

## Container

**Container** is a parent element which consists of child elements known as **items**.

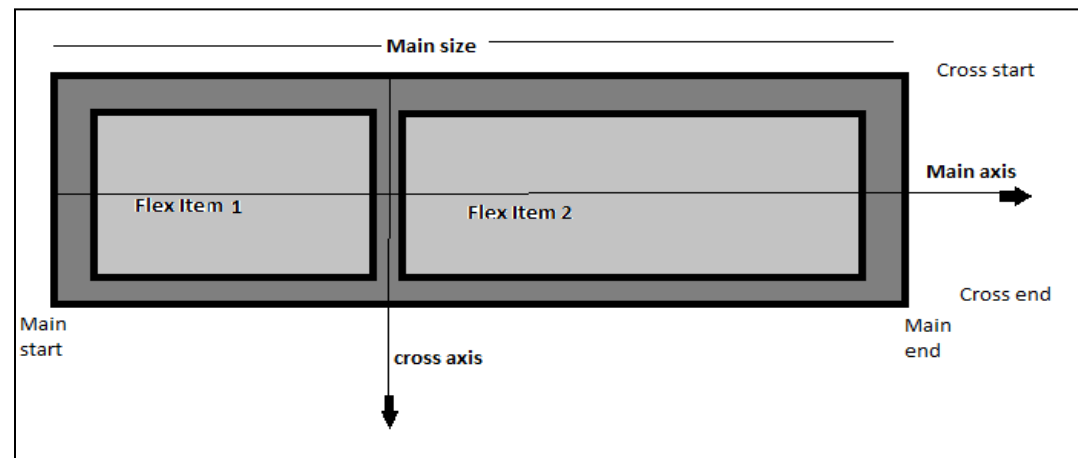
In the code shown, the div having “**class= container**” is a parent element of all the other div having “**class= item**” are items. So the outer div is a container and all the other div’s are items of that container.

## FLEXBOX LAYOUT

The main focus of Flexbox is to provide a more efficient way to layout, distribute space, and align items in a container irrespective of their sizes. Flexbox gives the ability to alter container items width and height, to best fill the available space.

Flex is a one dimensional layout model. Items will be laid out either toward the main axis (row) or cross axis (column) as shown in the image below.

```
<!DOCTYPE html>
<html>
  <head>
    <title>Flex</title>
  </head>
  <body>
    <div class="container">
      <div class="item"></div>
      <div class="item"></div>
      <div class="item"></div>
      <div class="item"></div>
      <div class="item"></div>
      <div class="item"></div>
      <div class="item"></div>
      <div class="item"></div>
    </div>
  </body>
</html>
```



## FlexBox Properties

1. Display
2. Flex Direction
3. Flex Wrap
4. Flex Flow
5. Justify Content
6. Align Items
7. Gap, Row gap, Column gap

1. **Display:** Display is a property which is used to show items in various ways.  
When **display : flex** is used, the items will be aligned with the main axis by default.

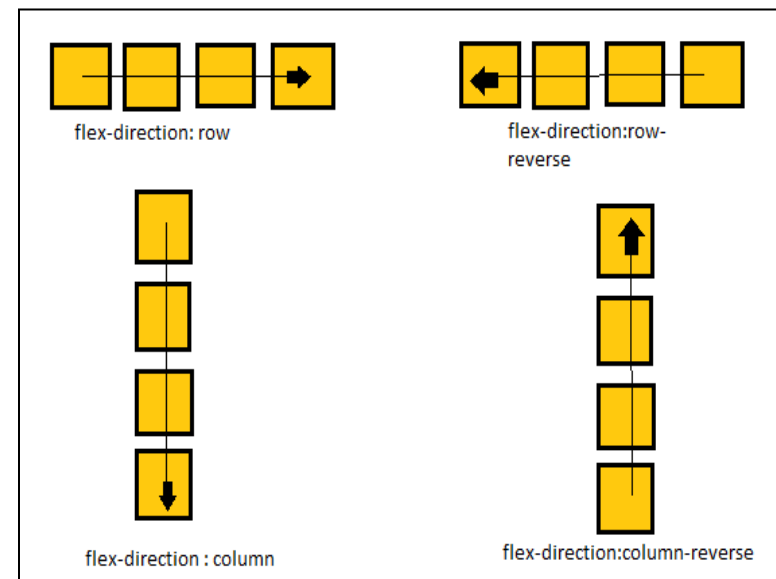
```
.container {
  Display: flex
}
```

2. **Flex Direction:** Defines the direction in which items can flow inside the container

There are four direction value

- I. **row:** display items from left to right
- II. **row-reverse:** display items from right to left
- III. **column:** display items from top to bottom
- IV. **column-reverse:** display item from bottom to top

```
.container {
  Flex-direction : row | row-reverse | column |
  column-reverse
}
```



**3. Flex wrap:** it wraps the items into single or multiple lines.

There are three wrap values

1. **wrap:** items will wrap onto multiple lines from top to bottom
2. **nowrap:** all the items will display on a single line
3. **wrap-reverse:** items will wrap onto multiple lines from bottom to top

```
.container {
  flex-wrap : wrap | nowrap |
  wrap-reverse
}
```

Flex Layout with Nowrap

1

2

3

4

5

6

7

8

9

10

Flex Layout with Wrap

1

2

3

4

5

6

7

8

9

10

Flex Layout with wrap-reverse

9

10

1

2

3

4

5

6

7

8

```
<!DOCTYPE html>
<html>
  <head>
    <link rel="stylesheet" href="Flex.css"/>
  </head>
  <body>
    <h2>Flex Layout </h2>
    <div class="flex-container">
      <div>1</div>
      <div>2</div>
      <div>3</div>
      <div>4</div>
      <div>5</div>
      <div>6</div>
      <div>7</div>
      <div>8</div>
      <div>9</div>
      <div>10</div>
    </div>
  </body>
</html>
```

```
// Flex.css code
.flex-container {
  display: flex;
  /* change flex-wrap value to see other
  Outputs*/
  flex-wrap: wrap;
  background-color: rgb(77,156,224);
}
.flex-container > div {
  background-color: #ebe0e0;
  width: 150px;
  margin: 8px;
  text-align: center;
  line-height: 60px;
  font-size: 25px;
}
```

**4. Flex flow:** it is used as a shorthand property for flex-direction and flex-wrap, it specifies direction along with a wrapping value. The default value is “row nowrap”

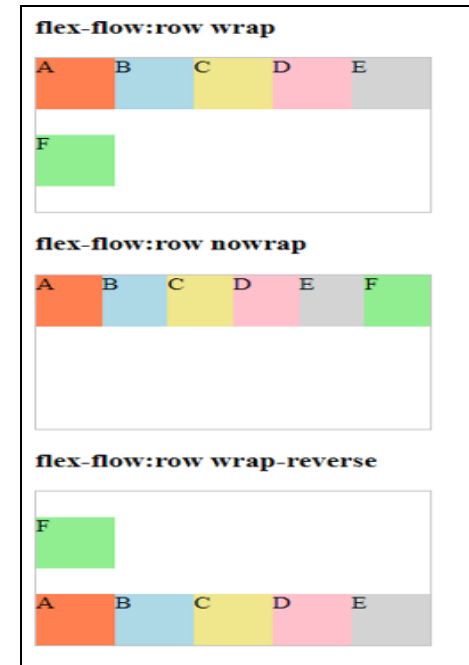
(Experiment with below combinations)

```
.container {
  Flex-flow : row nowrap | row wrap | row wrap-reverse
}

.container {
  Flex-flow : row-reverse nowrap | row-reverse wrap | row-reverse
wrap-reverse
}

.container {
  Flex-flow : column nowrap | column wrap | column wrap-reverse
}

.container {
  Flex-flow : column-reverse nowrap | column-reverse wrap | column-reverse
wrap-reverse
}
```



```
<!DOCTYPE html>
<html>
<head>
  <style>
    #container1 {
      width: 250px;
      height: 150px;
      border: 1px solid #c3c3c3;
      display: flex;
      flex-flow: row wrap;
    }
    #container1 div {
      width: 50px;
      height: 50px;
    }
  </style>
</head>
```

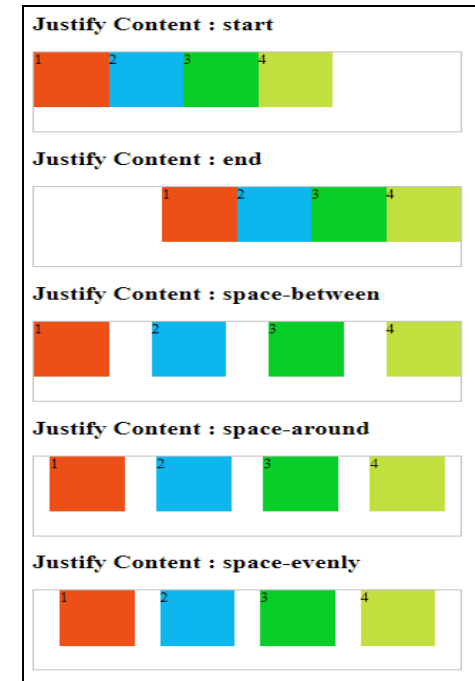
```
<body>
  <h3>flex-flow property</h3>
  <div id="container1">
    <div style="background-color:coral;">A</div>
    <div style="background-color:lightblue;">B</div>
    <div style="background-color:khaki;">C</div>
    <div style="background-color:pink;">D</div>
    <div style="background-color:lightgrey;">E</div>
    <div style="background-color:lightgreen;">F</div>
  </div>
</body>
</html>
```

**5. Justify-Content:** It aligns flex items along with the main axis and distributes leftover space between items, the default value is flex-start.

There are six values for justify-content

- I. **flex-start:** items will be packed towards the start of flex-direction in a container,
- II. **flex-end:** items will be packed towards the end of flex-direction in a container
- III. **Center:** items are placed in the center of the container.
- IV. **Space between:** items will be distributed evenly within a container, the first item will be at the start and the last item will be at the end of the container.
- V. **Space around:** it distributes space around the items in a container
- VI. **Space evenly:** items are distributed in such a way that space between any two items is equal

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



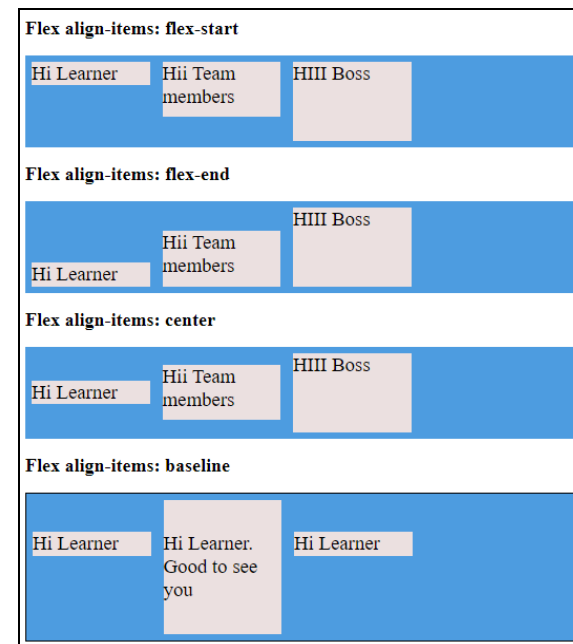
```
<!DOCTYPE html>
<html>
  <head>
    <style>
      #main {
        width: 400px;
        height: 100px;
        border: 1px solid #c3c3c3;
        display: flex;
        justify-content: flex-start;
      }
      #main div {
        width: 70px;
        height: 70px;
      }
    </style>
  </head>
```

```
<body>
  <h2>Justify Content</h2>
  <div id="main">
    <div style="background-color:rgb(236, 80, 22);">1</div>
    <div style="background-color:rgb(14, 182, 238);">2</div>
    <div style="background-color:rgb(9, 206, 42);">3</div>
    <div style="background-color:rgb(193, 223, 63);">4</div>
  </div>
  <h2>Justify Content : end</h2>
</body>
</html>
```

**6. Align item:** It aligns the flex items along the cross axis, the default value is stretch

1. **Flex-start:** items will be placed at the start of the cross-axis in the container.
2. **Flex-end:** items will be placed at the end of the cross-axis in the container
3. **center:** items will be placed at the center of cross-axis
4. **Stretch:** items are stretched throughout the cross axis to fill the container.
5. **baseline:** items are aligned such as their text baseline aligns.

```
.container {
  align-items: flex-start | flex-end | center | stretch | baseline
}
```



```
<!DOCTYPE html>
<html>
  <head>
    <style>
      .flex-container {
        display: flex;
        align-items: flex-start;
        background-color: rgb(77, 156, 224);
      }
      .flex-container > div {
        background-color: #ebe0e0;
        width: 150px;
        margin: 8px;
        line-height: 60px;
        font-size: 25px;
      }
    </style>
  </head>
```

```
<body>
  <h2>Flex Layout </h2>
  <div class="flex-container">
    <div style="min-height: 30px;">Hi Learner</div>
    <div style="min-height: 70px;">Hii Team members</div>
    <div style="min-height: 100px;">HIII Boss</div>
  </div>
</body>
</html>
```

**7. Gap:** It provides a gap between rows and columns, it is used as a shorthand for row-gap and column-gap

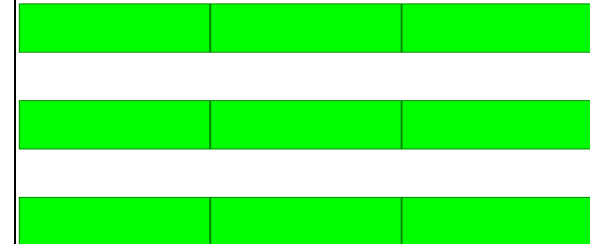
```
.container {
  Gap : 10px ;      // provides gap for both row and column
  Row-gap : 20 px ; // provides gap only for row
  Column-gap : 20px; // provides gap only for column
}
```

```
<!DOCTYPE html>
<html>
  <head>
    <style>
      #flexbox {
        display: flex;
        flex-wrap: wrap;
        width: 720px;
        row-gap: 50px;
      }

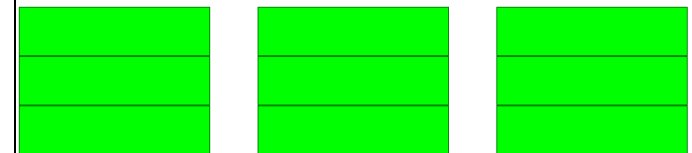
      #flexbox > div {
        border: 1px solid green;
        background-color: lime;
        width: 200px;
        height: 50px;
      }
    </style>
  </head>
```

```
<body>
  <div id="flexbox">
    <div></div>
    <div></div>
    <div></div>
    <div></div>
    <div></div>
    <div></div>
    <div></div>
    <div></div>
  </div>
</body>
</html>
```

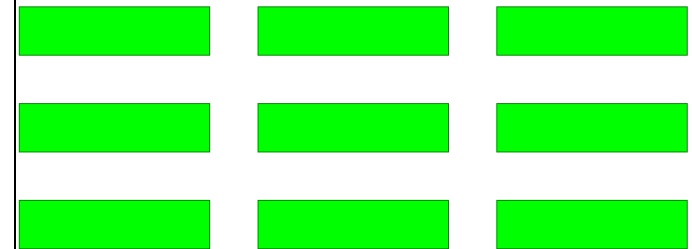
row-gap : 50px



column-gap : 50px



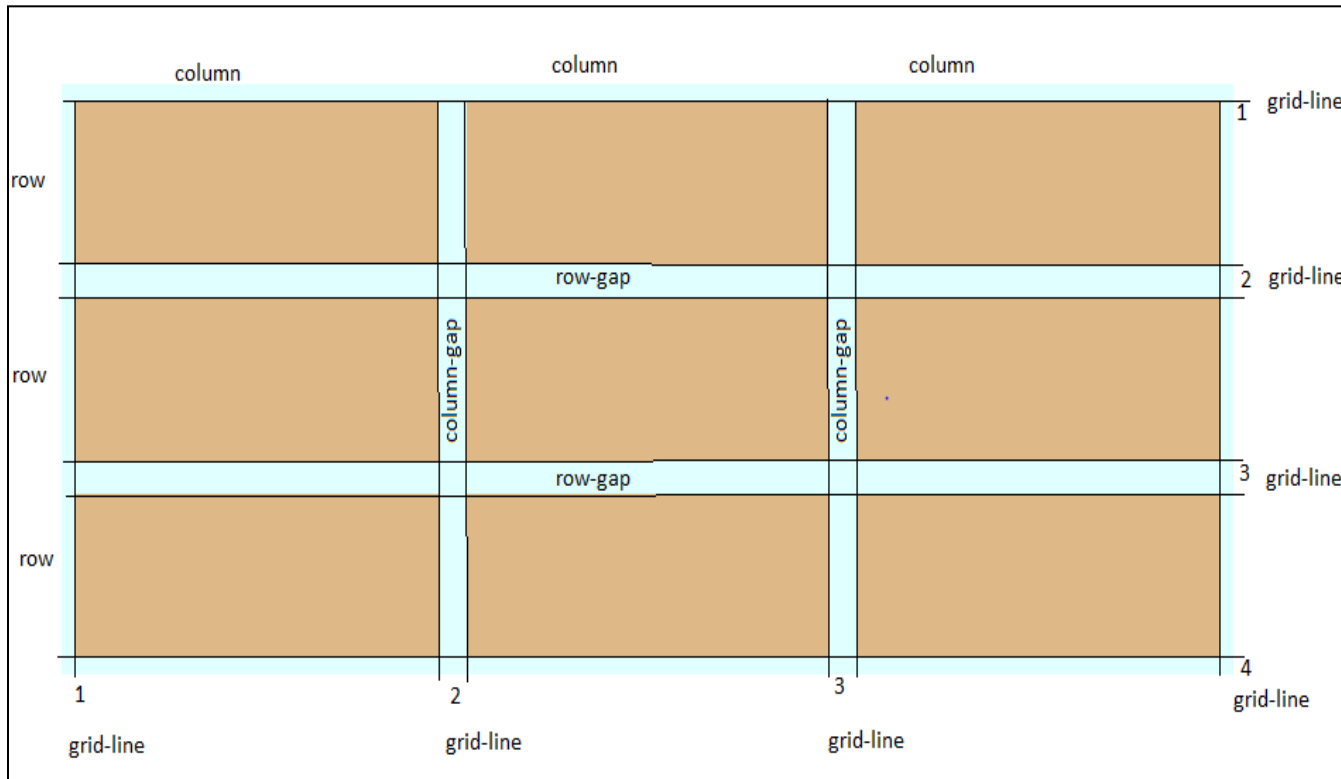
gap: <row>px <column>px



## GRID LAYOUT

CSS grid is a two-dimensional layout, it provides a grid-based layout system with rows and columns to make web pages designing easier.

A grid layout has a parent having one or more children. parents are known as containers and all children are items.





## **Grid Properties:**

**1.Display**

**2.Grid-template-column**

**3.Grid-template-row**

**4.Gird-gap (grid-column-gap ,grid-row-gap)**

**5.Positioning property**

i)grid-row-start

ii)grid-row-end

iii)grid-column-start

iv)grid-column-end

v) grid-row

vi) grid-column

vii) grid-area

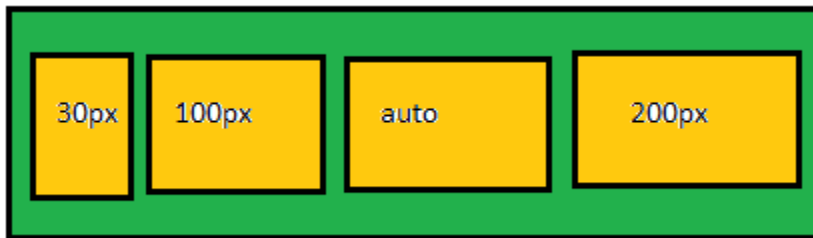
**1.Display** : it display the items in grid form inside a container

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

**2. grid-template-column** : it defines the size of the column and no of columns needed to display.

```
.container {  
  display:grid;  
  Grid-template-column: 30px 100px auto 200px;  
}
```

There will be four columns of size 30 , 100 ,auto , 200 px



For “ auto value” size of the column is determined based on the size of the container and the size of content present in an item.

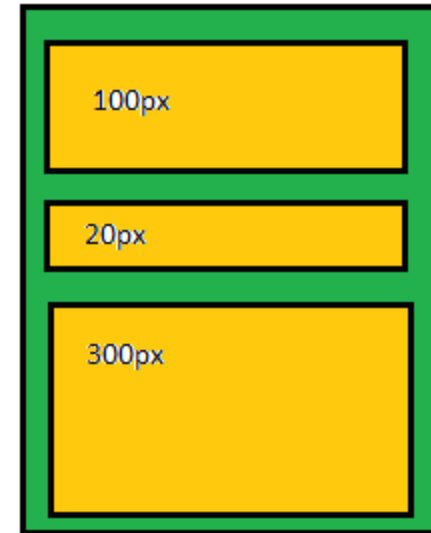
3. **grid-template-row**: it defines the number of rows in a container.

```
.container {  
  display:grid;  
  Grid-template-row: 100px 20px 300px;;  
}
```

There will be three rows of size 100,20,300 px.

Other values which can be assigned to grid-template-columns and grid-template-row are :

1. **max-content**: sets the size of each column and row to the size of the largest item.
2. **min-content**: sets the size of each column and row to the size of the smallest item.
3. **inherit** display items based on the size of the container.
4. **initial**: sets all the items to the default value.



4. **Grid-gap:** it provides a gap between rows and columns, it is a shorthand for grid-column-gap and grid-row-gap.

```
.container {  
  grid-gap: 20px 20px;  
}
```

The first value is for the row and the second one is for the column.

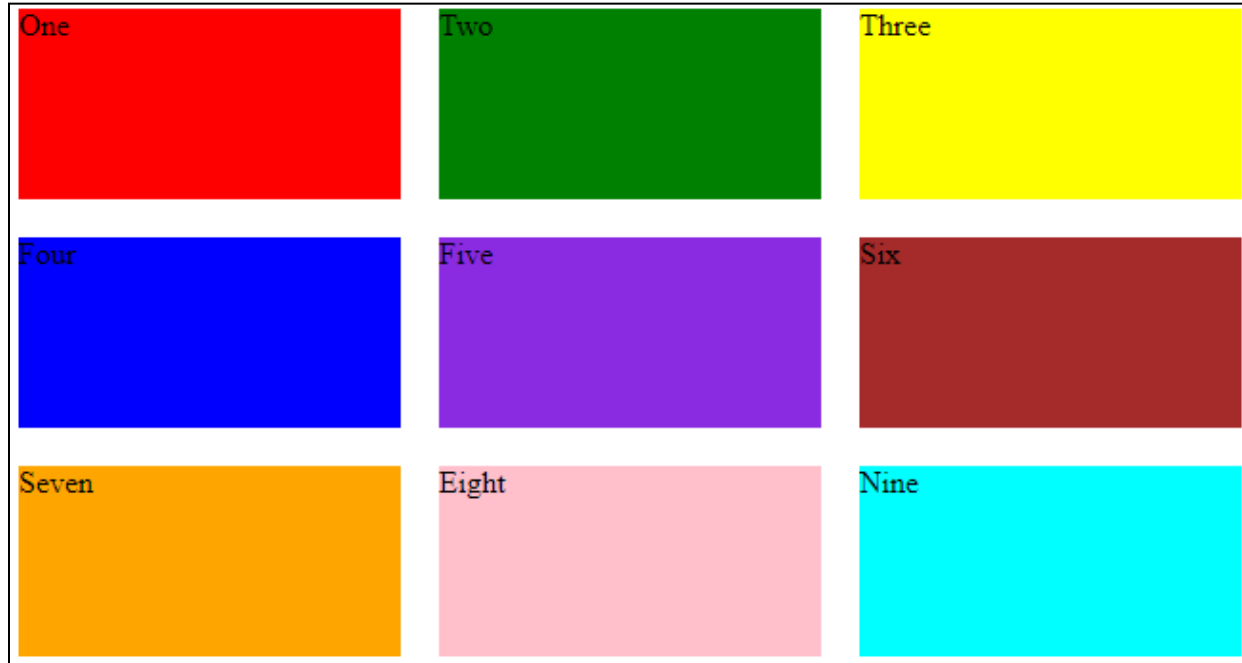


**5. Positioning property:** it deals with the position of items inside the container.

These property will be assigns to **items of container** directly

- i) **grid-row-start** : it defines on which row-line item will start .
- ii) **grid-row-end** : it defines on which row-line item will end.
- iii) **grid-column-start** : it defines on which column-line item will start
- iv) **grid-column-end** : it defines on which column line item will end.

Consider the below items which are in order of 1 to 9, we will change the position of item1 and item9 with each other using the above positioning properties.



```
<!DOCTYPE html>
<html>
  <head>
    <title>Page Title</title>
    <style>
      .container {
        display: grid;
        grid-template-columns: 200px 200px 200px;
        grid-template-rows: 100px 100px 100px;
        margin-top: 10px 10px 10px 10px;
        padding: 10px;
        width: 60%;
        grid-gap: 20px 20px;
      }
      .item1{
        background-color: red;
        grid-row-start: 3;
        grid-row-end: 4;
        grid-column-start: 3;
        grid-column-end: 4;
      }
      .item2{background-color: green;}
      .item3{background-color: yellow;}
      .item4{background-color: blue;}
```

```
.item5{background-color: blueviolet;}
.item6{background-color: brown;}
.item7{background-color: orange;}
.item8{background-color: pink;}
.item9{background-color: cyan;
        grid-area: 1/1/2/2;
        /* grid-area: grid-row-start, grid-column-start,
        grid-row-end, grid-column-end */
      }
    </style>
  </head>
  <body>
    <div class="container">
      <div class="item1">One</div>
      <div class="item2">Two</div>
      <div class="item3">Three</div>
      <div class="item4">Four</div>
      <div class="item5">Five</div>
      <div class="item6">Six</div>
      <div class="item7">Seven</div>
      <div class="item8">Eight</div>
      <div class="item9">Nine</div>
    </div>
  </body>
</html>
```



v) **grid-row** : it is shorthand for grid-row-start and grid-row-end

Grid-row: 2 / 3 (2 : start row position , 3 : end row position)

vi) **grid-column** : it is shorthand for grid-column-start and grid-column-end

Grid-column : 3 / 4 ( 3 : start column position , 4: end column position)

vii) **grid-area**: it is shorthand for row and column, with grid area we can give row and column values at a time.

Grid-area : row-start/column-start/row-end/column-end

Grid-area: 1/1/2/2