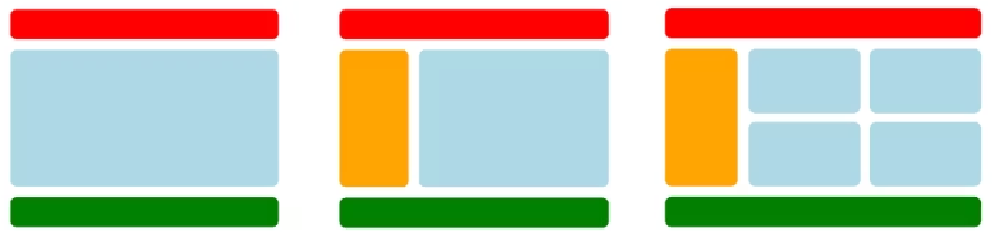
CSS: Grids

Lec-6

What is a Grid Layout Module ?

- The Grid Layout Module offers a **grid-based layout system,** with **rows and columns**.
- It can be useful to **layout entire web pages.**
- The Grid Layout Module **allows developers to easily create complex web layouts**.
- The Grid Layout Module makes it **easy to design** a **responsive layout structure**, without using **float** or **positioning**.



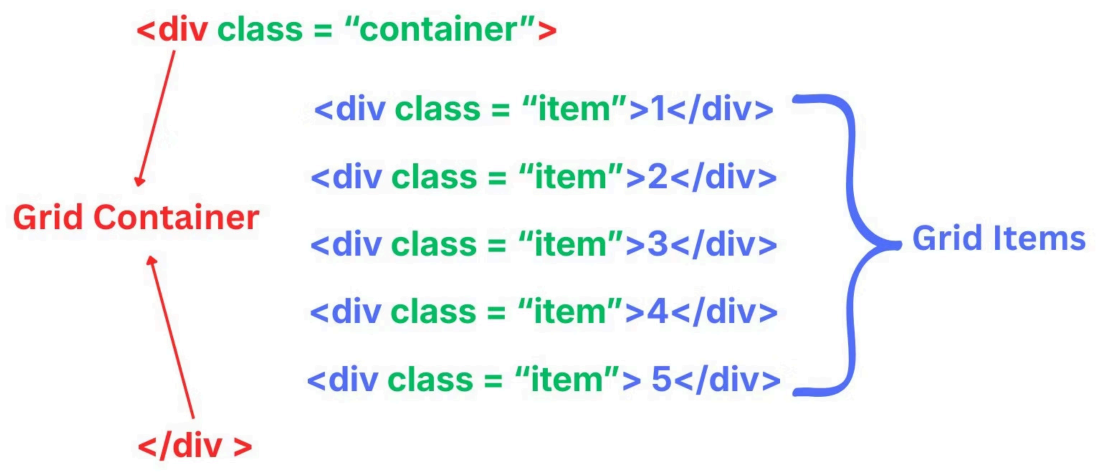
Flexbox Vs Grid

Flexbox	Grid
○ Mostly used to **position items in one-dimensional layout.**	○ Useful for **positioning items in two-dimensional layout.**
○ Items can be **positioned or moved across row or column at once.**	○ Items can be **positioned or moved across both row and column together.**

Creating a Grid

- To set up a grid, you need to have both:

1. **grid container -** It is an element on a page which contains **grid items.**
2. **grid-items -** All direct **child elements** of a grid container.



```
<div class = "container">
    <div class = "item">1</div>
    <div class = "item">2</div>
    <div class = "item">3</div>
    <div class = "item">4</div>
    <div class = "item"> 5</div>
</div >
```

- Any element can be grid container; **child** of grid container will **change size and location** in **response to the size and position of their parent container**.

Grid Container Properties

1. display: grid/inline-grid

- Any element **becomes an grid container** when it's **display** property is set to **grid** or **inline-grid.**
- We use **display: grid** for making an element as a **block-level grid container.**
- We use **display: inline-grid** for making an element as **inline grid container.**
- **Eg:**

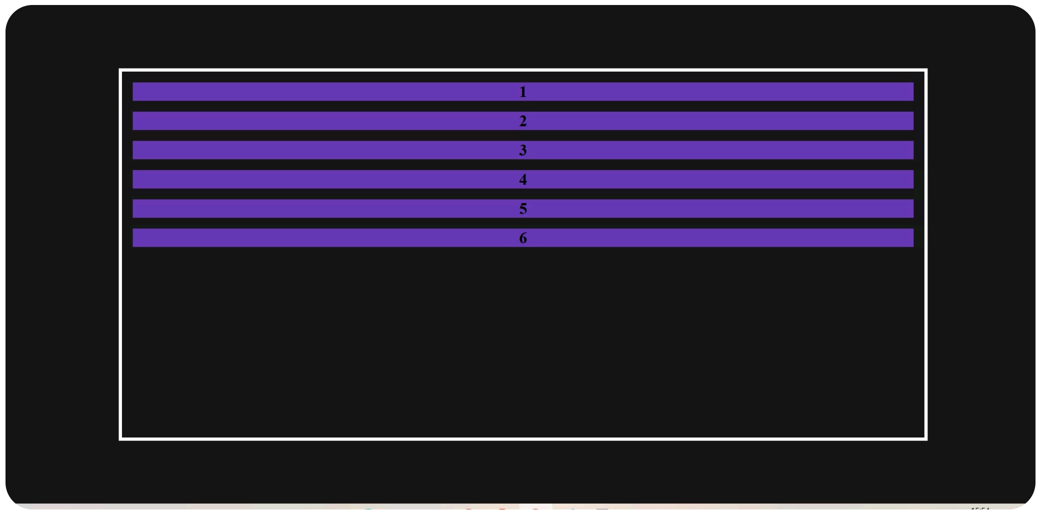
html

```
<div class="container">
  <div class="items">1</div>
  <div class="items">2</div>
  <div class="items">3</div>
  <div class="items">4</div>
  <div class="items">5</div>
  <div class="items">6</div>
</div>
```

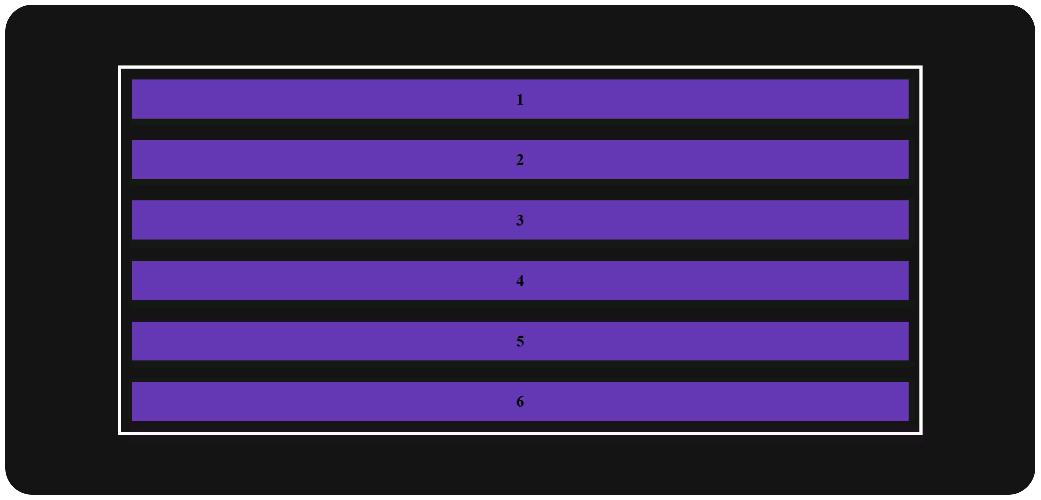
css

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

before



after



Grid Container Properties

2. grid-template-columns

- This property defines the **number of columns** and their **width** inside the grid container.
- It takes **space separated list** as a value, where each value defines the width of the **respective column.**

Common Values:

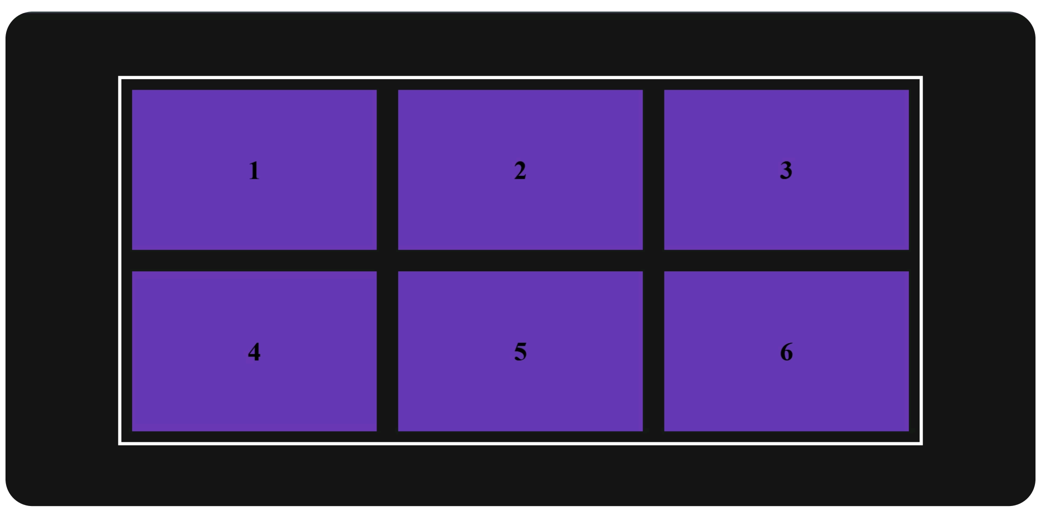
(i) auto: To create a grid layout with 3 columns of equal width, use the **"auto"** keyword.

Eg:

css

```
.container{
  display: grid;
  grid-template-columns: auto auto auto;
}
```

output



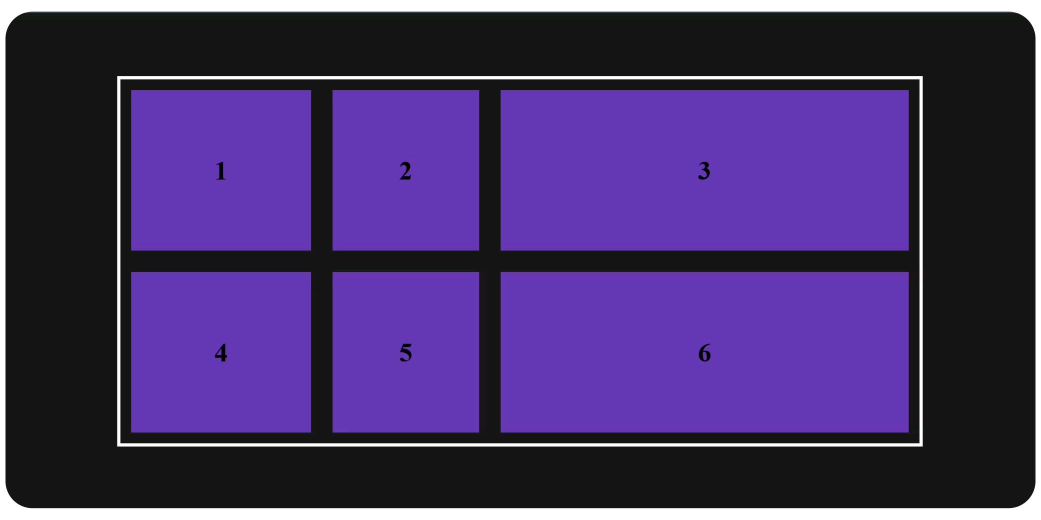
(ii) mix-width: To create a grid layout with 3 columns: **2 columns with a fixed width**, and **1 column with size "auto".**

Eg:

css

```
.container{
  display: grid;
  grid-template-columns: 300px 250px auto;
}
```

output



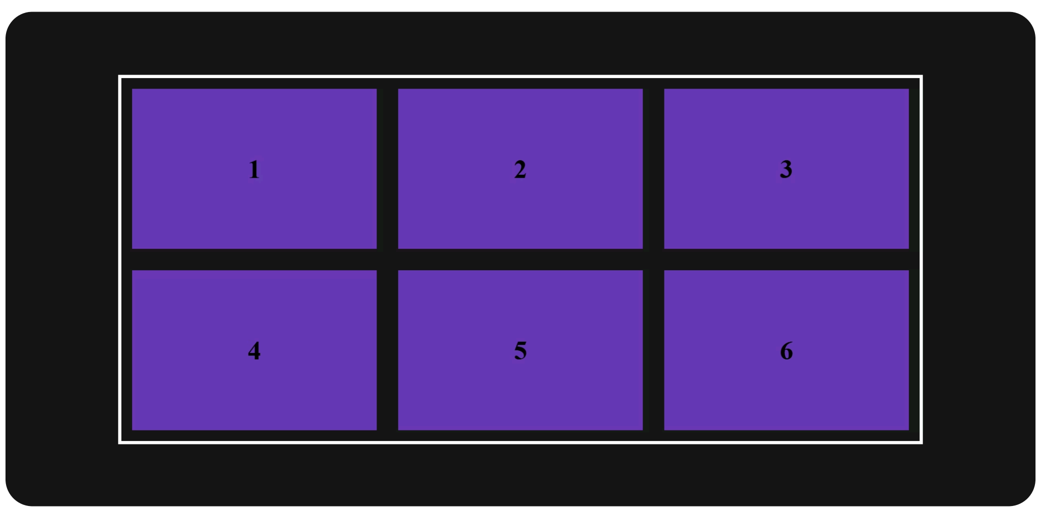
(iii) fr-unit: "fr" stands for **fractional unit** and **1fr = 33.33%** of a container width. **1fr** will take **1 fraction of available space** and similarly **2fr** will take **2 fraction of available space.**

Eg:

css

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

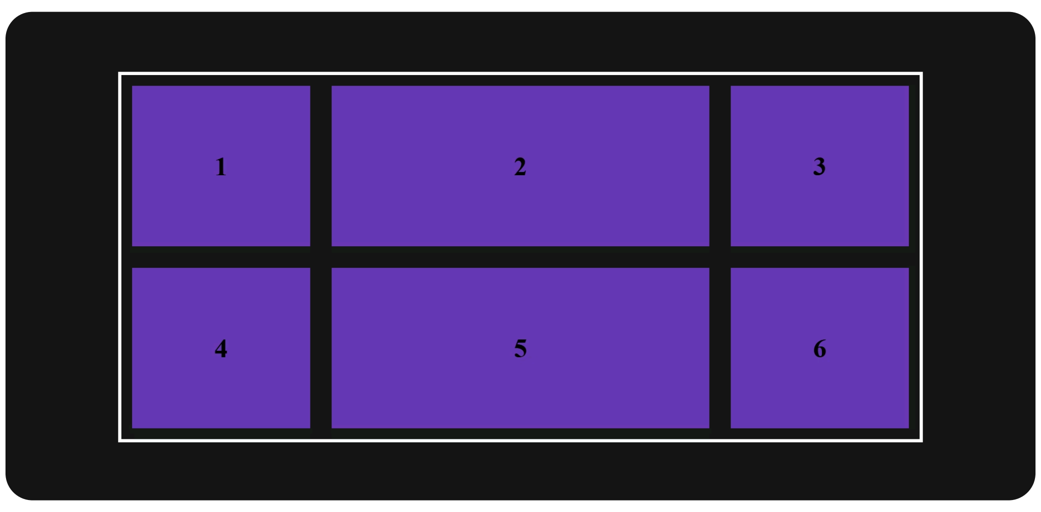
output



css

```
.container{
   display: grid;
   grid-template-columns: 1fr 2fr 1fr;
}
```

output



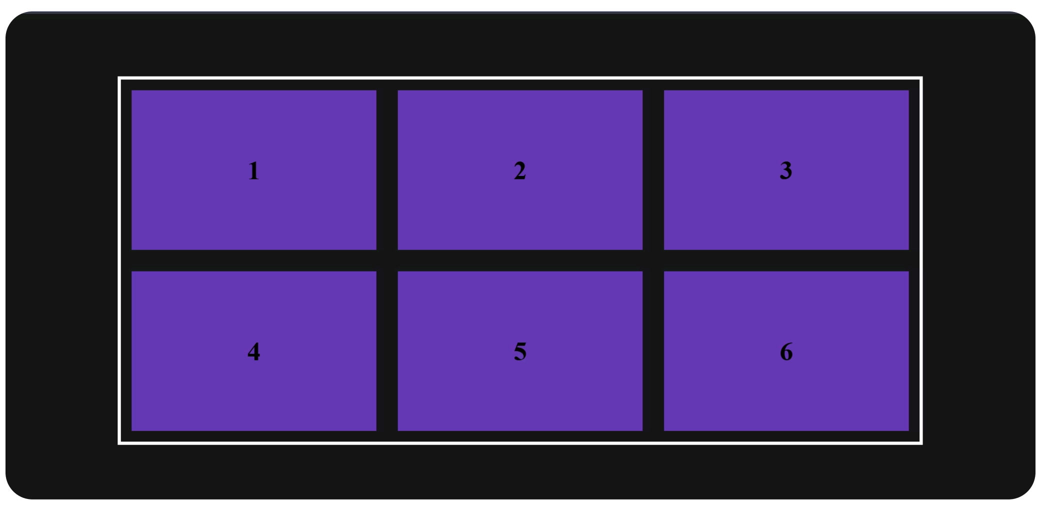
(iv) repeat(): This function is used to repeat a set of columns or rows in a grid.

Eg:

css

```
.container{
   display: grid;
   grid-template-columns: repeat(3, 1fr);
}
```

output



Grid Container Properties

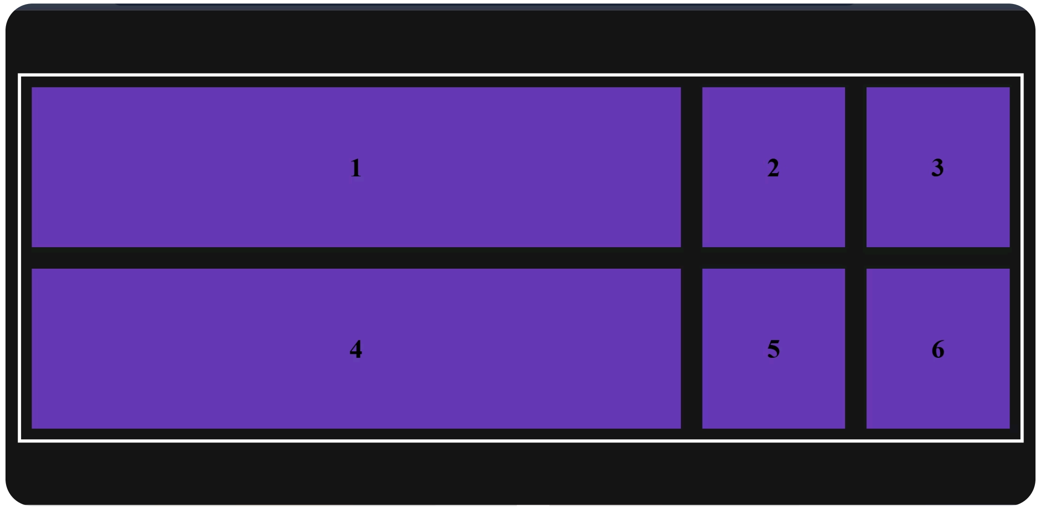
(v) minmax(): This function is used to define a size-range greater than or equal to a min value and less than or equal to a max value.

Eg:

css

```
.container{
 display: grid;
 grid-template-columns: minmax(200px, 1000px) auto
auto;
}
```

output



In above property the first column will be at least 200px wide and can grow up to 1000px.

Grid Container Properties

3. **grid-template-rows**

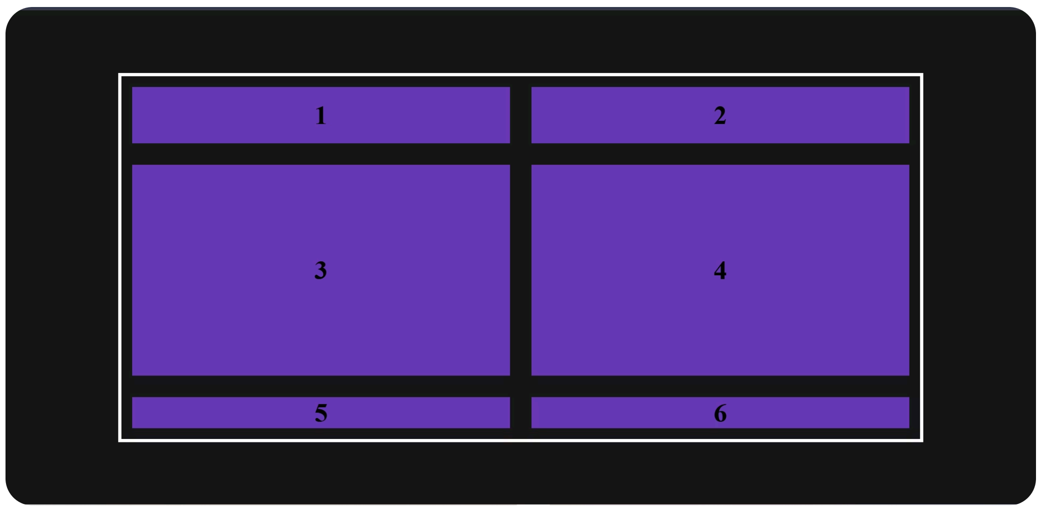
- This property defines the **number of rows** and their **height** inside the grid container.
- It takes **space separated list** as a value, where each value defines the width of the **respective column.**

Eg:

css

```
.container{
      display: grid;
      grid-template-columns: auto auto;
      grid-template-rows: 1fr 3fr;
}
```

output



Notice that the **first row will have height of 1fr**, **second row will have height 3fr** and the **third subsequent row will have the auto height based on the remaining space inside the grid container**.

Grid Template

- It is a **shorthand property** for **grid-template-rows** and **grid-template-columns.**
- **Syntax:**

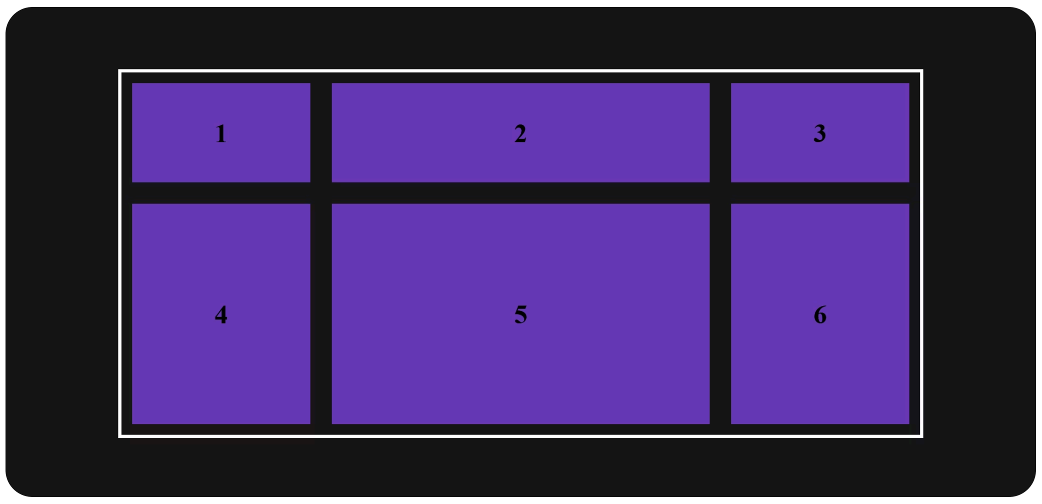
grid-template: grid-template-rows / grid-template-columns

Eg

css

```
.container{
      display: grid;
     grid-template: 1fr 2fr / 1fr 2fr 1fr;
}
```

output



In above example, **we generated a grid layout of 3 columns** with **middle column of width 2fr** and **the first and last column with 1fr of width respectively** and of **2 rows with height of first row as 1fr and that of second row in 2fr**.

Grid Container Properties

Grid Gaps / Gutters

- The **space between the rows and columns in a grid container** are called ***gaps* (or *gutters*)**.
- The gaps are **created between the grid rows and columns, not on the outer edges** of the grid container.
- The **default size of the gap is 0**, which means that **there are no spacing between grid items**.
- Grid gaps contain following properties:

1. **column-gap**
2. **row-gap**
3. **gap**

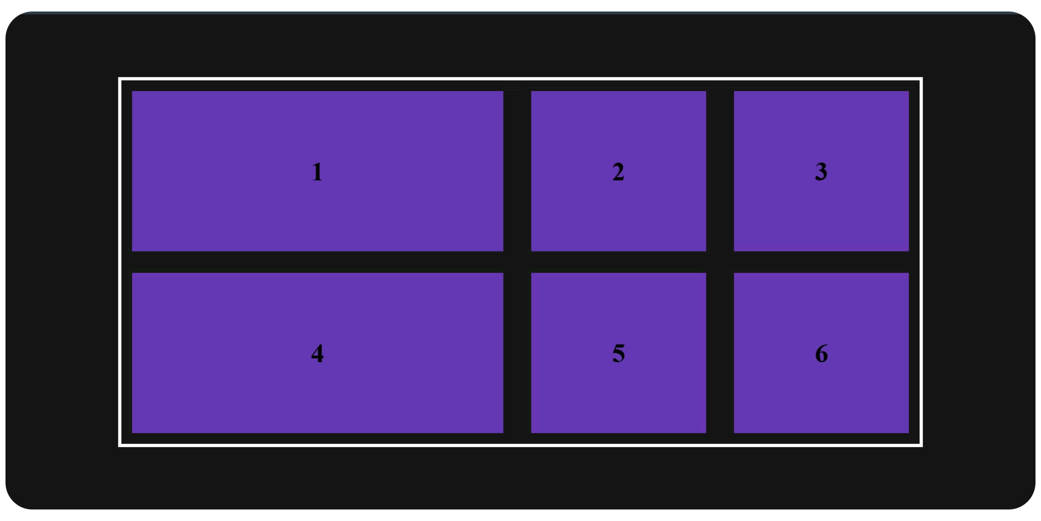
4. column-gap

- This property **specifies the gap between the columns in a grid**.
- **Eg:**

css

```
.container{
      display: grid;
      grid-template-columns: 2fr 1fr 1fr;
      column-gap: 10px;
}
```

output



Above property will **specify a 10 pixels gap between the grid columns**.

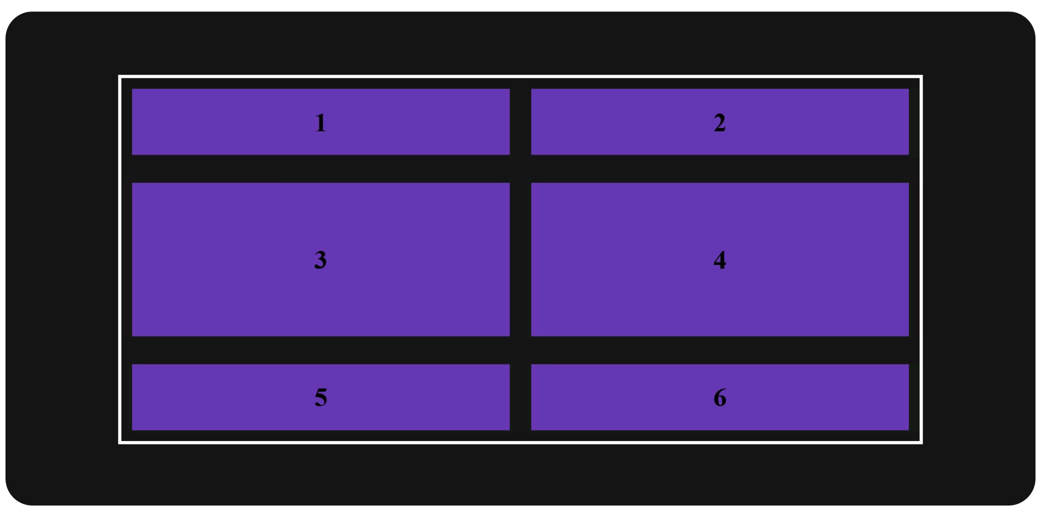
5. row-gap

- This property **specifies the gap between the rows in a grid.**
- **Eg:**

css

```
.container{
    display: grid;
    grid-template: 1fr 2fr 1fr / auto auto;
    row-gap: 10px;
}
```

output



Above property will **specify a 10 pixels gap between the grid rows**.

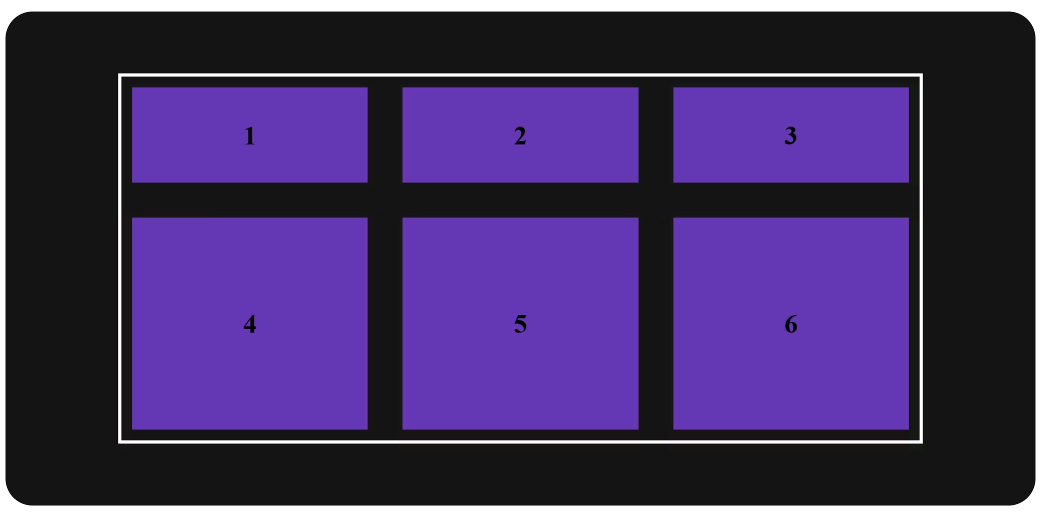
6. gap

- This is a **shorthand property** for **row-gap** and **column-gap.**
- If a **single value** is provided, **it applies the same gap to both rows and columns.**
- If **two values** are provided, the **first value** sets the **row-gap** and **second value** sets the **column-gap.**
- **Eg:**

css

```
.container{
    display: grid;
    grid-template: 1fr 2fr / auto auto auto;
    gap: 20px;
}
```

output

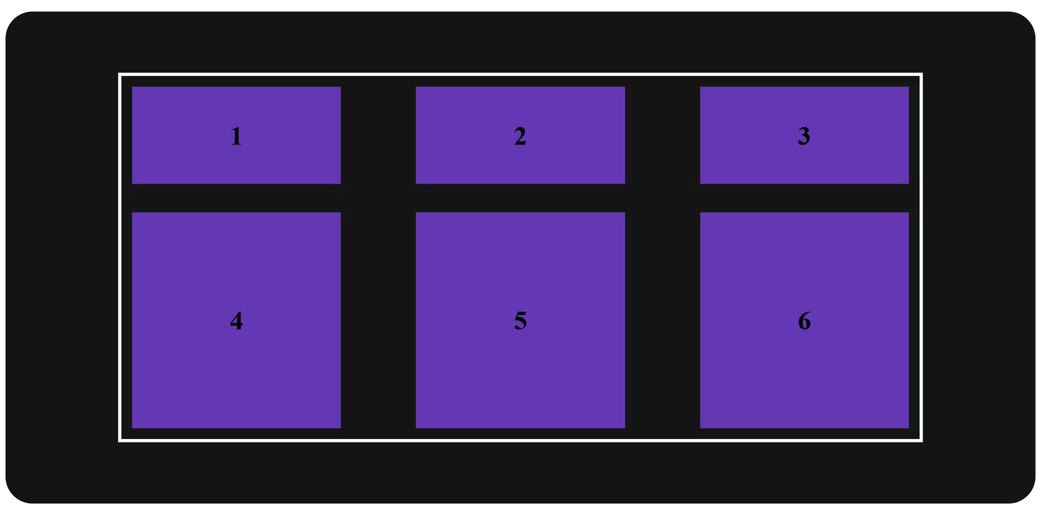


In above example **the gap between both grid row and columns will be 20px.**

css

```
.container{
    display: grid;
    grid-template: 1fr 2fr / auto auto auto;
    gap: 10px 80px;
}
```

output



In above example **the gap between both grid rows will be 10px and columns will be 80px.**

Grid Container Properties

7. justify-content:

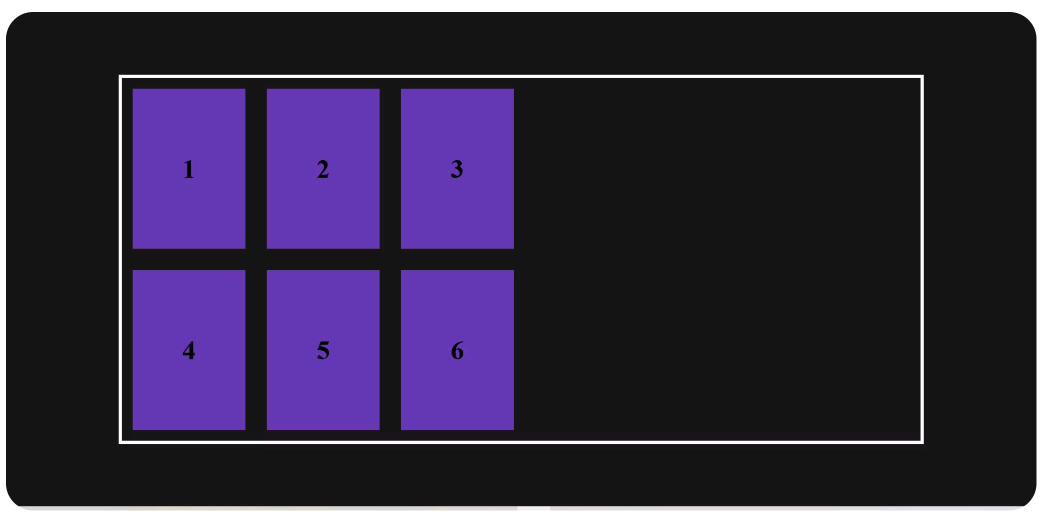
- This property is **used to align the grid content when it does not use all available space on the main-axis (horizontally)**.
- This property can have following values:

1. **space-evenly**
2. **space-between**
3. **space-around**
4. **start**
5. **end**
6. **center**

css

```
.container{
     display: grid;
     grid-template-columns: repeat(3, 200px);
     justify-content: start;
}
```

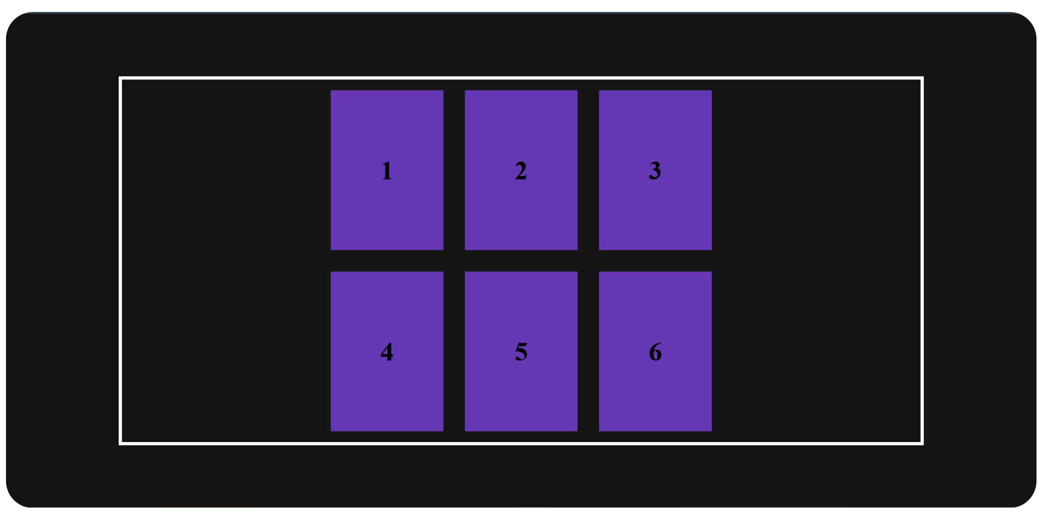
output



css

```
.container{
     display: grid;
     grid-template-columns: repeat(3, 200px);
     justify-content: center;
}
```

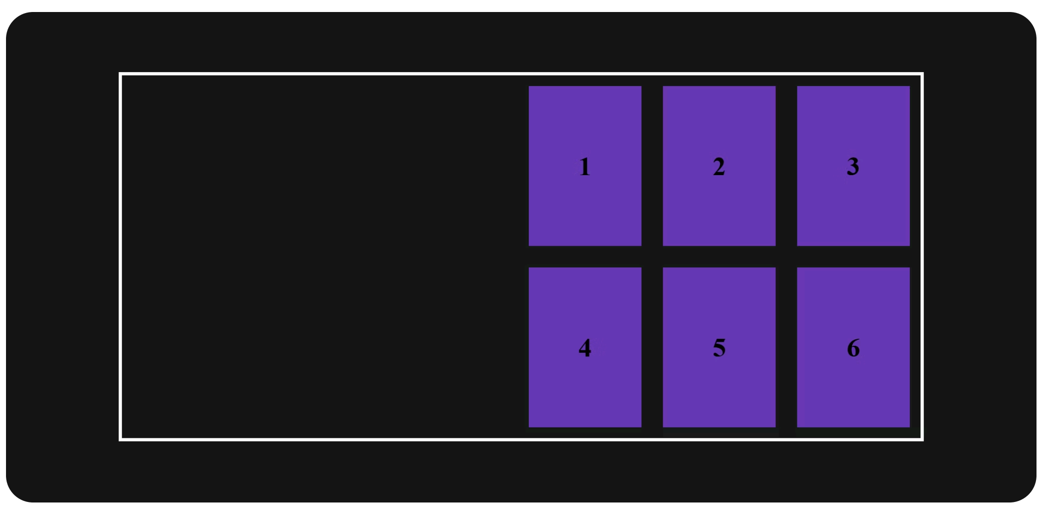
output



css

```
.container{
     display: grid;
     grid-template-columns: repeat(3, 200px);
     justify-content: end;
}
```

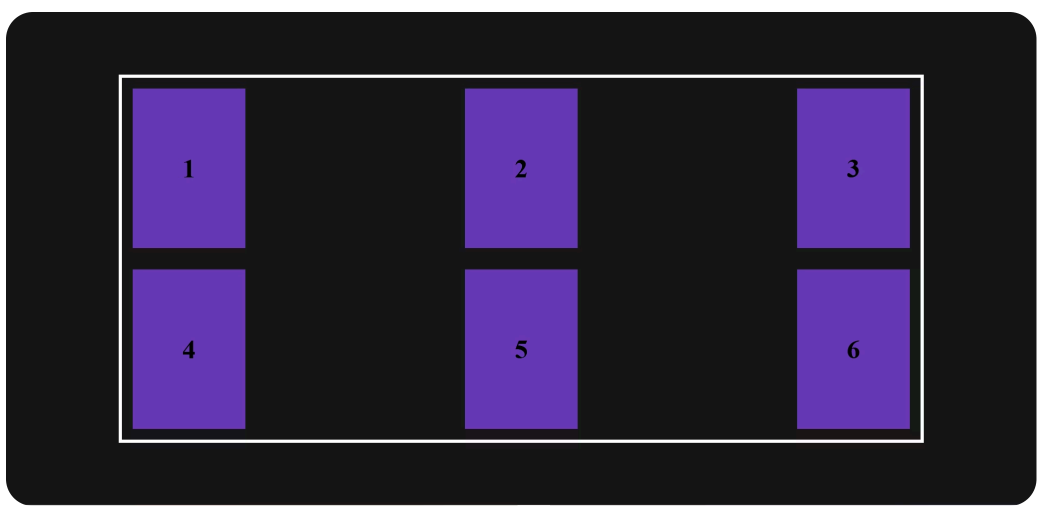
output



css

```
.container{
     display: grid;
     grid-template-columns: repeat(3, 200px);
     justify-content: space-between;
}
```

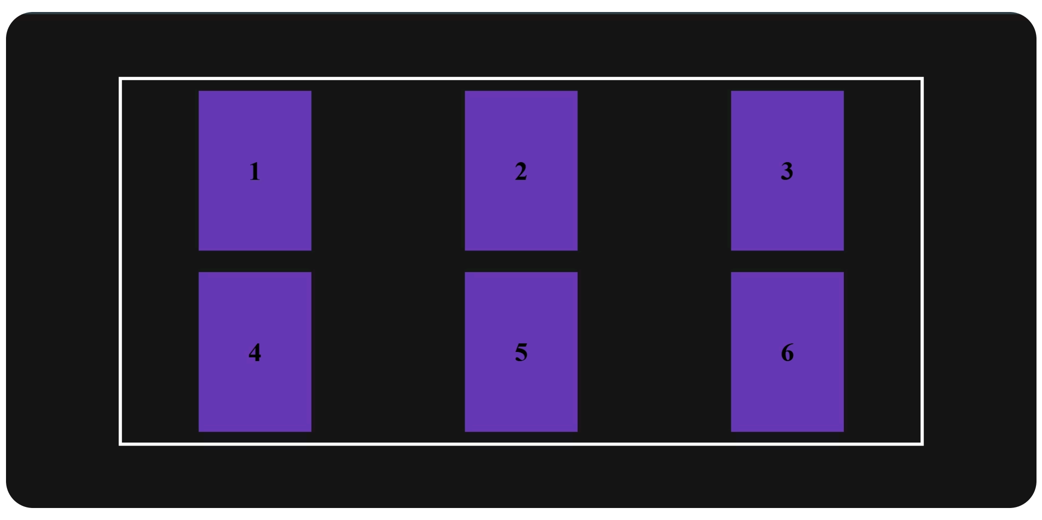
output



css

```
.container{
     display: grid;
     grid-template-columns: repeat(3, 200px);
     justify-content: space-around;
}
```

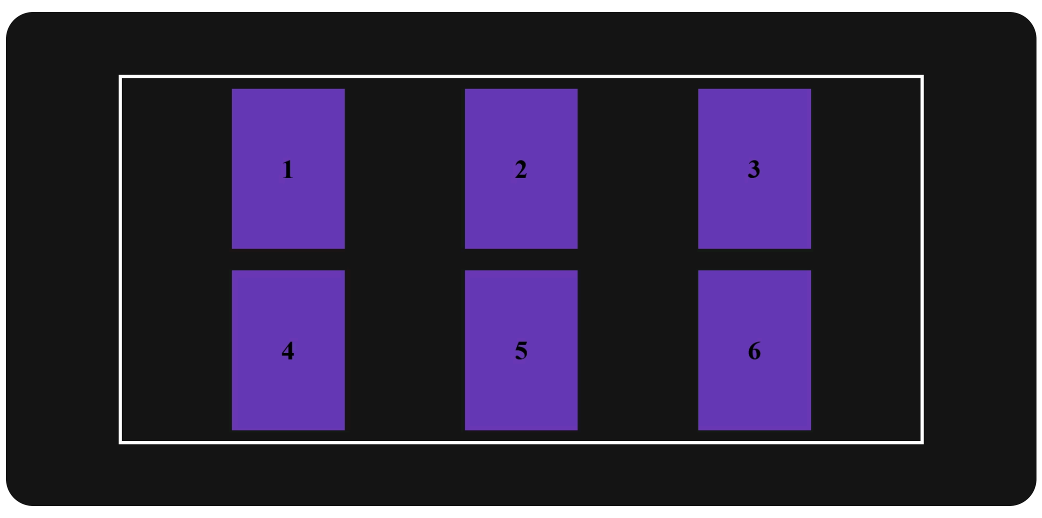
output



css

```
.container{
     display: grid;
     grid-template-columns: repeat(3, 200px);
     justify-content: space-evenly;
}
```

output



Grid Container Properties

8. align-content:

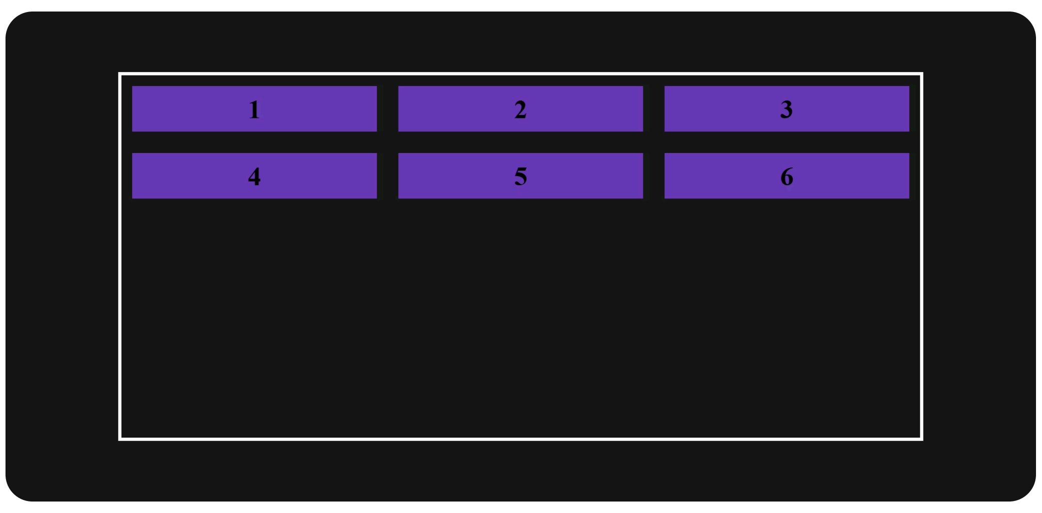
- This property is **used to align the grid content when it does not use all available space on the cross-axis (vertical)**.
- This property can have following values:

1. **space-evenly**
2. **space-between**
3. **space-around**
4. **start**
5. **end**
6. **center**

css

```
.container{
    display: grid;
    grid-template-columns: repeat(3, 200px);
    align-content: start;
}
```

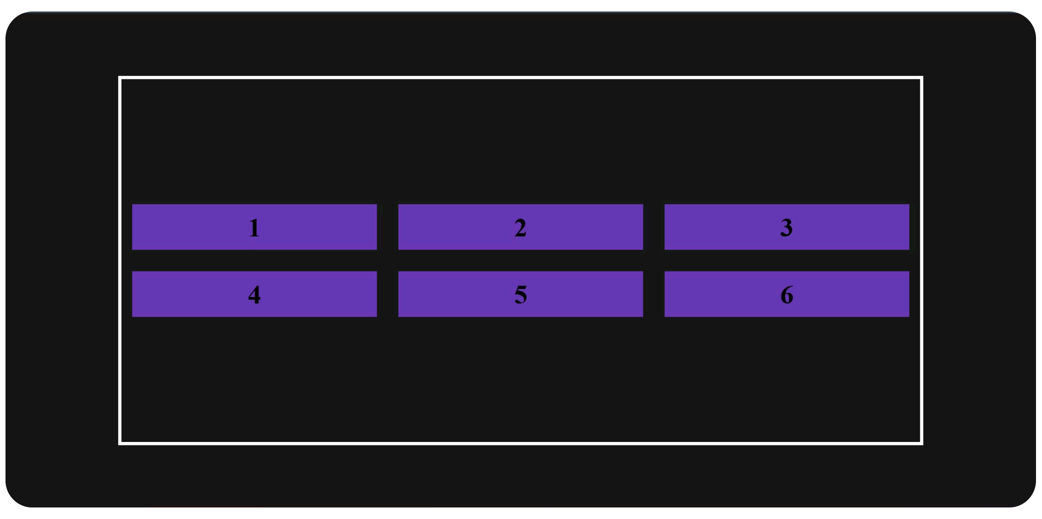
output



css

```
.container{
    display: grid;
    grid-template-columns: repeat(3, 200px);
    align-content: center;
}
```

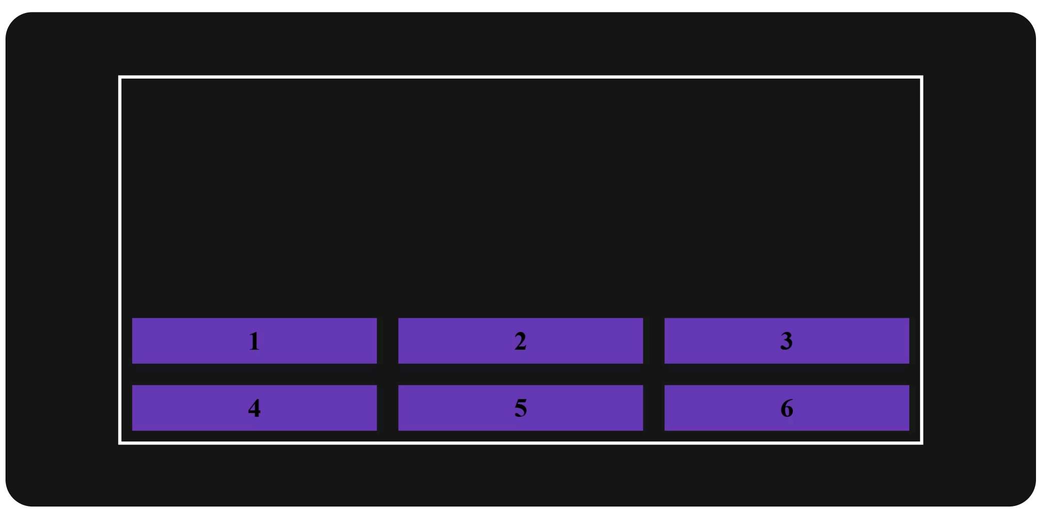
output



css

```
.container{
    display: grid;
    grid-template-columns: repeat(3, 200px);
    align-content: end;
}
```

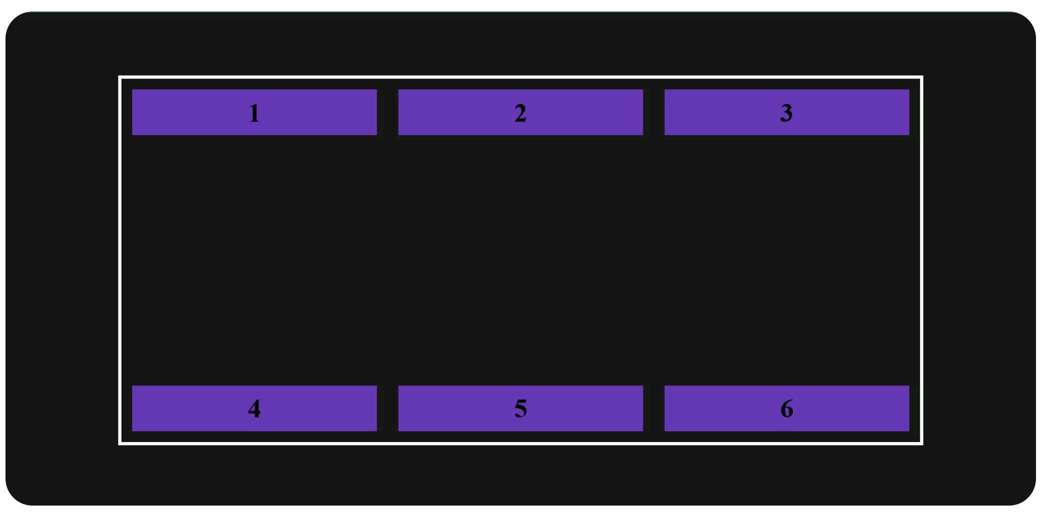
output



css

```
.container{
    display: grid;
    grid-template-columns: repeat(3, 200px);
    align-content: space-between;
}
```

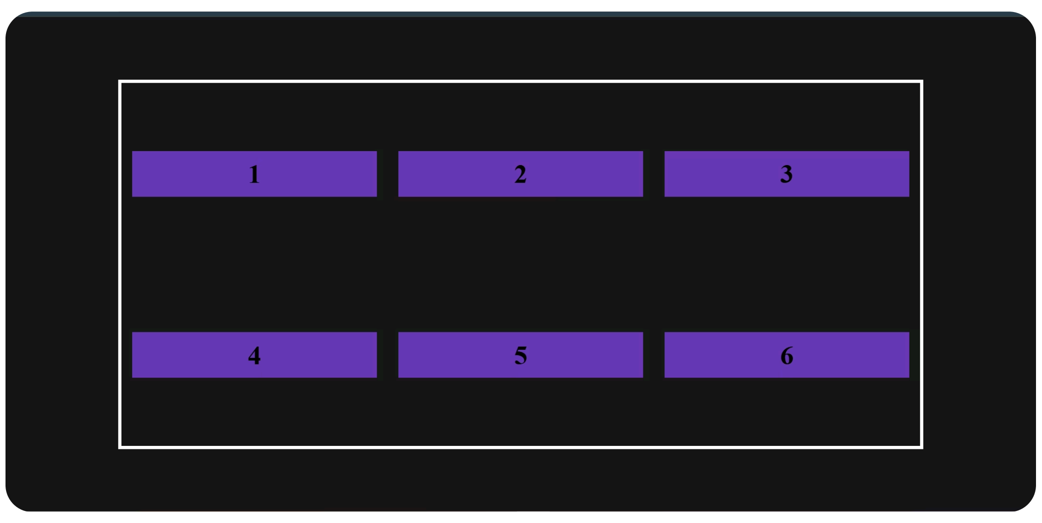
output



css

```
.container{
    display: grid;
    grid-template-columns: repeat(3, 200px);
    align-content: space-around;
}
```

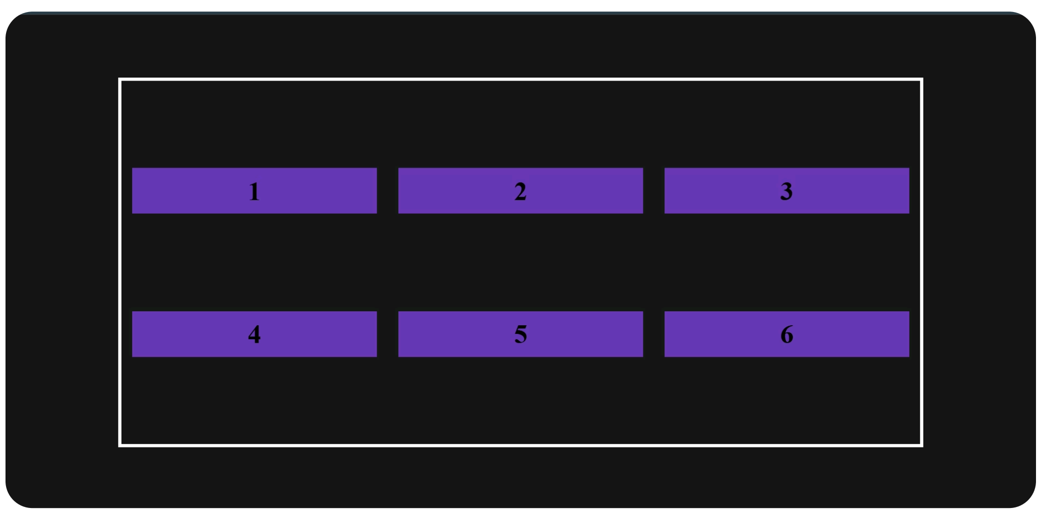
output



css

```
.container{
    display: grid;
    grid-template-columns: repeat(3, 200px);
    align-content: space-evenly;
}
```

output



Grid Container Properties

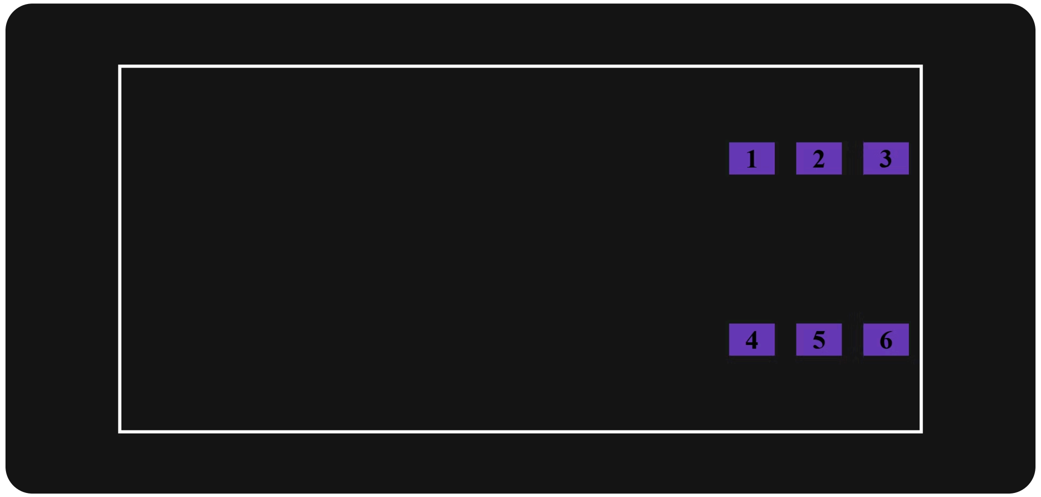
9. **place-content:**

- This property is a shorthand property for- **align-content** and **justify-content.**
- If **single value** is provided, it is applicable to both **align-content** and **justify-content.**
- If **two values** are provided, **first value** get's applicable to **align-content** and **second value** get's applicable to **justify-content.**
- **Eg:**

css

```
.container{
     display: grid;
     grid-template-columns: repeat(3, 100px);
     grid-template-rows: 80px 80px;
     place-content: space-around flex-end;
}
```

output



Made with GAMMA

Grid Items Properties

1. **grid-column-start** and **grid-column-end:**

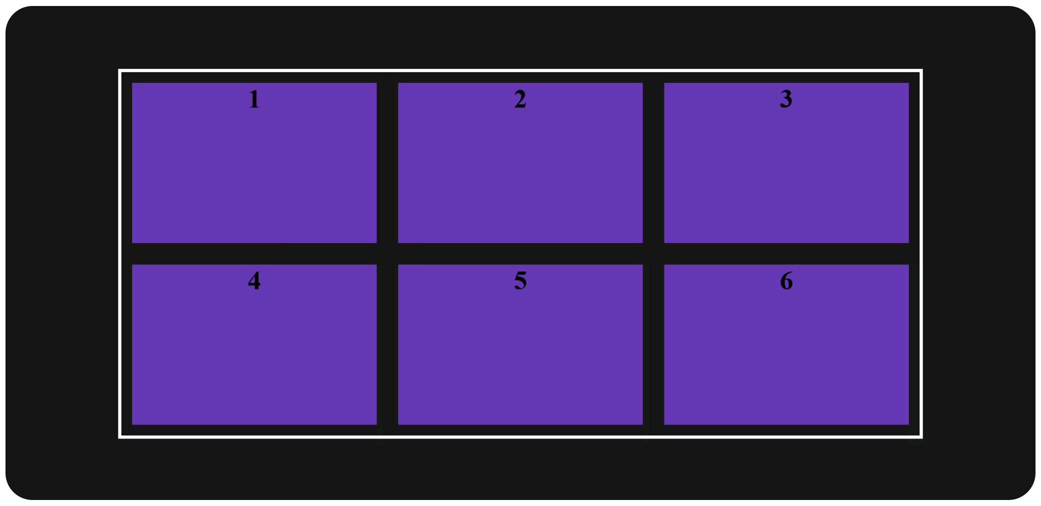
- **grid-column-start** specifies on which column-line the grid item will start.
- **grid-column-end** specifies on which column-line the grid item will end.
- **Eg:**

css

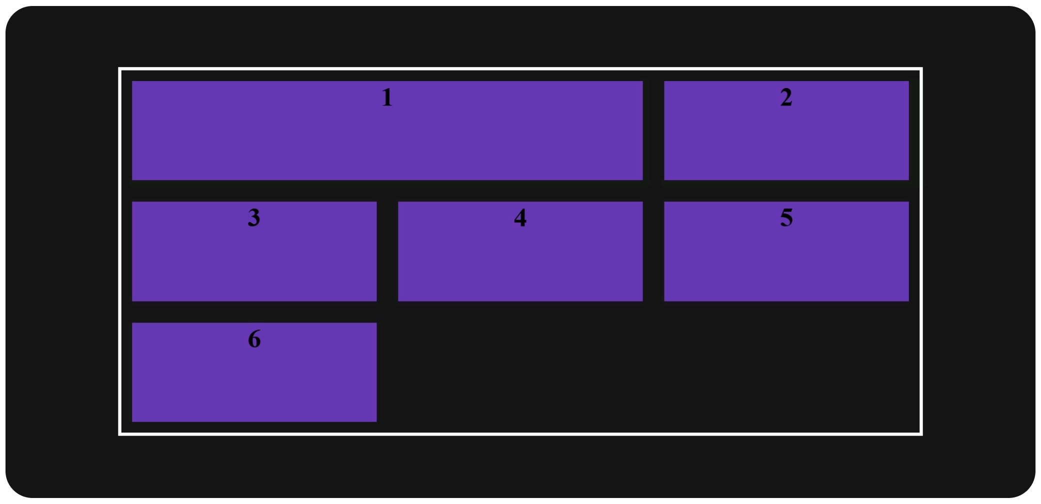
```
.container {
    display: grid;
    grid-template-columns: auto auto auto;
}
.container :first-child {
    grid-column-start: 1;
    grid-column-end: 3;
}
```

output

before



after



Note:

grid-column : It is a shorthand property for **grid-column-start** and **grid-column-end** properties.

Eg:

grid-column: 1 /span 2; => This means that the grid items will start at first column and will span 2 columns further.

Grid Items Properties

2. **grid-row-start** and **grid-row-end:**

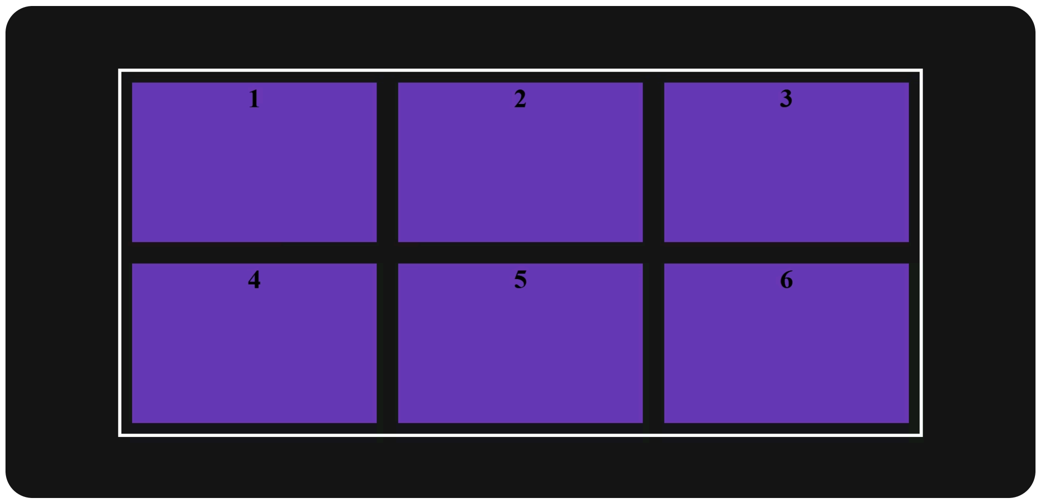
- **grid-row-start:** specifies on which row-line the grid item will start.
- **grid-row-end:** specifies on which row-line the grid item will end.
- **Eg:**

css

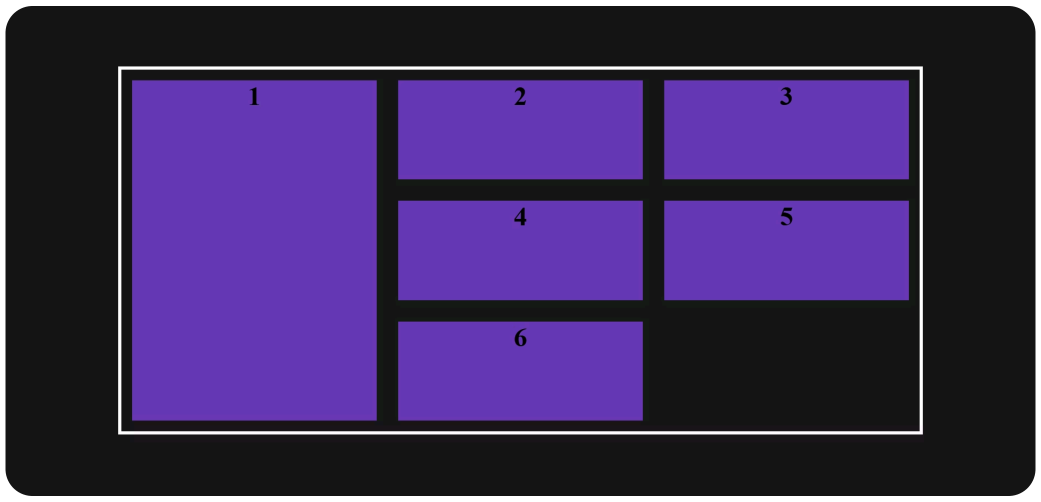
```
.container {
  display: grid;
  grid-template-columns: auto auto auto;
}
.container :first-child {
  grid-row-start: 1;
  grid-row-end: 4;
}
```

output

before



after



Note:

grid-row : It is a shorthand property for **grid-row-start** and **grid-row-end** properties.

Eg:

grid-row: 1 /span 2; => This means that the grid items will start at first row and will span 2 rows further.

Grid Items Properties

3. align-self:

- This property is used to **align a grid item within its grid cell in the block direction.**
- It takes the following values:

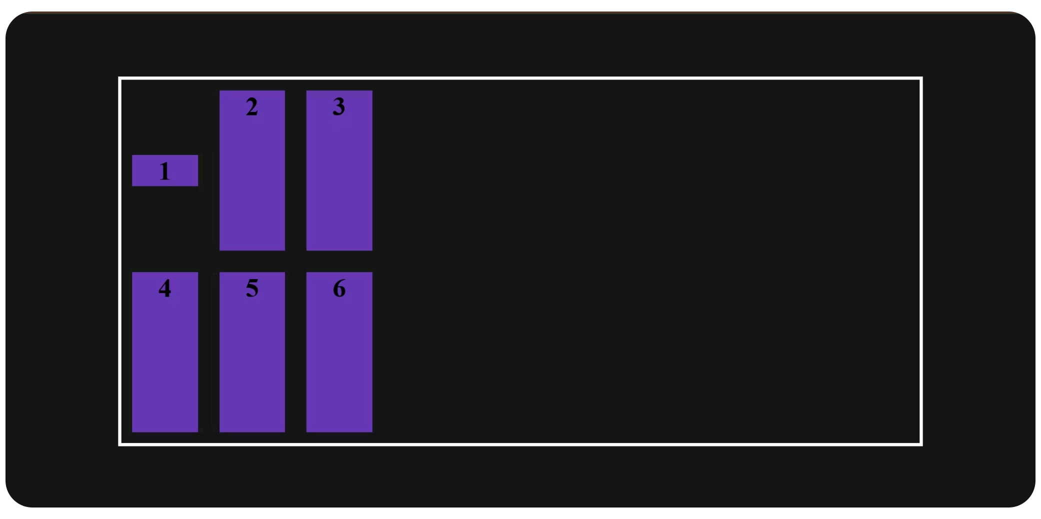
1. **auto**
2. **normal**
3. **stretch**
4. **start**
5. **end**
6. **center**

css

```
.container{
    display: grid;
    grid-template-columns: 130px 130px 130px;
}

.container :first-child {
     align-self: center;
}
```

output

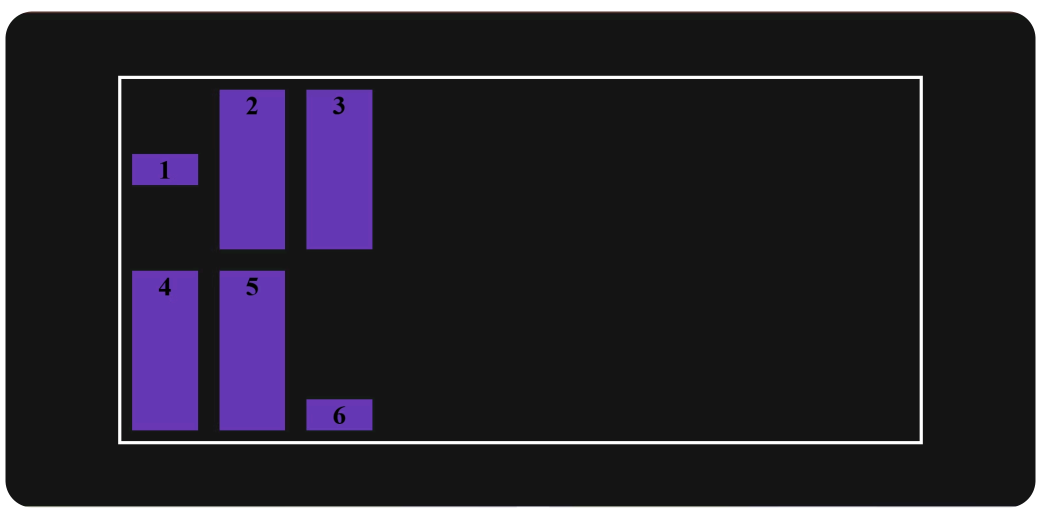


css

```
.container{
    display: grid;
    grid-template-columns: 130px 130px 130px;
}

.container :first-child {
     align-self: center;
}
.container :last-child {
     align-self: flex-end;
   }
```

output



4. justify-self:

- This property is used to **align a grid item within its grid cell in the inline direction.**
- It takes the following values

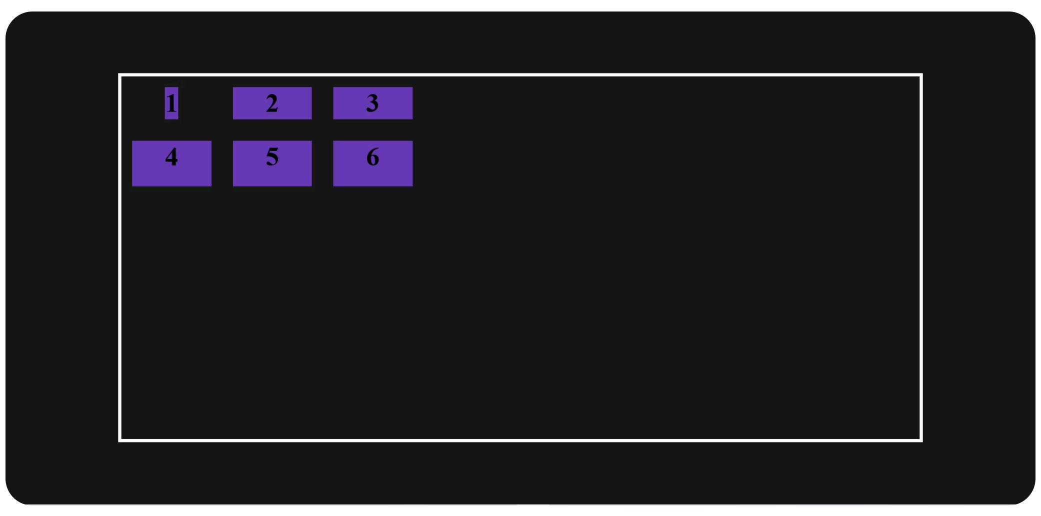
1. **auto**
2. **normal**
3. **stretch**
4. **start**
5. **end**
6. **center**
7. **end**
8. **right**

css

```
.container{
    display: grid;
    grid-template-columns: 130px 130px 130px;
}

.container :first-child {
     justify-self: center;
}
```

output

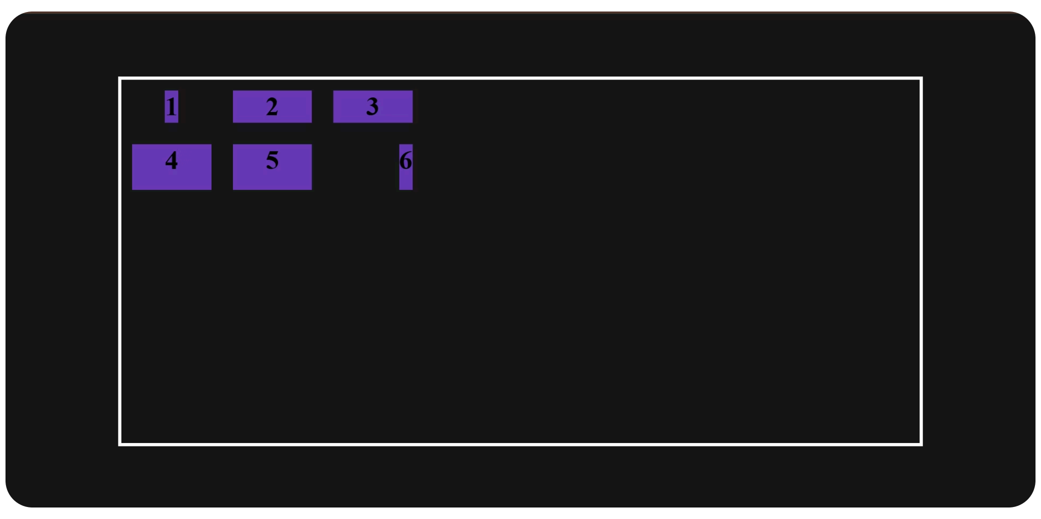


css

```
.container{
    display: grid;
    grid-template-columns: 130px 130px 130px;
}

.container :first-child {
     justify-self: center;
}
.container :last-child {
     justify-self: flex-end;
   }
```

output



Grid Items Properties

5. Order:

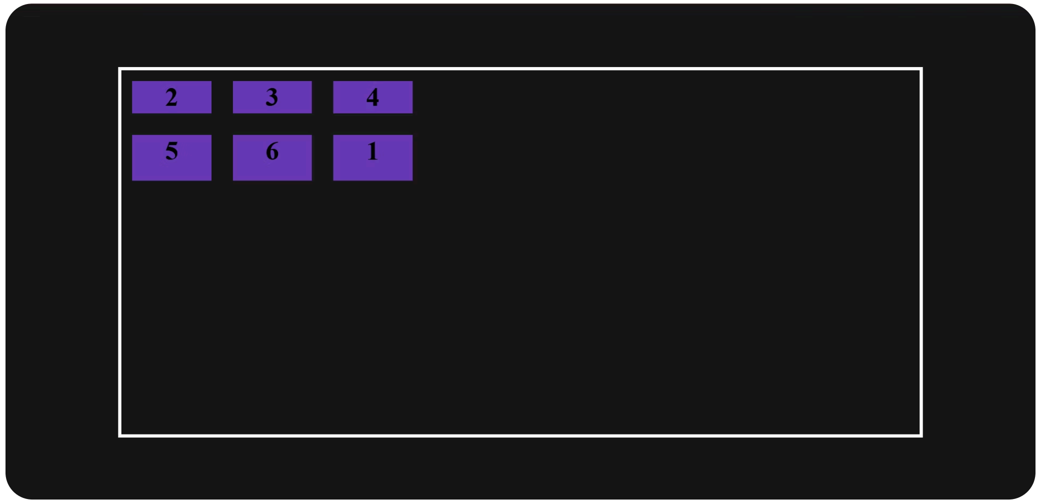
- This property can be **used to define the visual order of the grid items.**

css

```
.container{
    display: grid;
    grid-template: 80px 100px / 150px 150px 150px;
}

 .container :first-child {
    order: 6;
}
 .container :last-child {
    order: 4;
}
```

output



Made with GAMMA

Advanced Grid

6. grid-area:

- This is a **grid items property.**
- It is used to name the grid items and then reference it to **grid-template-area** property.

7. grid-template-areas:

- This is a **grid container property.**
- It specifies areas within the grid layout.
- Each area is defined within apostrophes.
- The named grid items in each area is defined inside the apostrophes, separated by a space.

Eg:

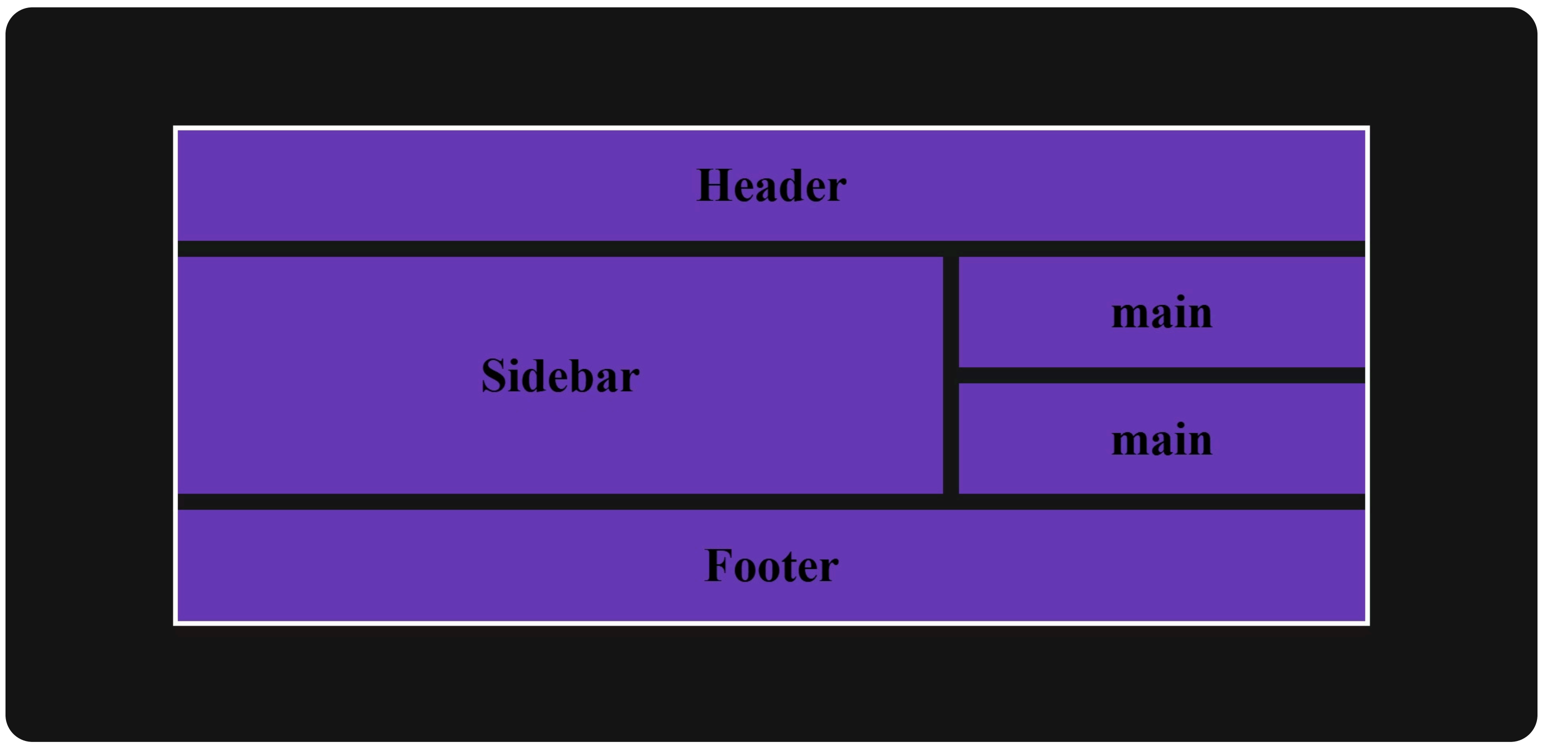
html

```
<div class="container">
   <div class="items" id="item1">Header</div>
   <div class="items" id="item2">Sidebar</div>
   <div class="items" id="item3">main</div>
   <div class="items" id="item4">main</div>
   <div class="items" id="item5">Footer</div>
</div>
```

css

```
.container{
        grid-template-areas:
          "myheader myheader myheader"
          "mymenu mymenu ..."
          "mymenu mymenu ..."
          "myfooter myfooter myfooter";
}
#item1 {
        grid-area: myheader;
 }
#item2 {
        grid-area: mymenu;
 }
#item5 {
        grid-area: myfooter;
 }
```

output


Header
Sidebar
main
main
Footer