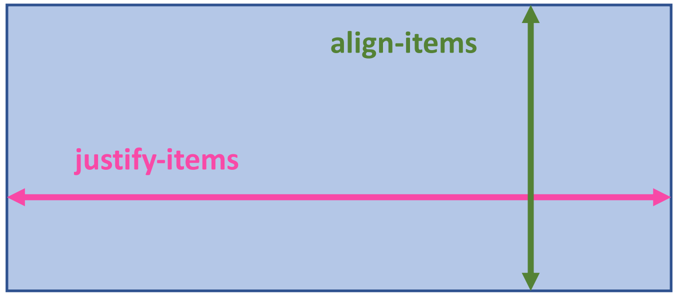
Aligning Grid Items

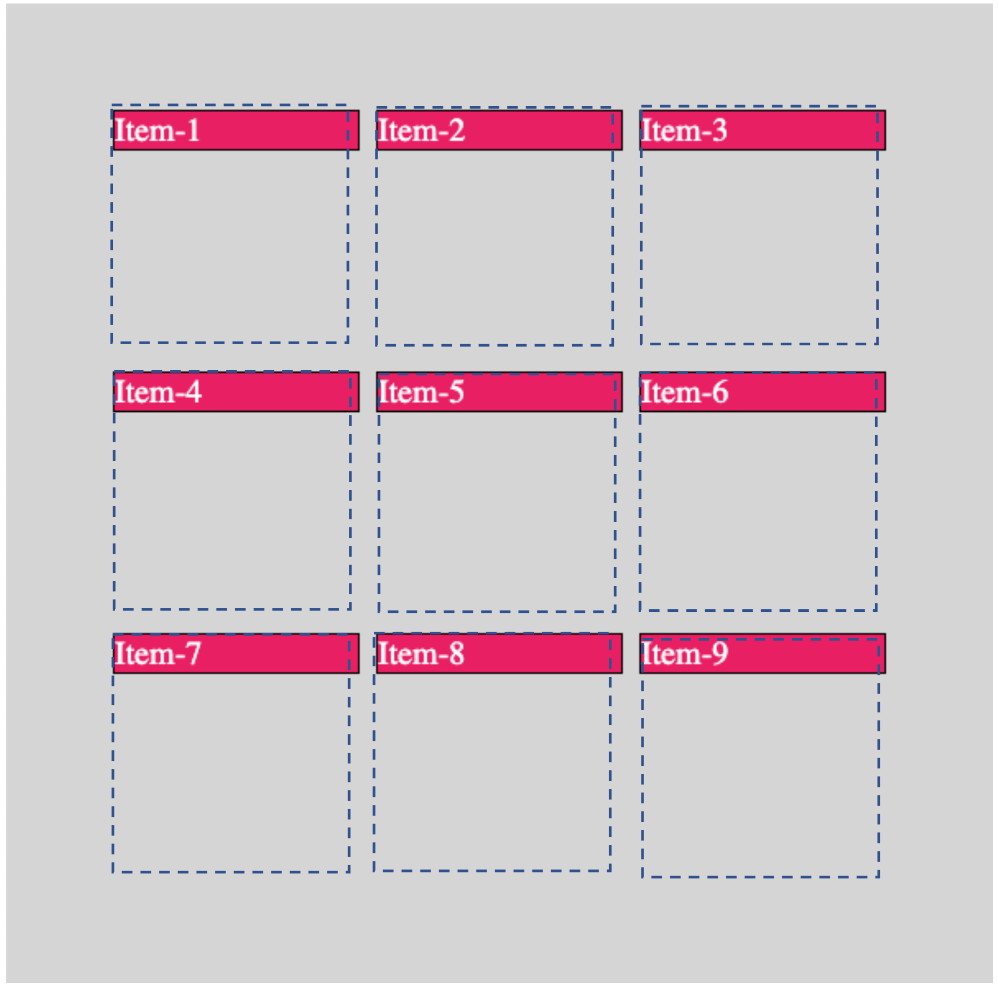
Grid Container – justify-items
align-items
place-items

align-items: flex-start / start;



```
justify-items: stretch;
align-items: flex-start;
```





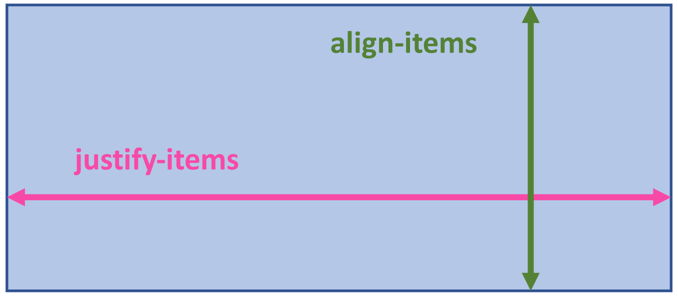
LAYOUT

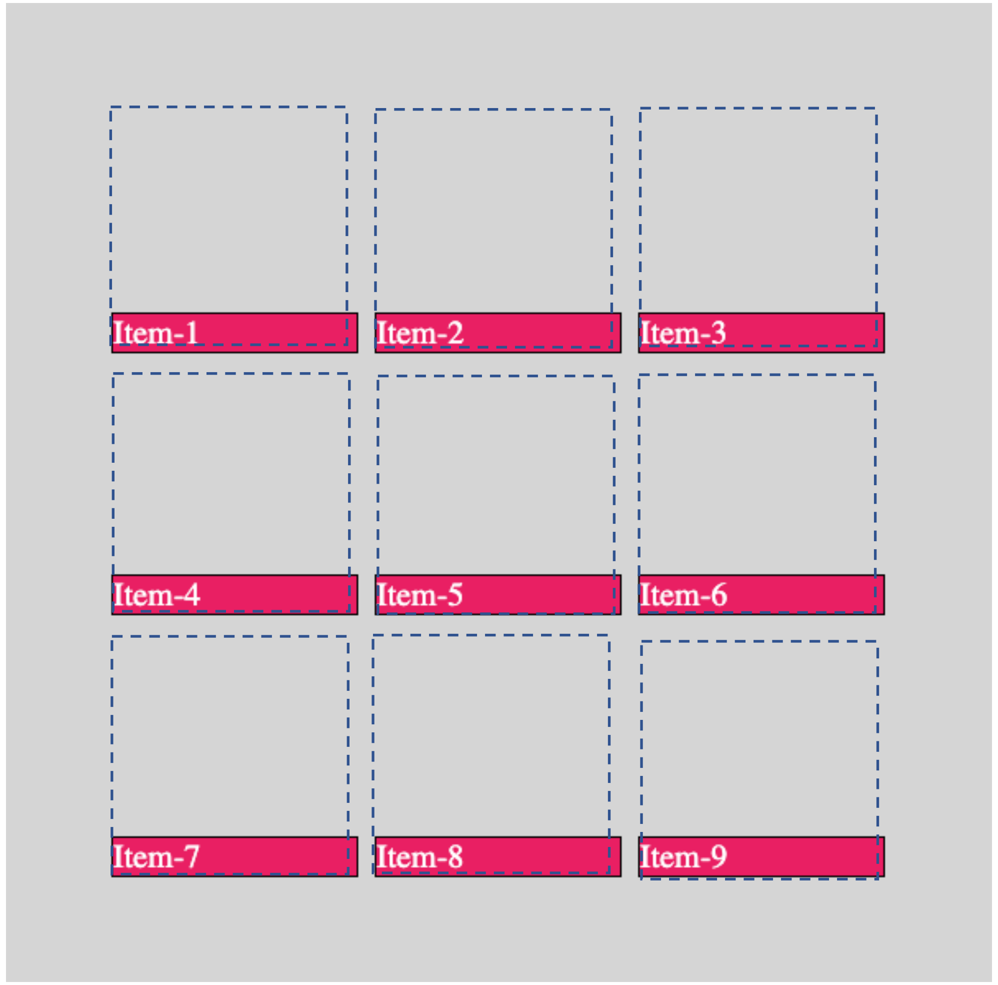
Aligning Grid Items

Grid Container – justify-items
align-items
place-items

align-items: flex-end / end;

```
justify-items: stretch;
align-items: flex-end;
```





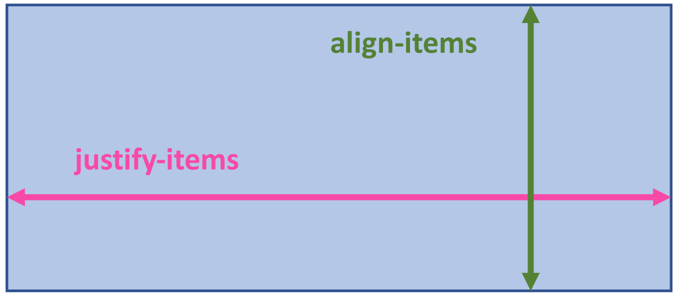
LAYOUT

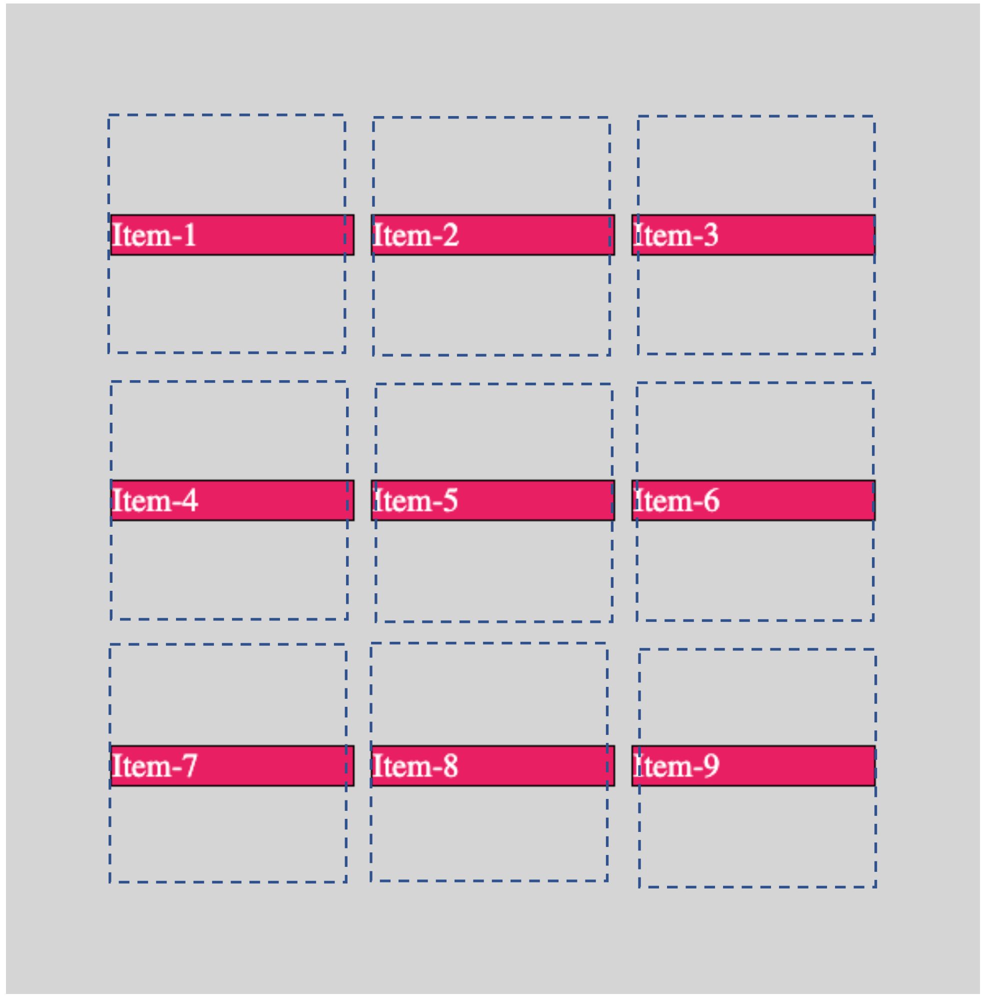
Aligning Grid Items

Grid Container – justify-items
align-items
place-items

align-items: flex-end / end;

```
justify-items: stretch;
align-items: center;
```

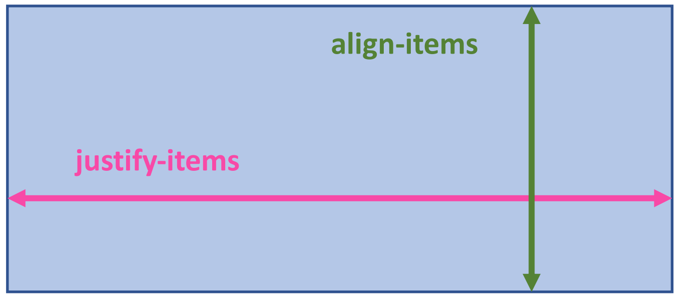




LAYOUT

Aligning Grid Items

Grid Container – justify-items
align-items
place-items

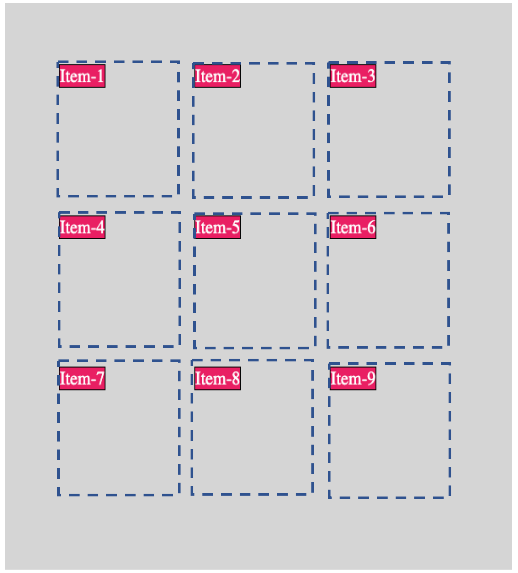


place-items sets both the align-items and justify-items properties in a single declaration.

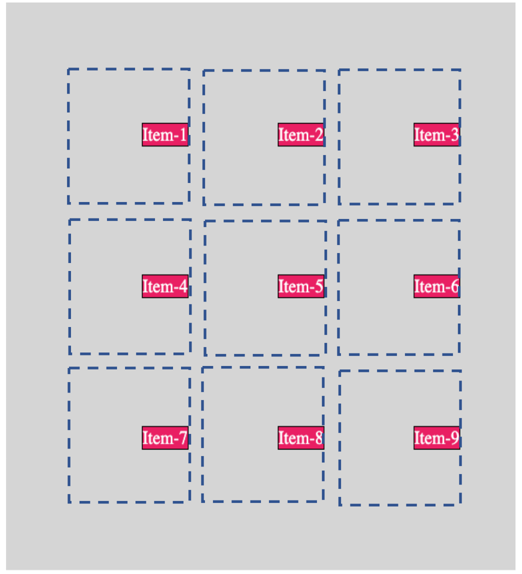
- **<align-items> <justify-items>** – The first value sets align-items, the second value justify-items.
- If the second value is omitted, the first value is assigned to both properties.

place-items: flex-start flex-start;

place-items: flex-start;



place-items: center flex-end;

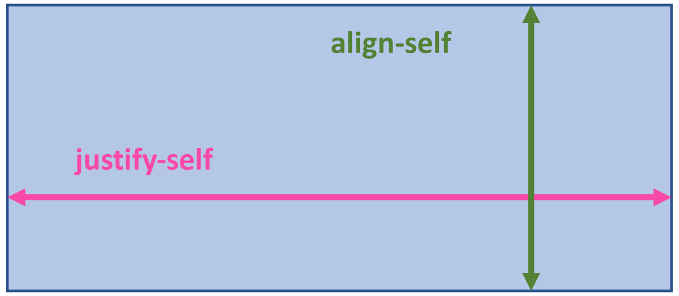


Aligning Individual Grid Items

Grid Items – **justify-self**
align-self
place-self

- **Individual items** can be self-aligned with the **align-self** and **justify-self** properties.
- These properties support the following values:
  - ✓ start
  - ✓ end
  - ✓ center
  - ✓ Stretch – by default

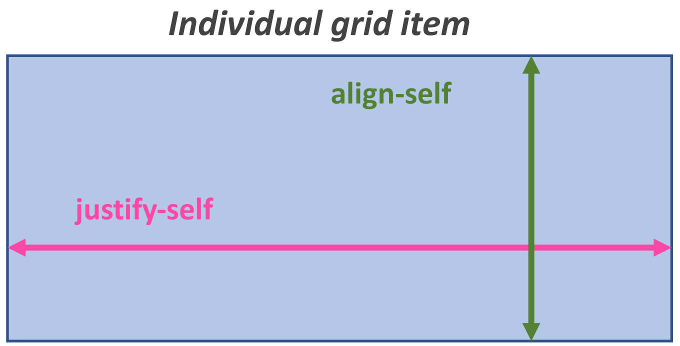
Individual grid item



LAYOUT

Aligning Individual Grid Items

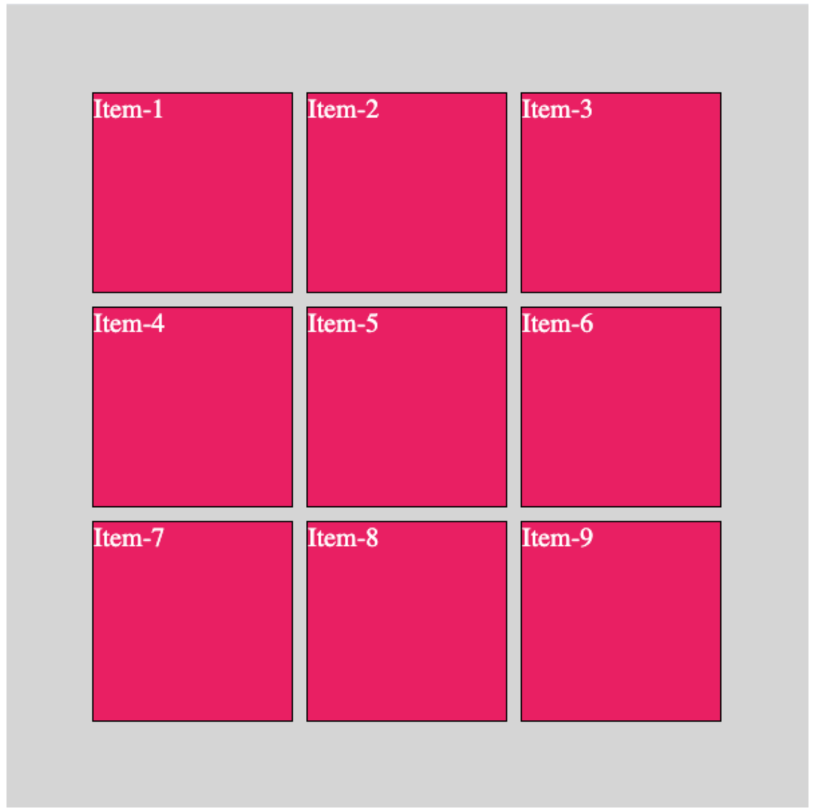
Grid Items – justify-self
align-self
place-self



```css
.just-align-content {
    width: 600px;
    height: 600px;
    display: grid;
    grid-template-columns: repeat(3, 150px);
    grid-template-rows: repeat(3, 150px);
    grid-gap: 10px;
    color: white;
    background-color: #d5d5d5;
    padding: 10px;

    justify-content: center;
    align-content: center;

    justify-items: stretch;
    align-items: stretch;
}

.just-align-content div {
    background-color: #E91E63;
    font-size: 20px;
    border: 1px solid black;
}
```

```css
.one {
    justify-self: strectch;
    align-self: stretch;
}
```

```html
<div class="just-align-content">
    <div class="one">Item-1</div>
    <div>Item-2</div>
    <div>Item-3</div>
    <div>Item-4</div>
    <div>Item-5</div>
    <div>Item-6</div>
    <div>Item-7</div>
    <div>Item-8</div>
    <div>Item-9</div>
</div>
```



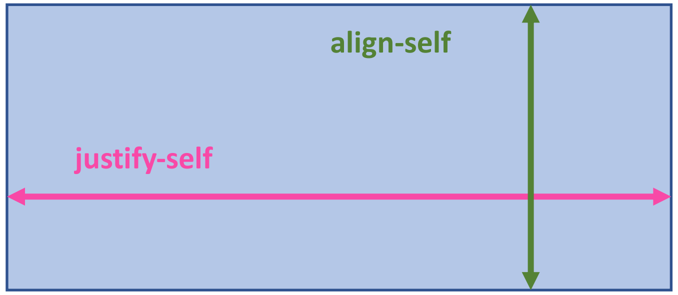
LAYOUT

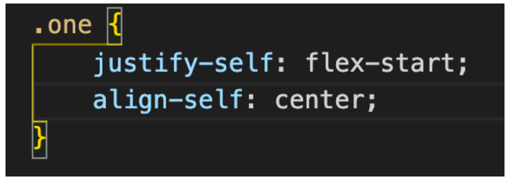
Aligning Individual Grid Items

Grid Items – justify-self
align-self
place-self

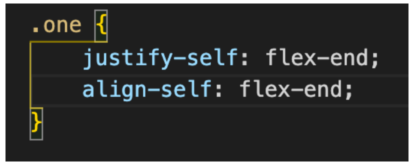
place-self: align-self justify-self;

Individual grid item

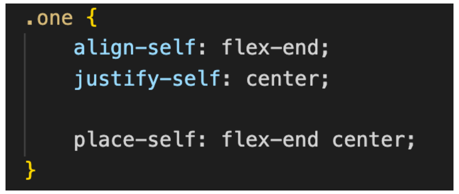




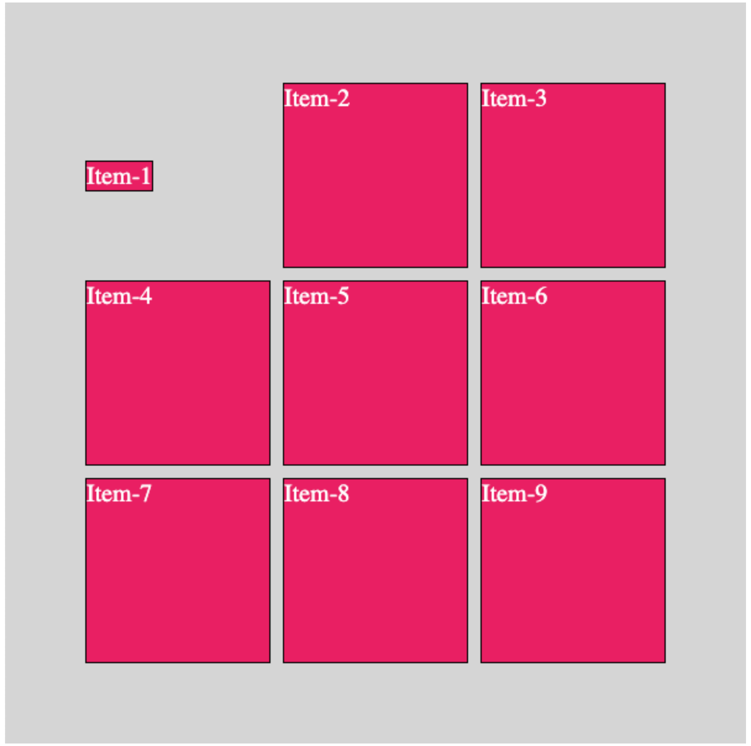
```
.one {
    justify-self: flex-start;
    align-self: center;
}
```

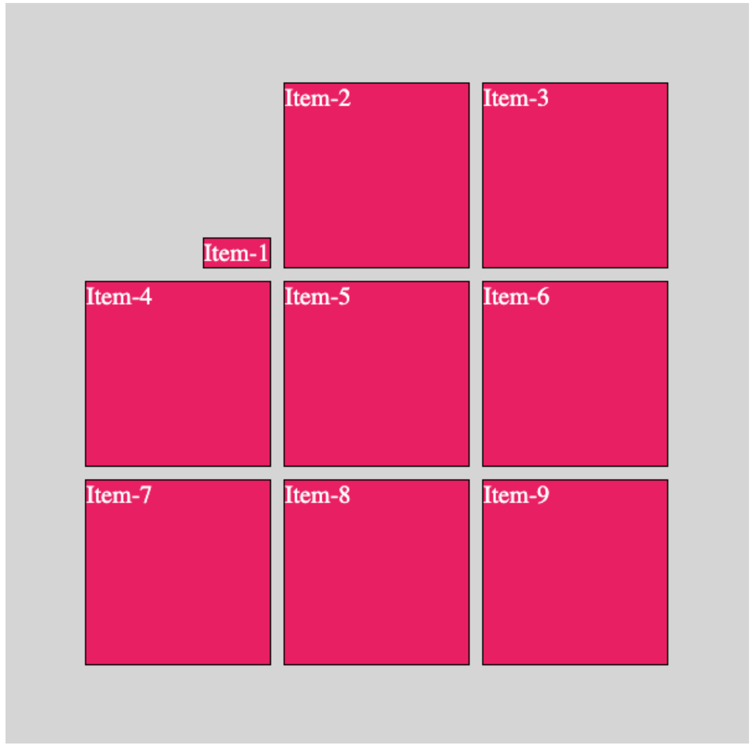


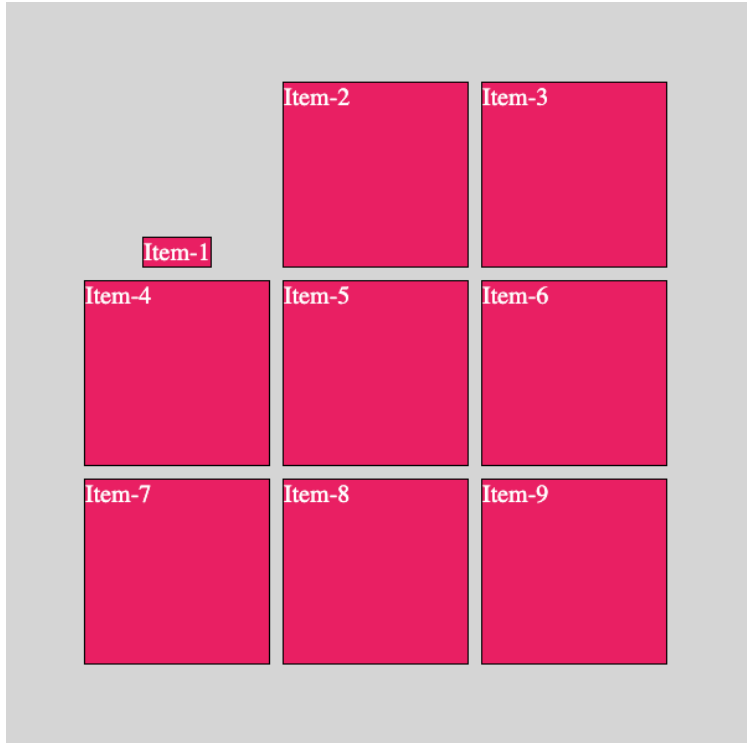
```
.one {
    justify-self: flex-end;
    align-self: flex-end;
}
```



```
.one {
    align-self: flex-end;
    justify-self: center;

    place-self: flex-end center;
}
```


Item-1
Item-2
Item-3
Item-4
Item-5
Item-6
Item-7
Item-8
Item-9


Item-1
Item-2
Item-3
Item-4
Item-5
Item-6
Item-7
Item-8
Item-9


Item-1
Item-2
Item-3
Item-4
Item-5
Item-6
Item-7
Item-8
Item-9