CSS – Grid

A grid can be defined as an intersecting set of horizontal lines and vertical lines. CSS Grid layout divides a page into major regions. It defines the relationship between the parts of a control built from HTML primitives in terms of layer, position, and size.

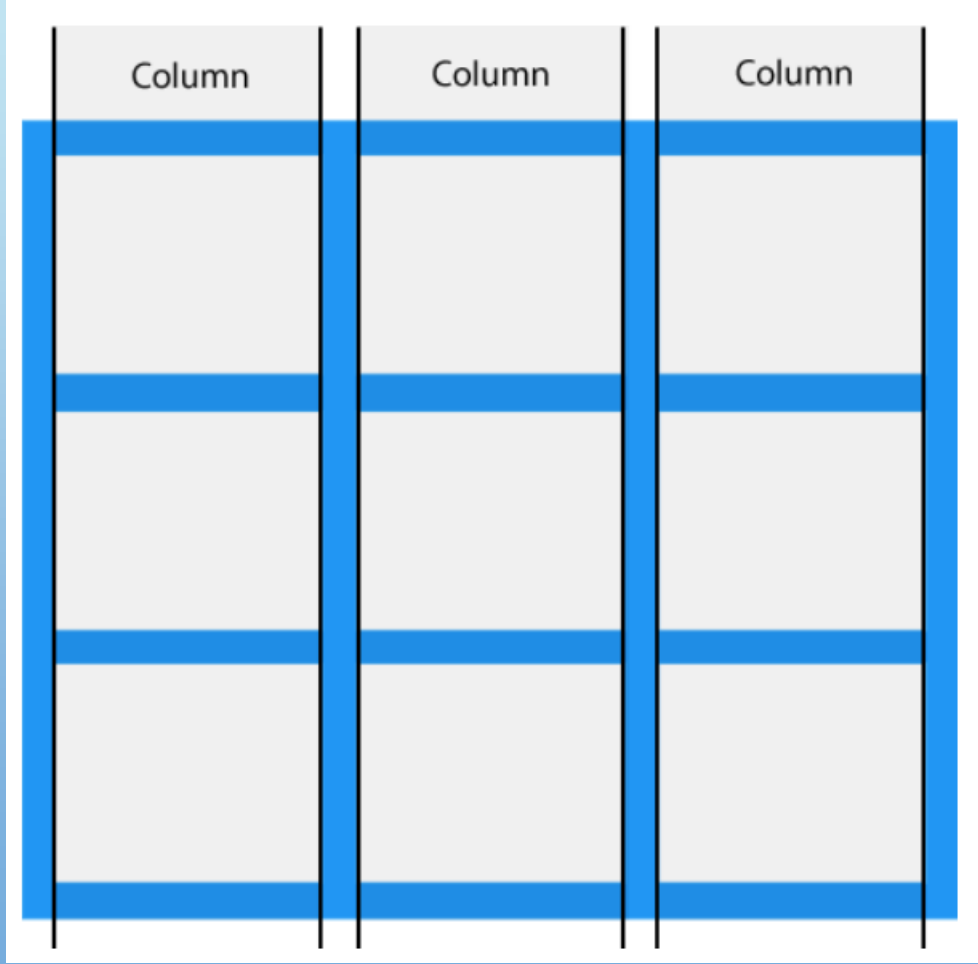
Grid property offers a grid-based layout system having rows and columns. It makes the designing of web pages easy without positioning and floating.

Similar to the table, it enables a user to align the elements into rows and columns. But compare to tables, it is easy to design layout with the CSS grid. Can define columns and rows on the grid by using **grid-template-rows** and **grid-template-columns** properties.

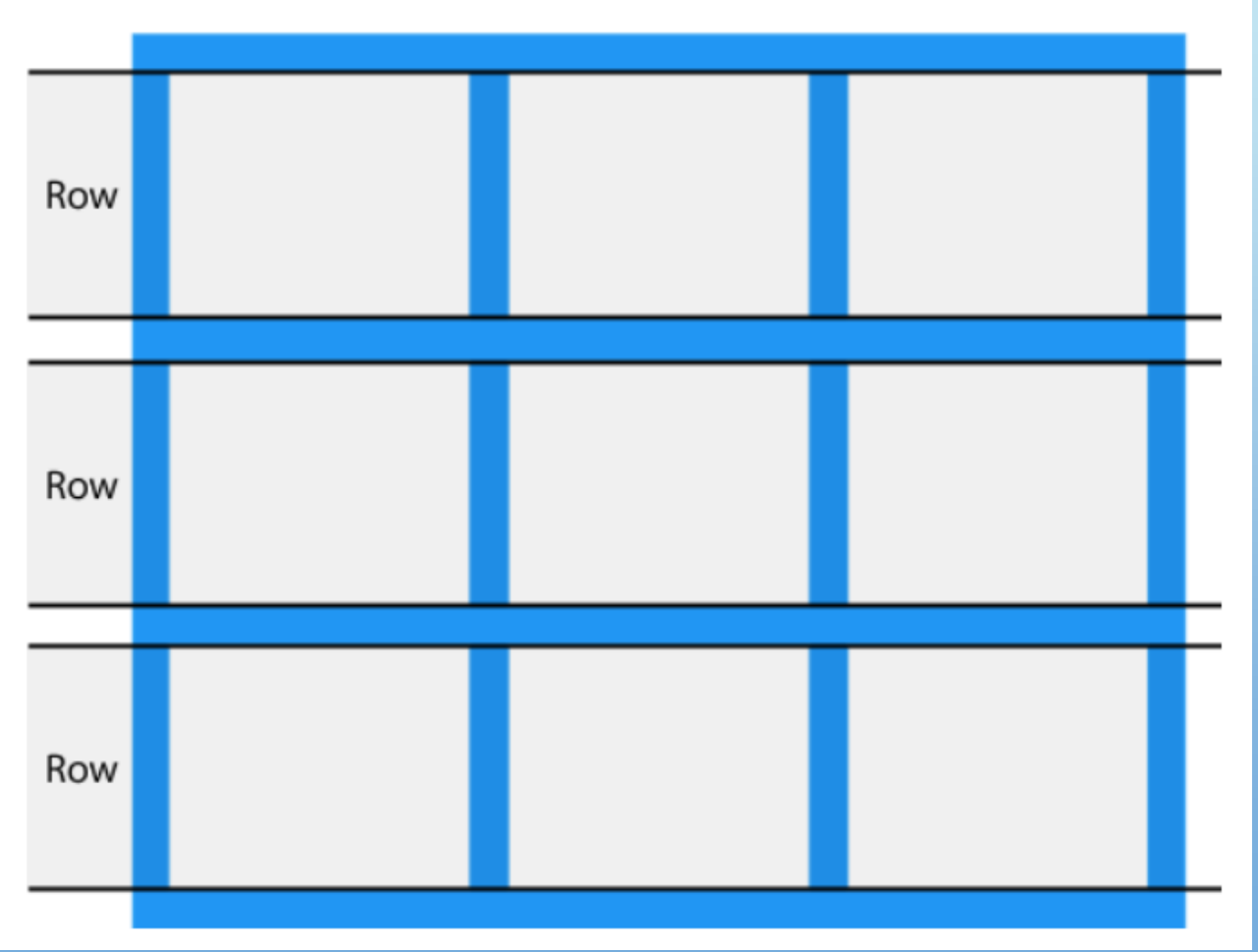
1	2	3
4	5	6
7	8	9

CSS – Grid Columns and Rows

The vertical lines of grid items are called **columns**.

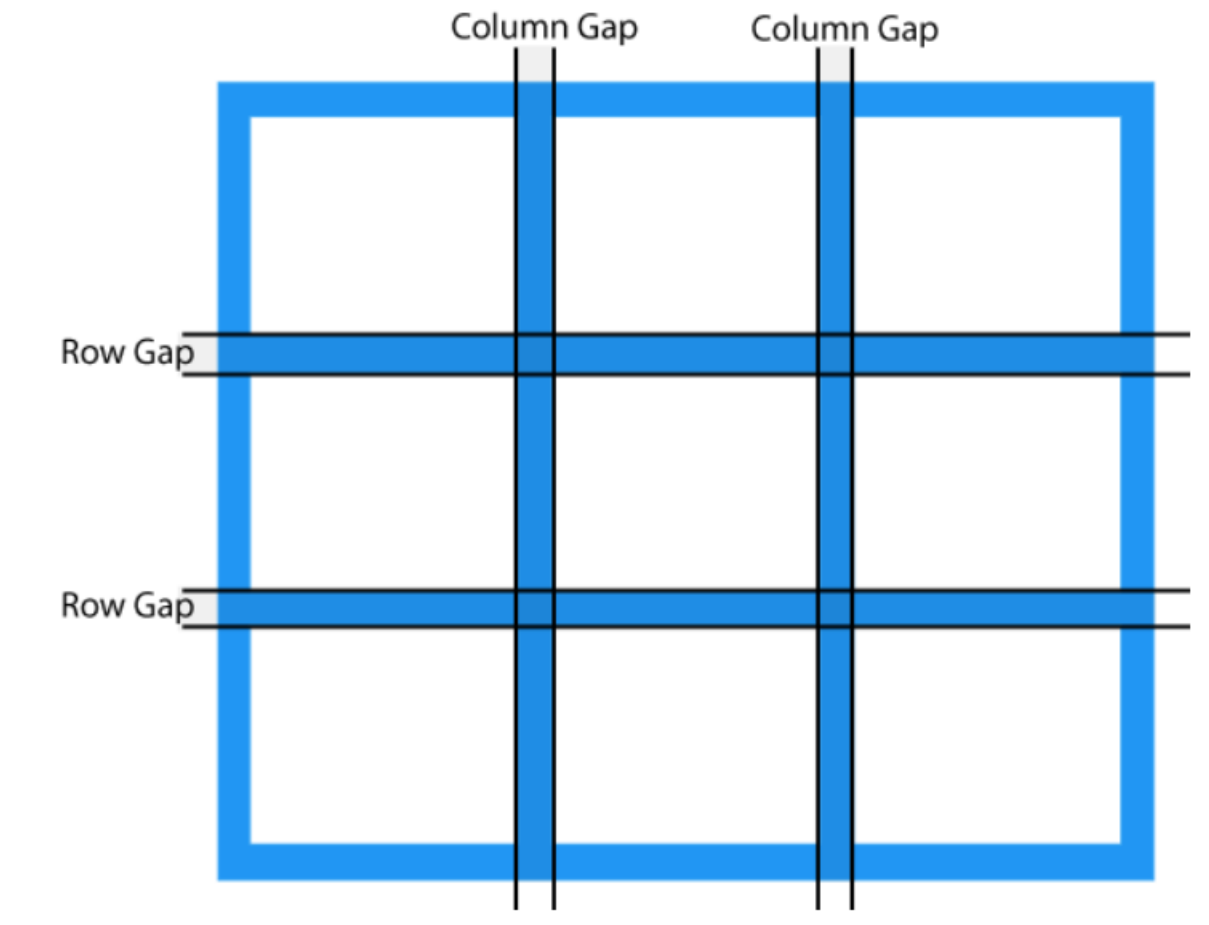


The horizontal lines of grid items are called **rows**.



CSS – Grid Gap

The spaces between each column/row are called *gaps*.



can adjust the gap size by using one of the following properties:

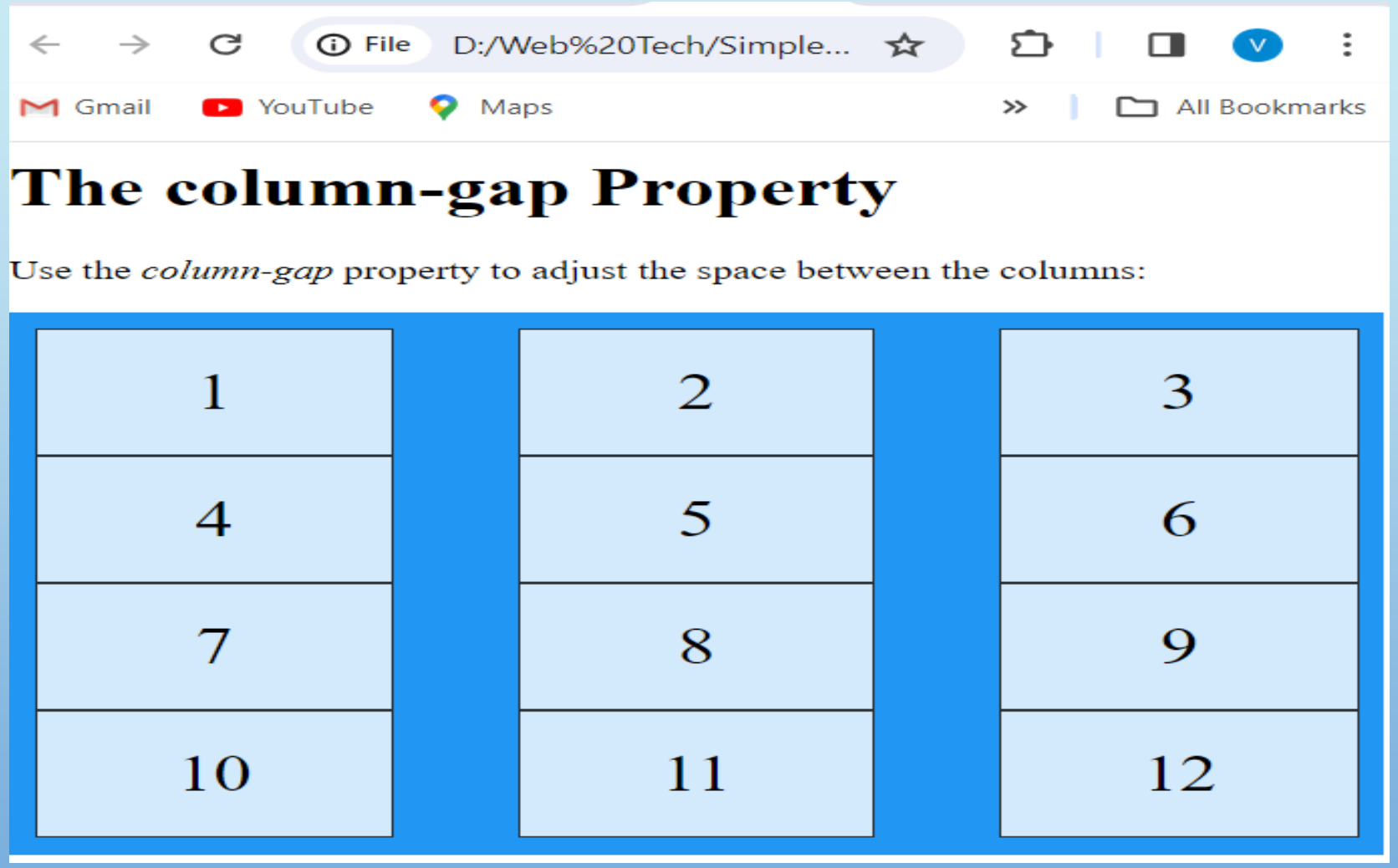
- column-gap
- row-gap
- gap

CSS – Grid Column Gap Example

```
<style>
.grid-container {
  display: grid;
  column-gap: 50px;
  grid-template-columns: auto auto auto;
  background-color: #2196F3;
  padding: 10px;
}

.grid-item {
  background-color: rgba(255, 255, 255, 0.8);
  border: 1px solid rgba(0, 0, 0, 0.8);
  padding: 20px;
  font-size: 30px;
  text-align: center;
}
</style>
```

```
<h1>The column-gap Property</h1>
<p>Use the <em>column-gap</em> property to adjust the
space between the columns:</p>
<div class="grid-container">
  <div class="grid-item">1</div>
  <div class="grid-item">2</div>
  <div class="grid-item">3</div>
  <div class="grid-item">4</div>
  <div class="grid-item">5</div>
  <div class="grid-item">6</div>
  <div class="grid-item">7</div>
  <div class="grid-item">8</div>
  <div class="grid-item">9</div>
 <div class="grid-item">10</div>
  <div class="grid-item">11</div>
  <div class="grid-item">12</div>
</div>
```

CSS – Grid Column Gap Example



CSS – Grid Row Gap Example

```
<style>
.grid-container {
  display: grid;
  row-gap: 50px;
  grid-template-columns: auto auto auto;
  background-color: #2196F3;
  padding: 10px;
}

.grid-item {
  background-color: rgba(255, 255, 255, 0.8);
  border: 1px solid rgba(0, 0, 0, 0.8);
  padding: 20px;
  font-size: 30px;
  text-align: center;
}
</style>
```

```
<body>
<h1>The row-gap Property</h1>
<p>Use the <em>row-gap</em> property to adjust
the space between the rows:</p>

<div class="grid-container">
  <div class="grid-item">1</div>
  <div class="grid-item">2</div>
  <div class="grid-item">3</div>
  <div class="grid-item">4</div>
  <div class="grid-item">5</div>
  <div class="grid-item">6</div>
  <div class="grid-item">7</div>
  <div class="grid-item">8</div>
  <div class="grid-item">9</div>
</div>

</body>
```

CSS – Grid Row Gap Example

The row-gap Property

Use the *row-gap* property to adjust the space between the rows:

1	2	3
4	5	6
7	8	9

CSS – Grid Row & Column Gap Example

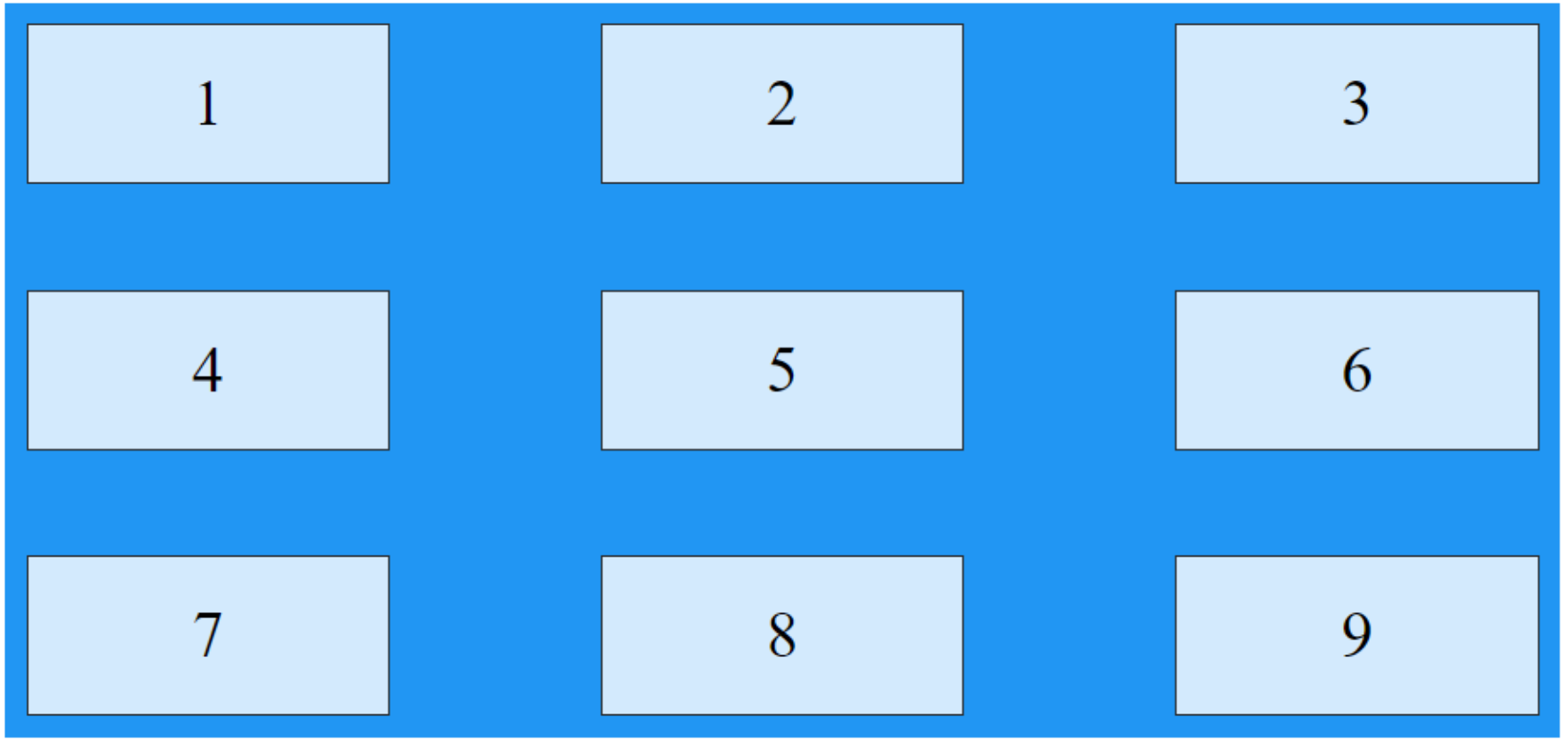
```
<style>
.grid-container {
  display: grid;
  gap: 50px 100px;
  grid-template-columns: auto auto auto;
  background-color: #2196F3;
  padding: 10px;
}

.grid-item {
  background-color: rgba(255, 255, 255, 0.8);
  border: 1px solid rgba(0, 0, 0, 0.8);
  padding: 20px;
  font-size: 30px;
  text-align: center;
}
</style>
```

```
<h1>The gap Property</h1>

<p>Use the <em>gap</em> shorthand property to adjust the space between the columns and rows.</p>

<div class="grid-container">
  <div class="grid-item">1</div>
  <div class="grid-item">2</div>
  <div class="grid-item">3</div>
  <div class="grid-item">4</div>
  <div class="grid-item">5</div>
  <div class="grid-item">6</div>
  <div class="grid-item">7</div>
  <div class="grid-item">8</div>
  <div class="grid-item">9</div>
</div>

</body>
```

CSS – Grid Row & Column Gap Example

The gap Property

Use the *gap* shorthand property to adjust the space between the columns and rows.



CSS – Grid Properties

An HTML element becomes a grid container when its display property is set to grid or inline-grid.

```
.grid-container {
  display: grid;
  grid-template-columns: auto auto auto;
  background-color: #2196F3;
  padding: 10px;
}
```

```
.grid-container {
  display: inline-grid;
  grid-template-columns: auto auto auto;
  background-color: #2196F3;
  padding: 10px;
}
```

Use display: grid; to make a block-level grid container:

1	2	3
4	5	6
7	8	9

Use display: inline-grid;
to make an inline grid container:

1	2	3
4	5	6
7	8	9

CSS – Grid Properties

Property	Description
column-gap	Specifies the gap between the columns
gap	A shorthand property for the *row-gap* and the *column-gap* properties
grid	A shorthand property for the *grid-template-rows, grid-template-columns, grid-template-areas, grid-auto-rows, grid-auto-columns*, and the *grid-auto-flow* properties
grid-area	Either specifies a name for the grid item, or this property is a shorthand property for the *grid-row-start*, *grid-column-start*, *grid-row-end*, and *grid-column-end* properties
grid-auto-columns	Specifies a default column size
grid-auto-flow	Specifies how auto-placed items are inserted in the grid
grid-auto-rows	Specifies a default row size
grid-column	A shorthand property for the *grid-column-start* and the *grid-column-end* properties
grid-column-end	Specifies where to end the grid item
grid-column-gap	Specifies the size of the gap between columns
grid-column-start	Specifies where to start the grid item

CSS – Grid Properties

Property	Description
grid-gap	A shorthand property for the *grid-row-gap* and *grid-column-gap* properties
grid-row	A shorthand property for the *grid-row-start* and the *grid-row-end* properties
grid-row-end	Specifies where to end the grid item
grid-row-gap	Specifies the size of the gap between rows
grid-row-start	Specifies where to start the grid item
grid-template	A shorthand property for the *grid-template-rows*, *grid-template-columns* and *grid-areas* properties
grid-template-areas	Specifies how to display columns and rows, using named grid items
grid-template-columns	Specifies the size of the columns, and how many columns in a grid layout
grid-template-rows	Specifies the size of the rows in a grid layout
row-gap	Specifies the gap between the grid rows

CSS – Grid Container

To make an HTML element behave as a grid container, you have to set the display property to grid or inline-grid.

Grid containers consist of grid items, placed inside columns and rows.

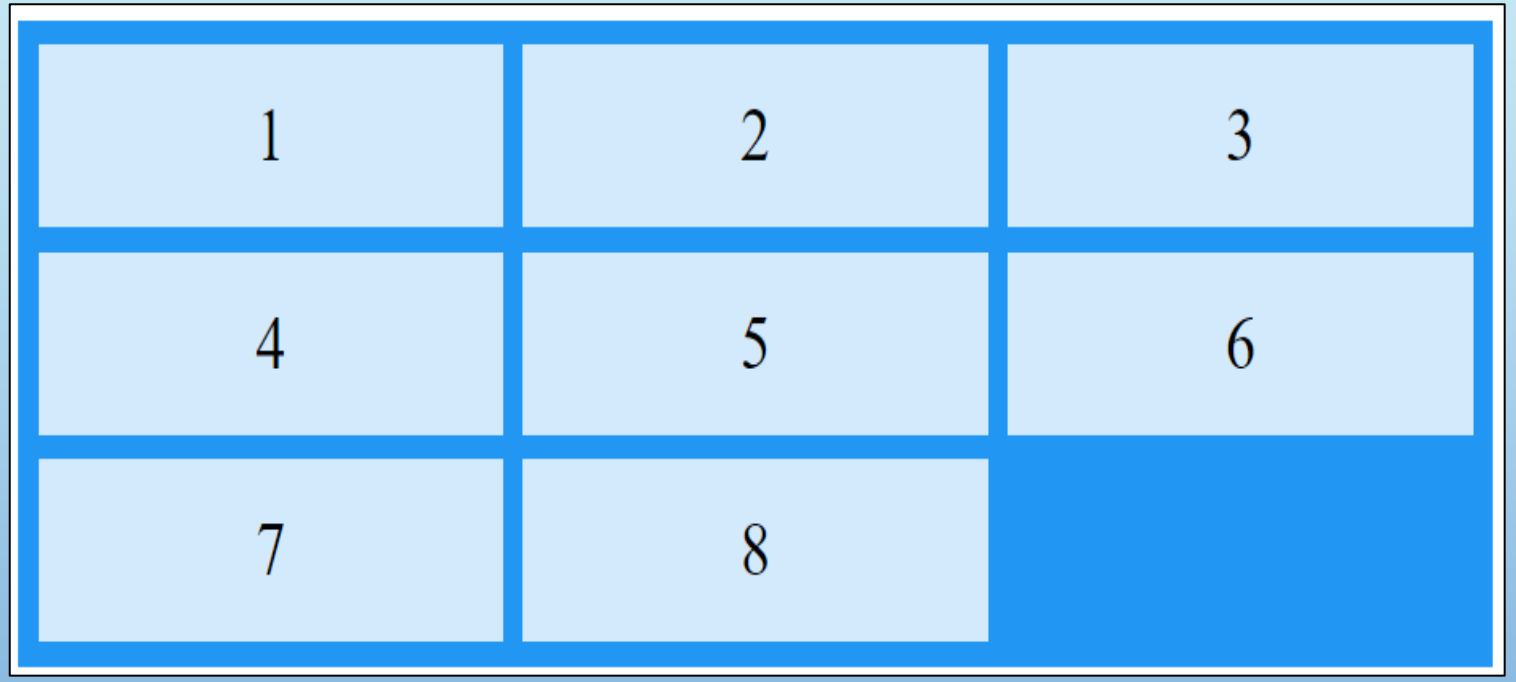
The grid-template-columns property defines the number of columns in your grid layout, and it can define the width of each column.

The value is a space-separated-list, where each value defines the width of the respective column.

If you want your grid layout to contain 4 columns, specify the **width of the 4 columns**, or "**auto**" if all columns should have the same width.

CSS – Grid Properties

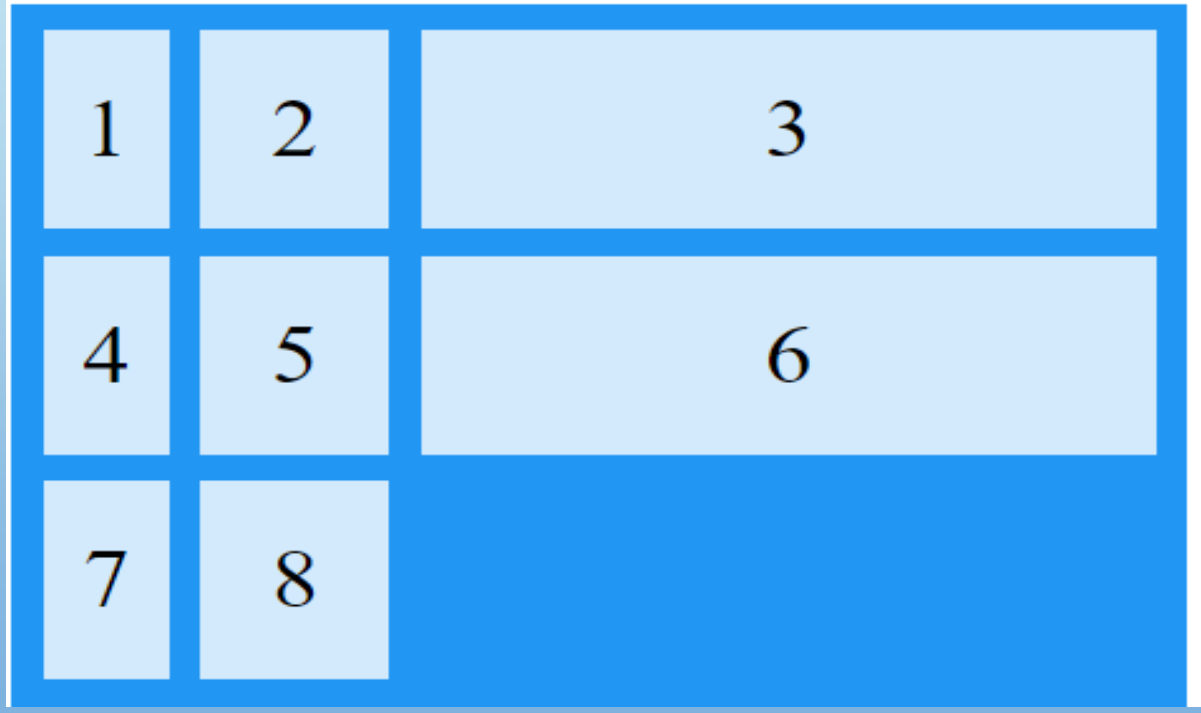
```
<style>
.grid-container {
  display: grid;
  grid-template-columns: auto auto auto;
  gap: 10px;
  background-color: #2196F3;
  padding: 10px;
}

.grid-container > div {
  background-color: rgba(255, 255, 255, 0.8);
  text-align: center;
  padding: 20px 0;
  font-size: 30px;
}
</style>
```



CSS – Grid Properties

```
<style>
.grid-container {
  display: grid;
  grid-template-columns: 40px 60px auto ;
  gap: 10px;
  background-color: #2196F3;
  padding: 10px;
}

.grid-container > div {
  background-color: rgba(255, 255, 255, 0.8);
  text-align: center;
  padding: 20px 0;
  font-size: 30px;
}
</style>
```

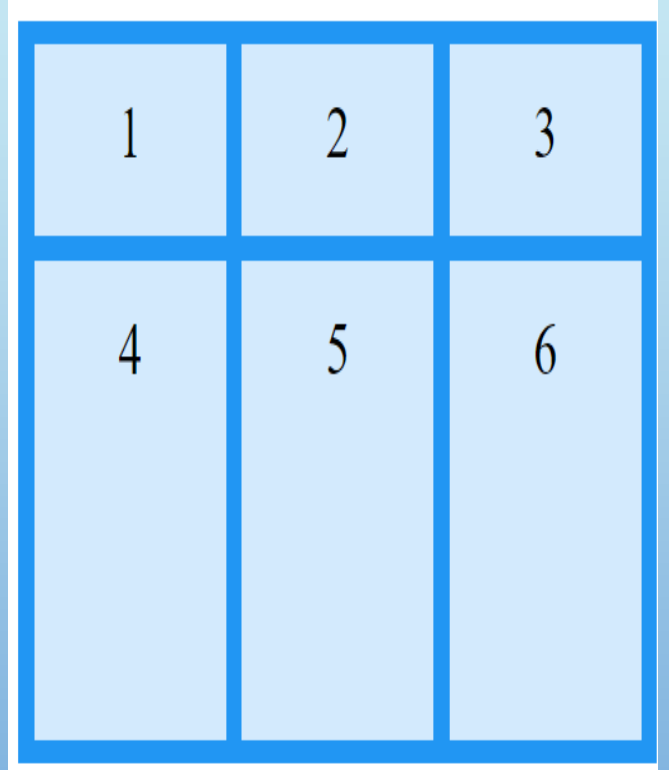
Use the *grid-template-columns* property to specify the size of each column.



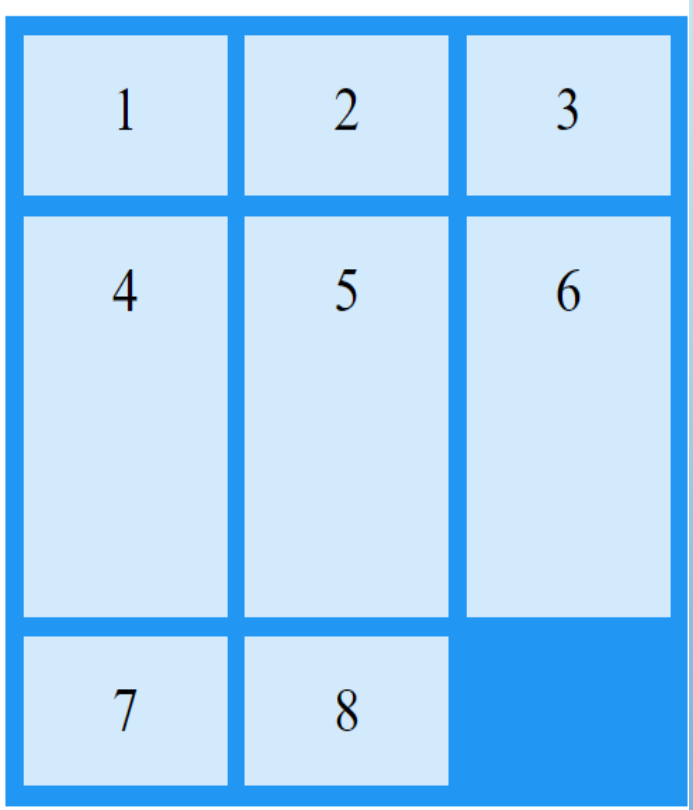
CSS – Grid Properties

The grid-template-rows property defines the height of each row. The value is a space-separated-list, where each value defines the height of the respective row:

```
<style>
.grid-container {
  display: grid;
  grid-template-columns: auto auto auto;
  grid-template-rows: 80px 200px;
  gap: 10px;
  background-color: #2196F3;
  padding: 10px;
}

.grid-container > div {
  background-color: rgba(255, 255, 255, 0.8);
  text-align: center;
  padding: 20px 0;
  font-size: 30px;
}
</style>
```



Use the *grid-template-rows* property to specify the size (height) of each row.



Use the *grid-template-rows* property to specify the size (height) of each row.

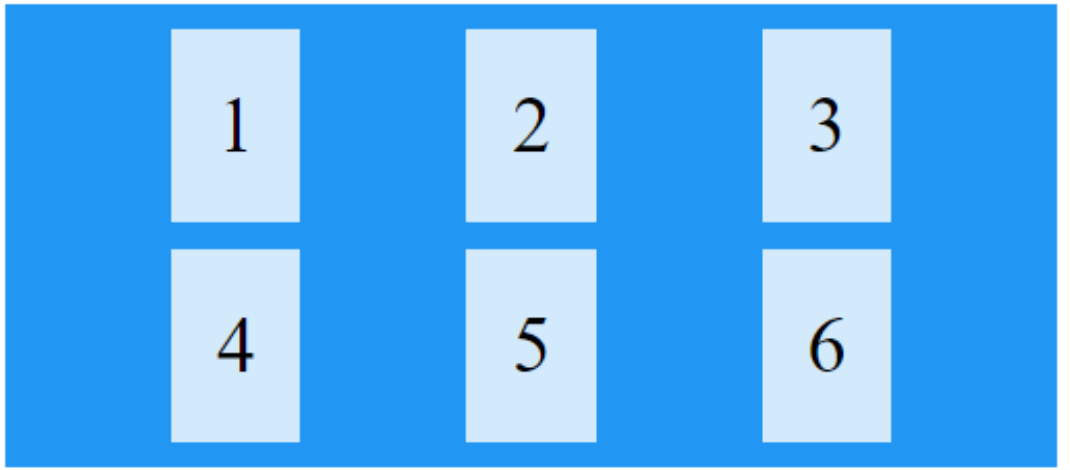
CSS – Justify Property

The justify-content property is used to align the whole grid inside the container.

```
<style>
.grid-container {
  display: grid;
  justify-content: space-evenly;
  /*Make the grid smaller than the container*/
  grid-template-columns: 50px 50px 50px;
  gap: 10px;
  background-color: #2196F3;
  padding: 10px;
}

.grid-container > div {
  background-color: rgba(255, 255, 255, 0.8);
  text-align: center;
  padding: 20px 0;
  font-size: 30px;
}
</style>
```

Use the *justify-content* property to align the grid inside the container.

The value "space-evenly" will give the columns equal amount of space between, and around them:

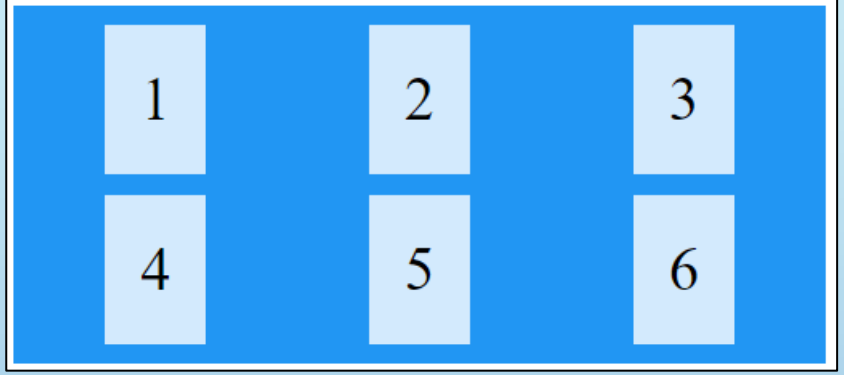


CSS – Grid Justify Content Property

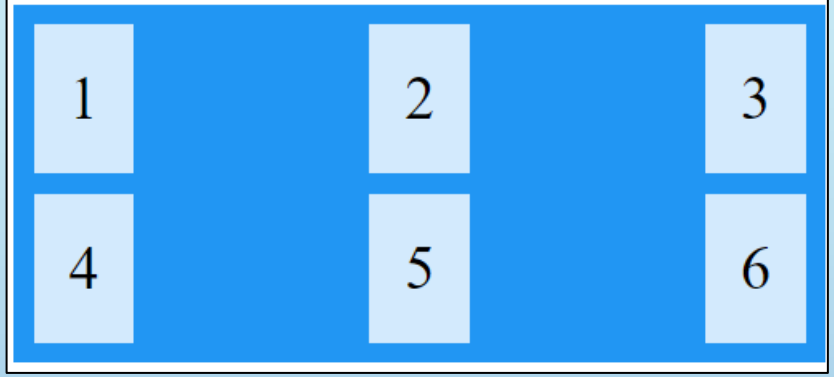
- **space-evenly**: It provides equal space in between or around the columns.
- **space-around**: It provides equal space around the columns.
- **space-between**: It gives an equal amount of space between the columns.
- **center**: It is used to align the grid in the middle of the container.
- **start**: It is used to align the grid at the beginning of the container.
- **end**: It is used to align the grid at the end of the container.

CSS – Justify Properties

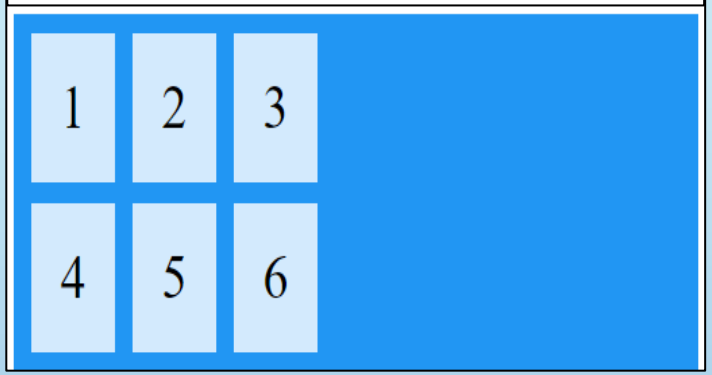
justify-content: space-evenly;



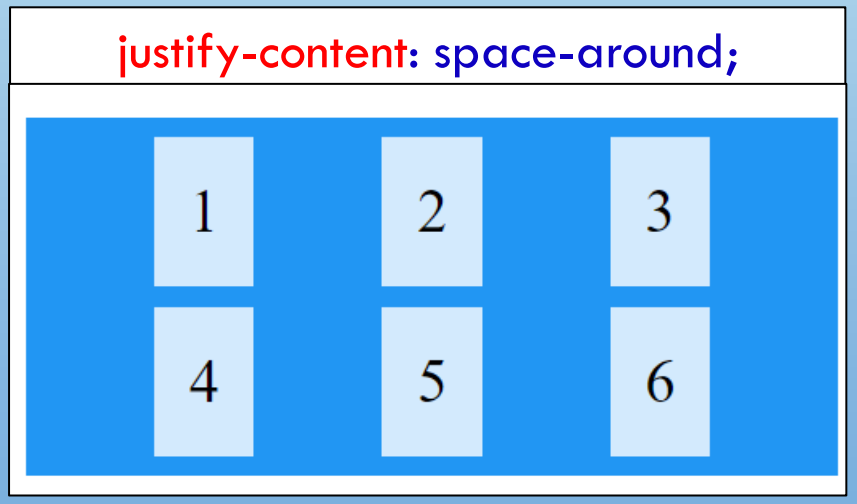
justify-content: space-between;



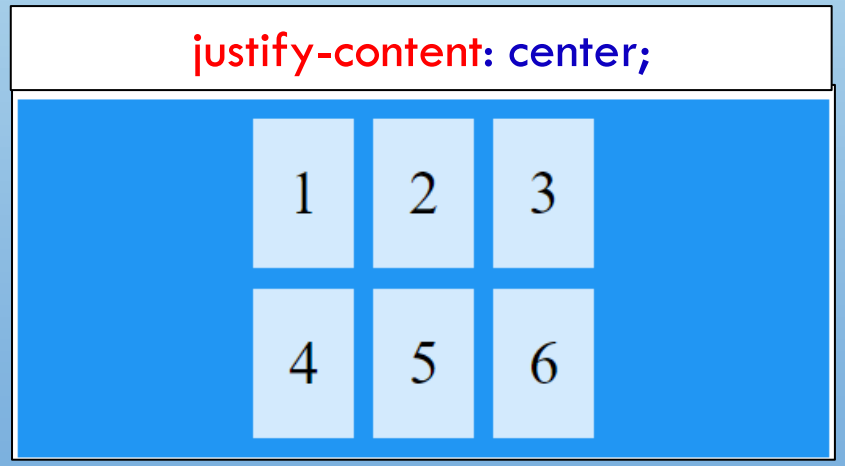
justify-content: start;



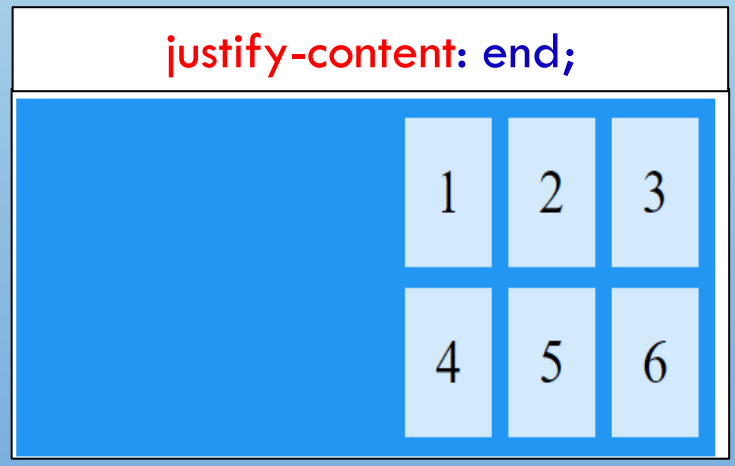
justify-content: space-around;



justify-content: center;



justify-content: end;



CSS – Grid Column Properties

A grid *container* contains grid *items*. By default, a container has one grid item for each column, in each row, but you can style the grid items so that they will span multiple columns and/or rows.

The grid-column property defines on which column(s) to place an item. You define where the item will start, and where the item will end.

The **grid-column property** is a shorthand property for the **grid-column-start** and the **grid-column-end** properties.

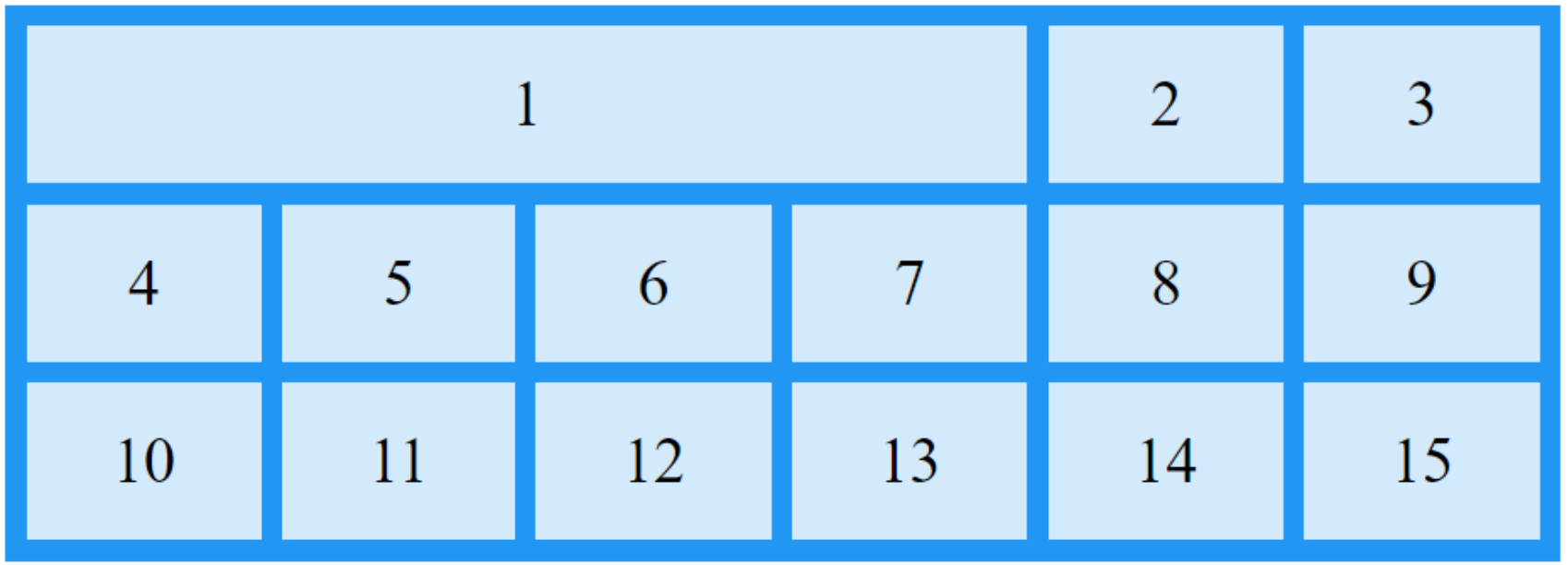
CSS – Grid Column Properties

```
<!DOCTYPE html>
<html> <head><style>
.grid-container {
  display: grid;
  grid-template-columns: auto auto auto auto auto auto;
  gap: 10px;
  background-color: #2196F3;
  padding: 10px;}
.grid-container > div {
  background-color: rgba(255, 255, 255, 0.8);
  text-align: center;
  padding: 20px 0;
  font-size: 30px;}
.item1 {
  grid-column: 1 / span 4;
}
</style> </head> <body>
<p>Use the <em>grid-column</em> property to specify
where to place an item.</p>
<p>Item1 will start on column 1 and span 3 columns</p>
```

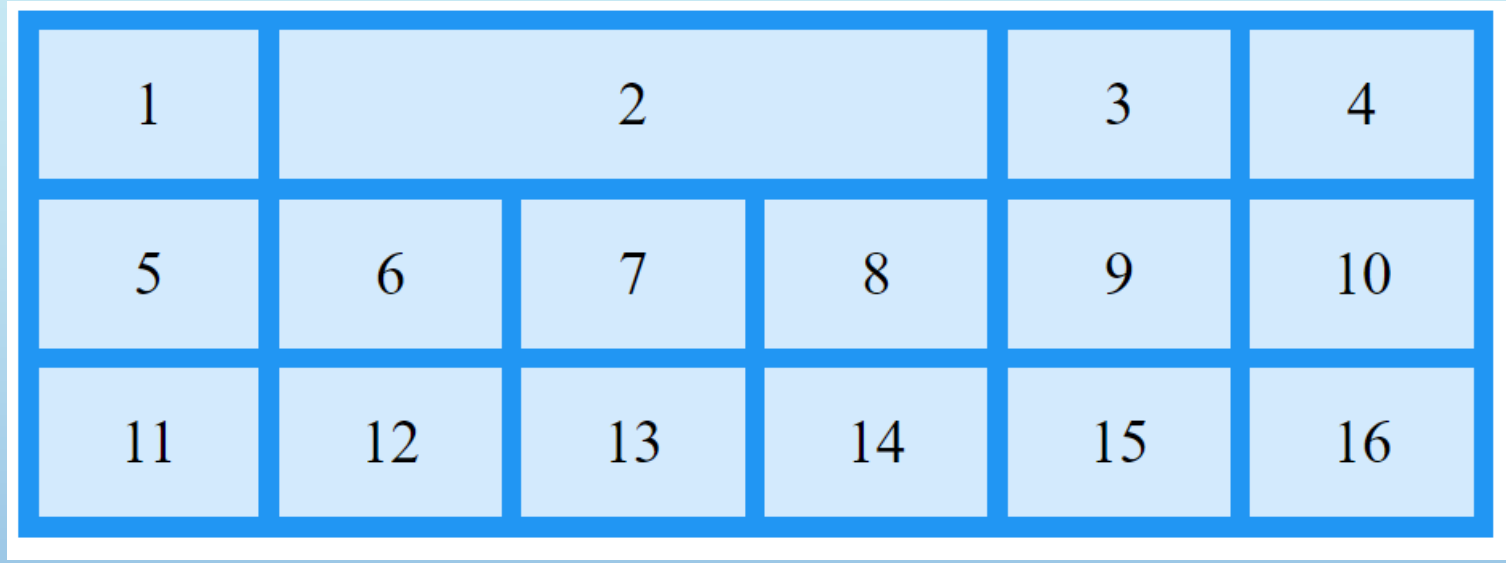
```
<div class="grid-container">
  <div class="item1">1</div>
  <div class="item2">2</div>
  <div class="item3">3</div>
  <div class="item4">4</div>
  <div class="item5">5</div>
  <div class="item6">6</div>
  <div class="item7">7</div>
  <div class="item8">8</div>
  <div class="item9">9</div>
  <div class="item10">10</div>
  <div class="item11">11</div>
  <div class="item12">12</div>
  <div class="item13">13</div>
  <div class="item14">14</div>
  <div class="item15">15</div>
</div>
</body> </html>
```

CSS – Grid Column Properties

Item1 will start on column 1 and end before column 5:



CSS – Grid Column Properties



What is the style for item-?

.item2{ grid-column: 2/ span 3;}

CSS – Grid Row Properties

The grid-row property defines on which row to place an item.

You define where the item will start, and where the item will end.

The **grid-row** property is a shorthand property for the **grid-row-start** and the **grid-row-end** properties.

CSS – Grid Row Properties

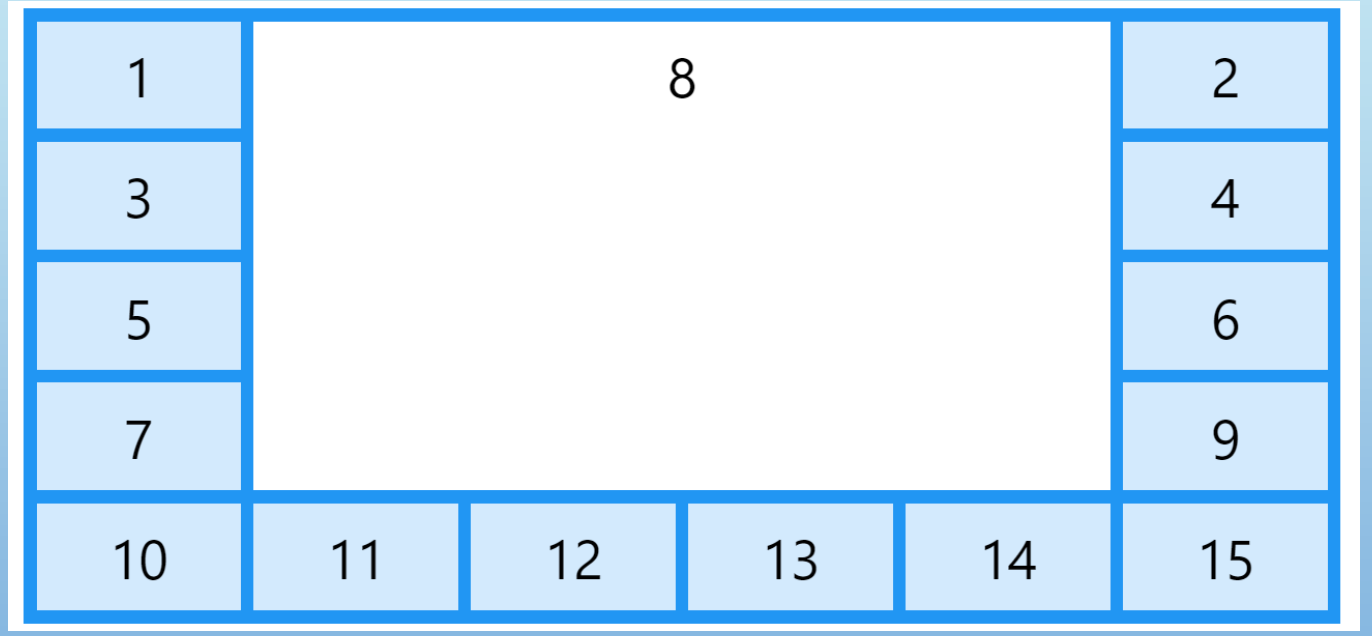
```
<style>
.grid-container {
  display: grid;
  grid-template-columns: auto auto auto auto auto auto;
  gap: 10px;
  background-color: #2196F3;
  padding: 10px;
}

.grid-container > div {
  background-color: rgba(255, 255, 255, 0.8);
  text-align: center;
  padding: 20px 0;
  font-size: 30px;
}

.item1 {
  grid-row: 1 / span 2;
}
```

Item1 will start on row 1 and span 2 rows:

1	2	3	4	5	6
	7	8	9	10	11
12	13	14	15	16	17

CSS – Grid Area Properties

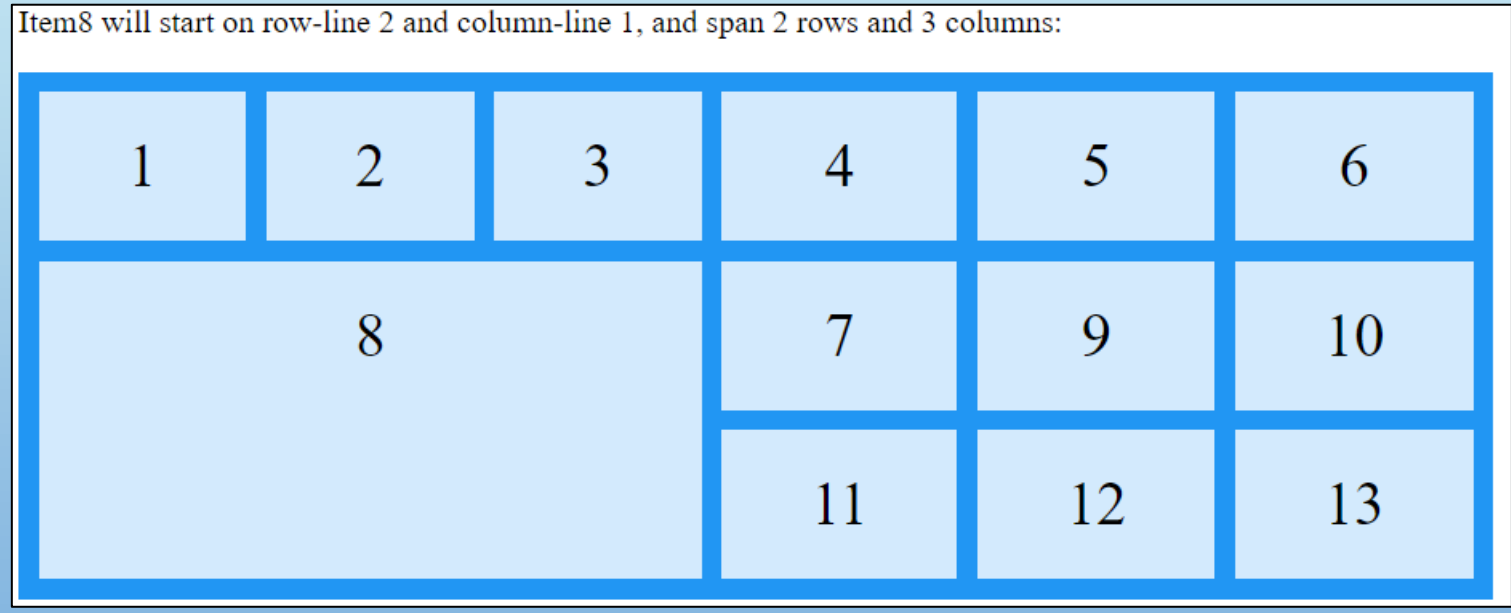
The **grid-area** property can be used as a shorthand property for the **grid-row-start**, **grid-column-start**, **grid-row-end** and the **grid-column-end** properties.



CSS – Grid Area Properties

Make "item8" start on row-line 2 and column-line 1, and span 2 rows and 3 columns:

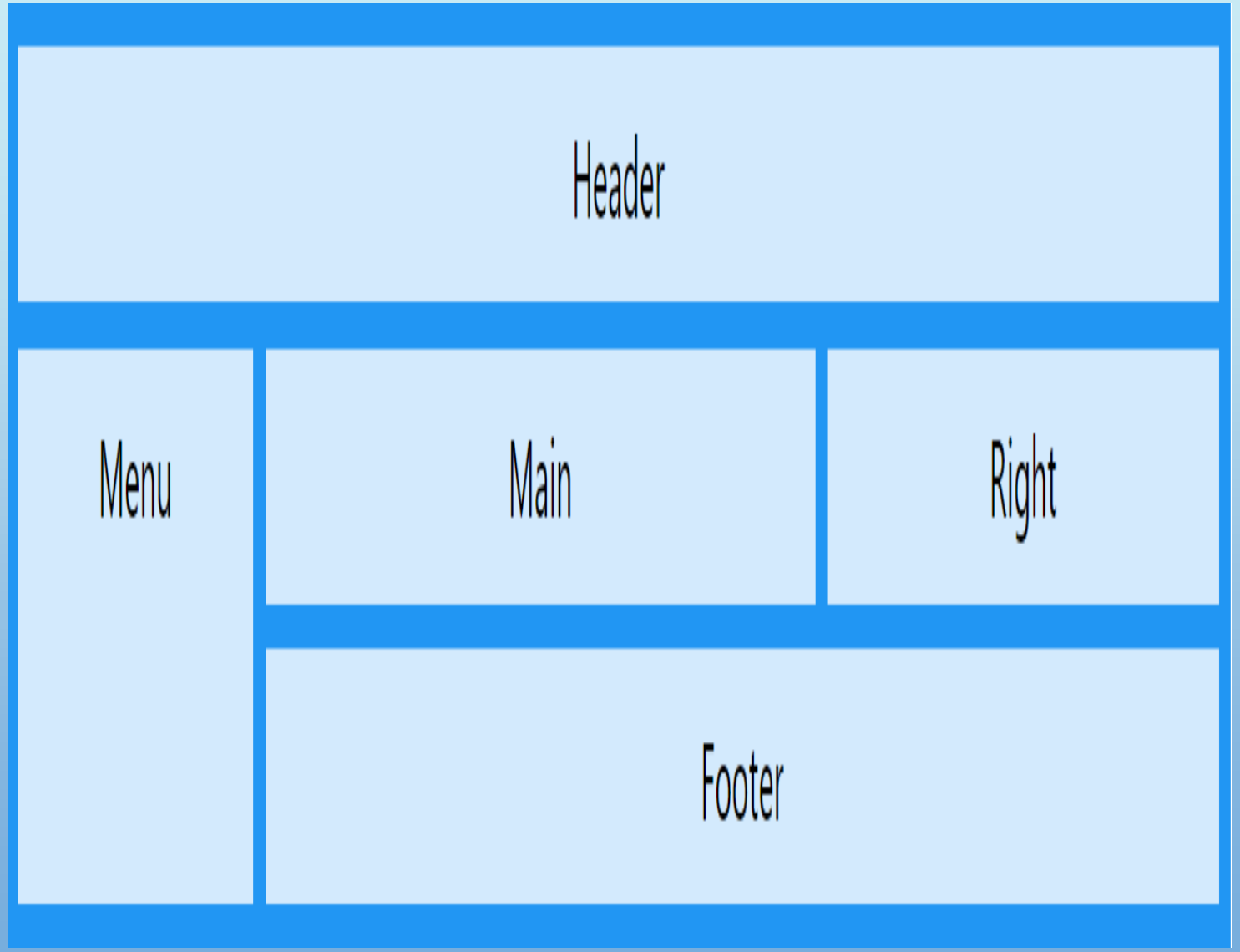
```
<style>
.grid-container {
  display: grid;
  grid-template-columns: auto auto auto auto auto auto;
  gap: 10px;
  background-color: #2196F3;
  padding: 10px;
}

.grid-container > div {
  background-color: rgba(255, 255, 255, 0.8);
  text-align: center;
  padding: 20px 0;
  font-size: 30px;
}

.item8 {
  grid-area: 2 / 1 / span 2 / span 3;
}
```



CSS – Grid Customize Layout

The grid-area property can also be used to assign names to grid items.

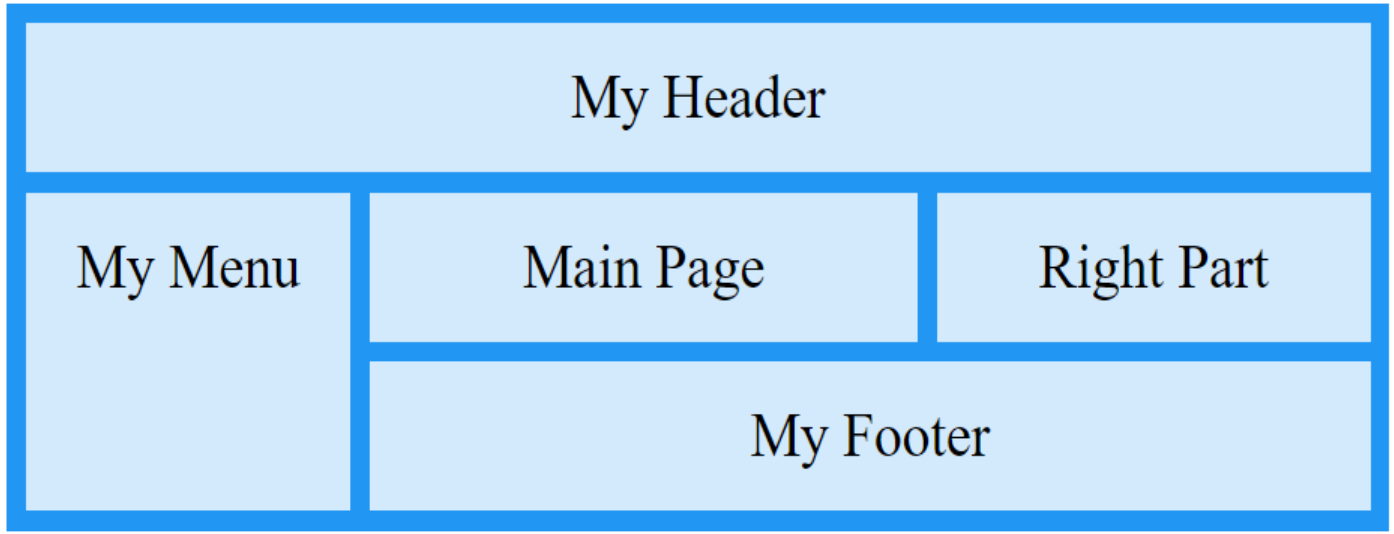
The grid-area property can also be used to assign names to grid items.



CSS – Grid Customize Layout

```
<head><style>
.item1 { grid-area: header; }
.item2 { grid-area: menu; }
.item3 { grid-area: main; }
.item4 { grid-area: right; }
.item5 { grid-area: footer; }
.grid-container {
  display: grid;
  grid-template-areas:
    'header header header header header header '
    'menu main main main right right'
    'menu footer footer footer footer footer';
  gap: 10px;
  background-color: #2196F3;
  padding: 10px;
}
.grid-container > div {
  background-color: rgba(255, 255, 255, 0.8);
  text-align: center;
  padding: 20px 0;
  font-size: 30px;
}</style></head>
```

```
<div class="grid-container">
  <div class="item1">My Header</div>
  <div class="item2">My Menu</div>
  <div class="item3"> Main Page</div>
  <div class="item4">Right Part</div>
  <div class="item5">My Footer</div>
</div></body></html>
```

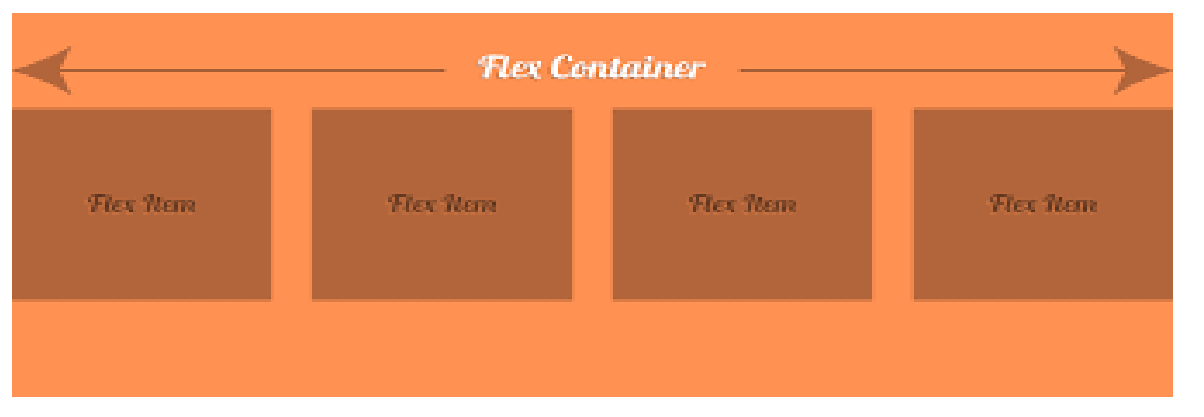


References

https://www.w3schools.com/css/

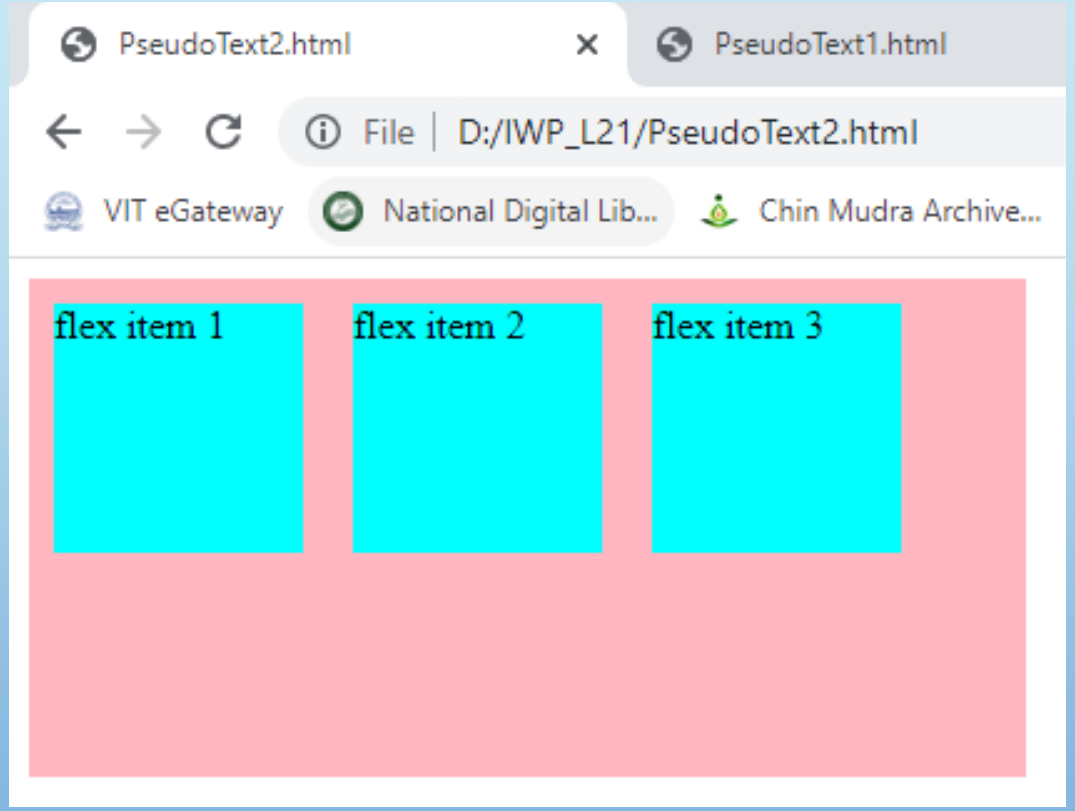
CSS – Flexbox

- **Flexbox** is a new layout mode in CSS3.
- The CSS3 flexbox is used to make the elements behave predictably when they are used with different screen sizes and different display devices. It provides a more efficient way to layout, align and distribute space among items in the container.
- The CSS3 flexbox contains **flex containers** and **flex items**.
- **Flex container:** The flex container specifies the properties of the parent. It is declared by setting the display property of an element to either flex or inline-flex.
- **Flex items:** The flex items specify properties of the children. There may be one or more flex items inside a flex container.



CSS – Flexbox Example

```
<!DOCTYPE html>
<html> <head> <style>
.flex-container {
   display: -webkit-flex;
    display: flex;
    width: 400px;
    height: 200px;
    background-color: lightpink;  }
.flex-item {
    background-color: cyan;
    width: 100px;
    height: 100px;
    margin: 10px;  }
</style>
</head>
<body>
<div class="flex-container">
  <div class="flex-item">flex item 1</div>
  <div class="flex-item">flex item 2</div>
  <div class="flex-item">flex item 3</div>
</div> </body> </html>
```



CSS – Flexbox Properties

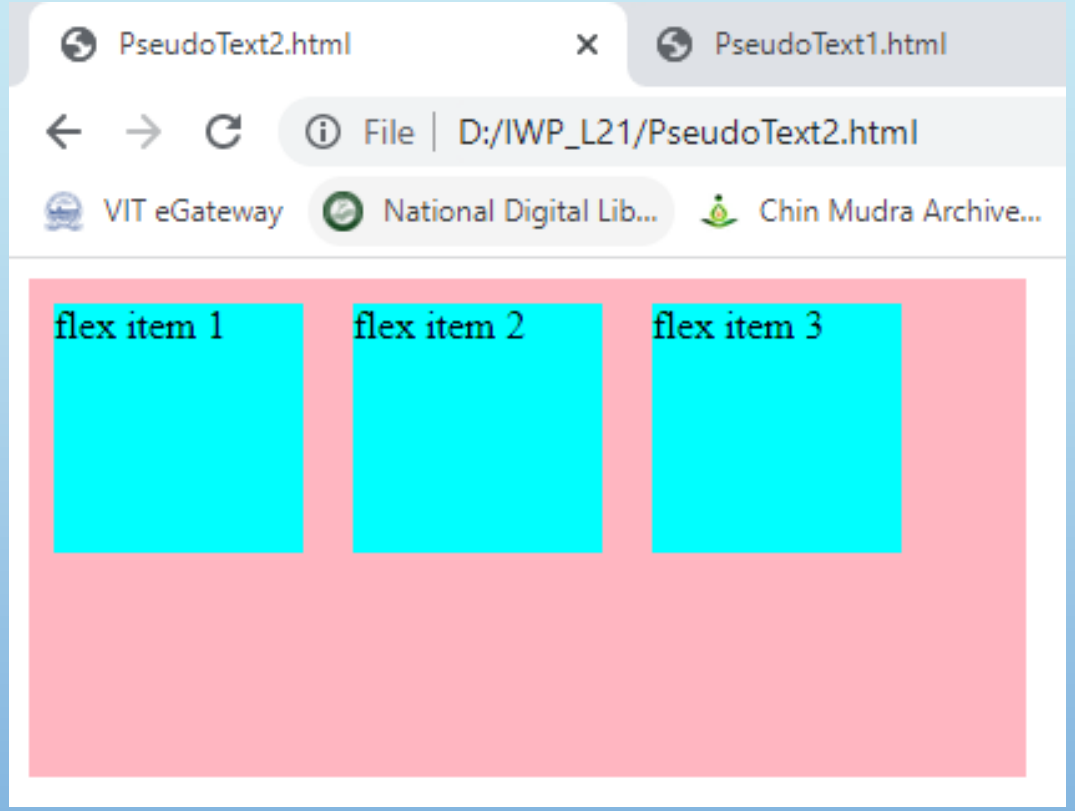
property	description
display	it is used to specify the type of box used for an html element.
flex-direction	it is used to specify the direction of the flexible items inside a flex container. The possible values are **row-reverse**, **column**, **column-reverse**.
justify-content	it is used to align the flex items horizontally when the items do not use all available space on the main-axis. Possible values are **flex-start**, **flex-end**, **center**, **space-between**, **space-around**
align-items	it is used to align the flex items vertically when the items do not use all available space on the cross-axis. Possible values are **stretch, flex-start, flex-end, center, baseline**.
flex-wrap	it specifies whether the flex items should wrap or not, if there is not enough room for them on one flex line. Possible values are **nowrap, wrap, wrap-reverse**

CSS – Flexbox Properties

property	description
align-content	it is used to modify the behavior of the flex-wrap property. it is similar to align-items, but instead of aligning flex items, it aligns flex lines. . Possible values are **flex-start**, **flex-end**, **center**, **space-between**, **space-around**
flex-flow	it specifies a shorthand property for flex-direction and flex-wrap.
order	it specifies the order of a flexible item relative to the rest of the flex items inside the same container.
align-self	it is used on flex items. it overrides the container's align-items property.
flex	it specifies the length of a flex item, relative to the rest of the flex items inside the same container.

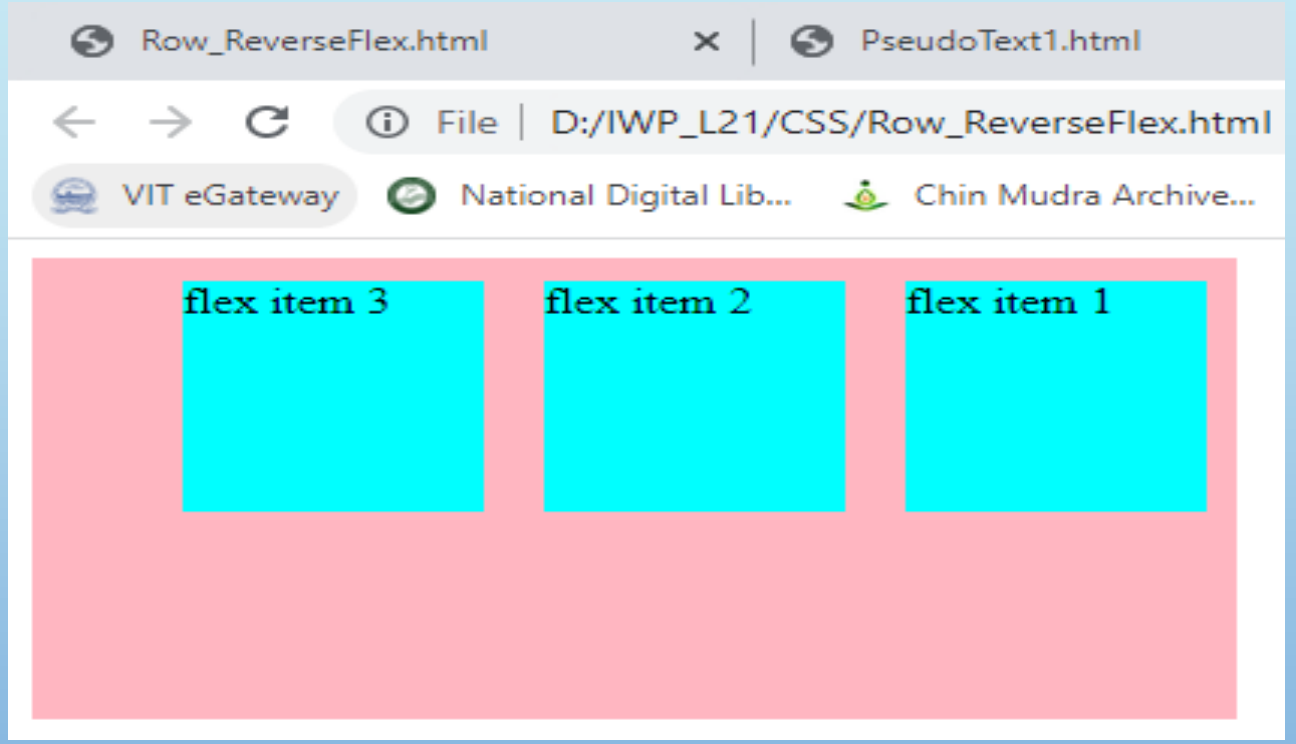
CSS – Flexbox Example

```
<!DOCTYPE html>
<html> <head> <style>
.flex-container {
   display: -webkit-flex;
    display: flex;
    width: 400px;
    height: 200px;
    background-color: lightpink;  }
.flex-item {
    background-color: cyan;
    width: 100px;
    height: 100px;
    margin: 10px;  }
</style>
</head>
<body>
<div class="flex-container">
  <div class="flex-item">flex item 1</div>
  <div class="flex-item">flex item 2</div>
  <div class="flex-item">flex item 3</div>
</div> </body> </html>
```



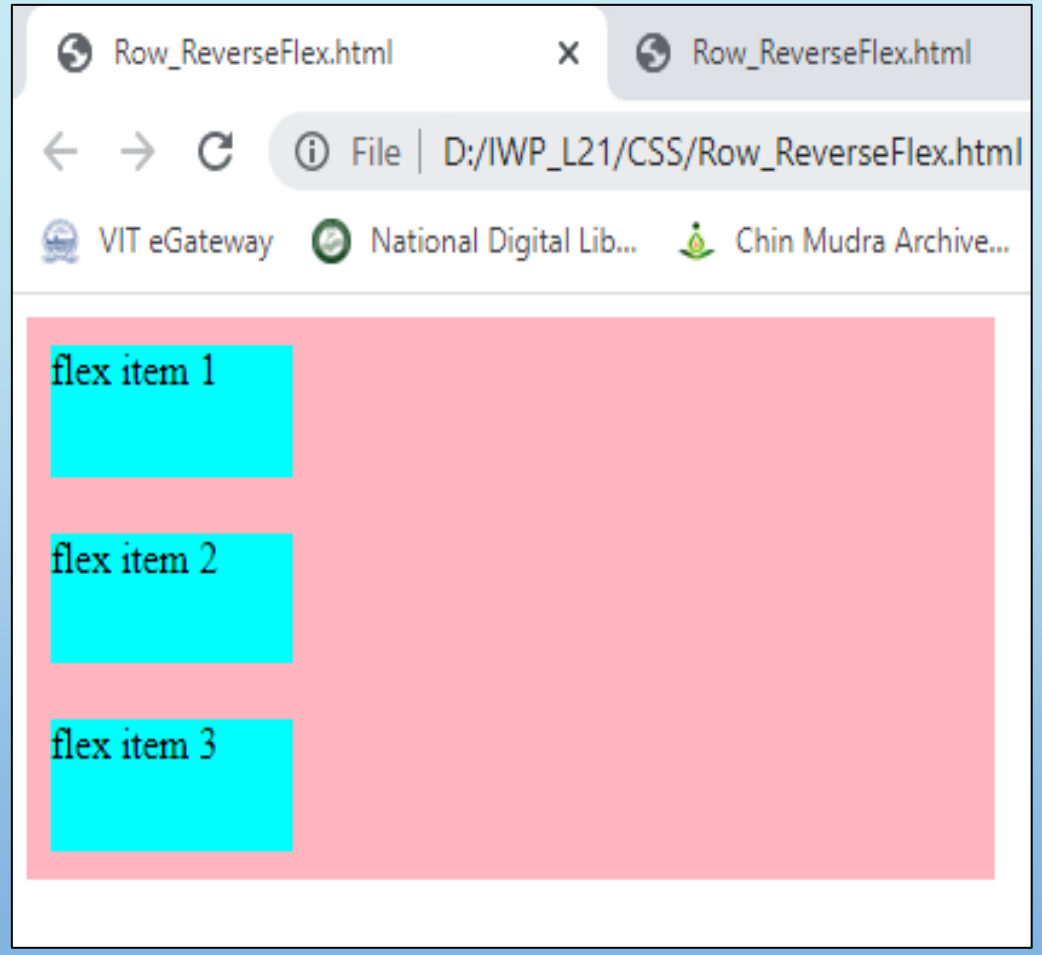
CSS – Flexbox Direction (row-reverse)

```
<!DOCTYPE html>
<html> <head> <style>
.flex-container {
   display: -webkit-flex;
 -webkit-flex-direction: row-reverse;
    display: flex;
    width: 400px;
    height: 200px;
    background-color: lightpink;  }
.flex-item {
    background-color: cyan;
    width: 100px;
    height: 100px;
    margin: 10px;  }
</style>
</head>
<body>
<div class="flex-container">
  <div class="flex-item">flex item 1</div>
  <div class="flex-item">flex item 2</div>
  <div class="flex-item">flex item 3</div>
</div> </body> </html>
```



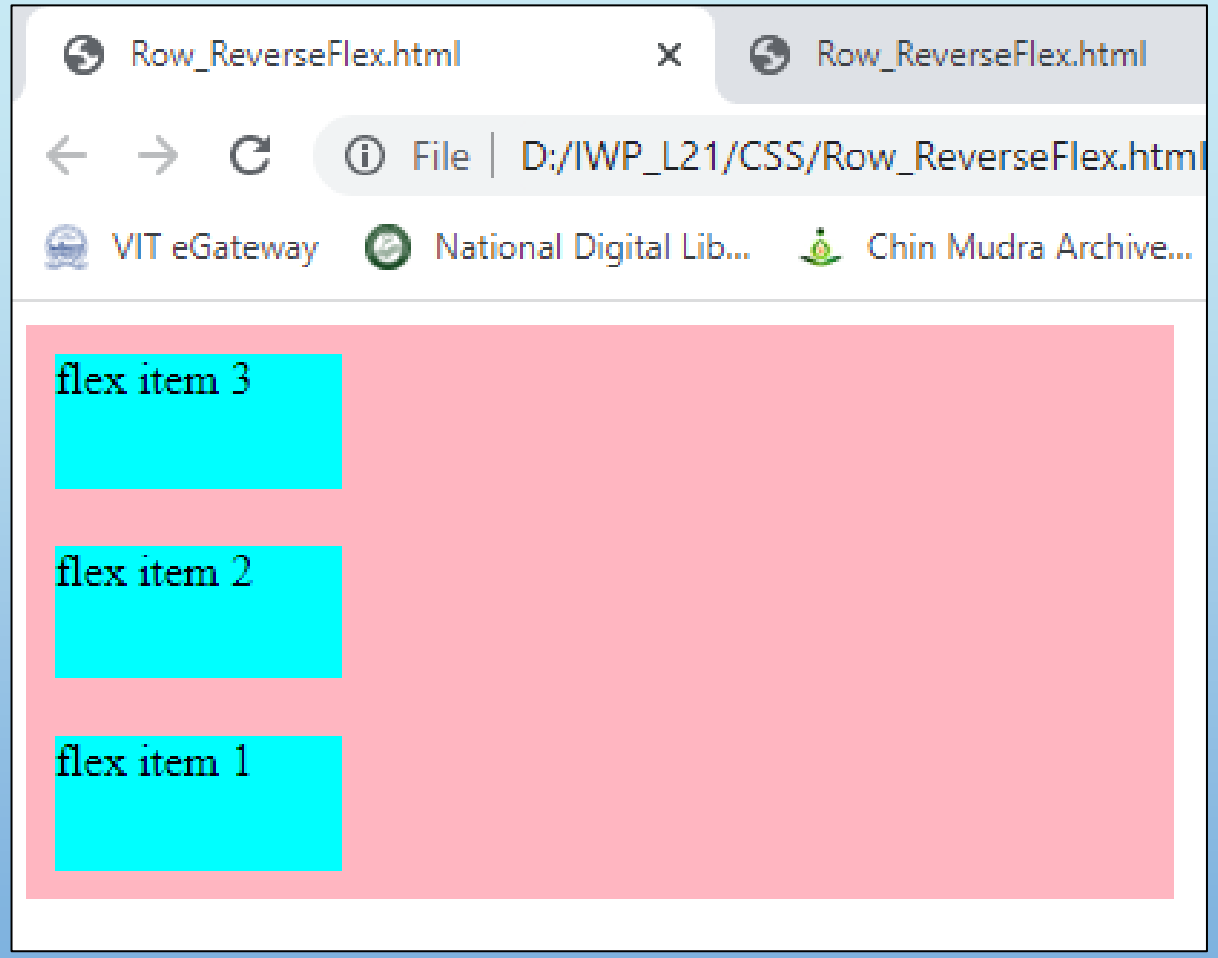
CSS – Flexbox Direction (column)

```
<!DOCTYPE html>
<html> <head> <style>
.flex-container {
   display: -webkit-flex;
 -webkit-flex-direction: column;
    display: flex;
    width: 400px;
    height: 200px;
    background-color: lightpink;  }
.flex-item {
    background-color: cyan;
    width: 100px;
    height: 100px;
    margin: 10px;  }
</style>
</head>
<body>
<div class="flex-container">
  <div class="flex-item">flex item 1</div>
  <div class="flex-item">flex item 2</div>
  <div class="flex-item">flex item 3</div>
</div> </body> </html>
```



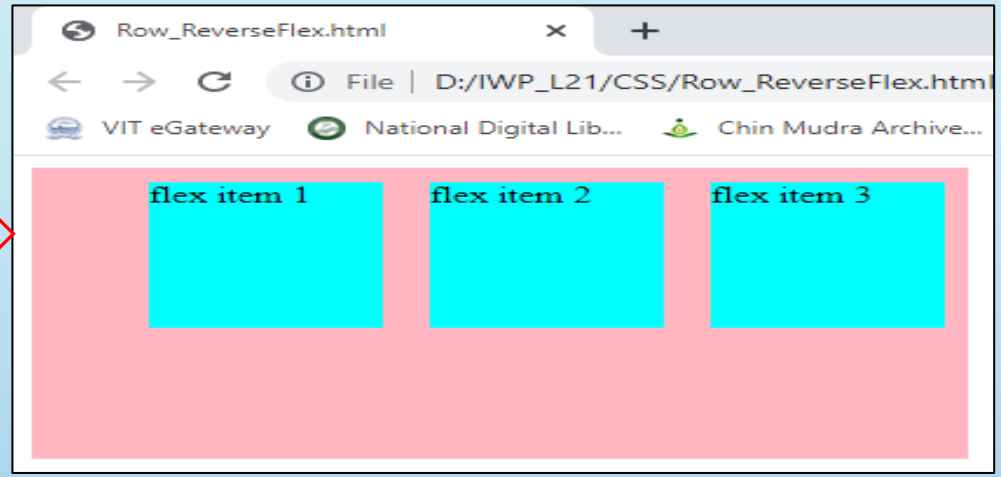
CSS – Flexbox Direction (column-reverse)

```
<!DOCTYPE html>
<html> <head> <style>
.flex-container {
   display: -webkit-flex;
 -webkit-flex-direction: column-reverse;
    display: flex;
    width: 400px;
    height: 200px;
    background-color: lightpink;  }
.flex-item {
    background-color: cyan;
    width: 100px;
    height: 100px;
    margin: 10px;  }
</style>
</head>
<body>
<div class="flex-container">
  <div class="flex-item">flex item 1</div>
  <div class="flex-item">flex item 2</div>
  <div class="flex-item">flex item 3</div>
</div> </body> </html>
```

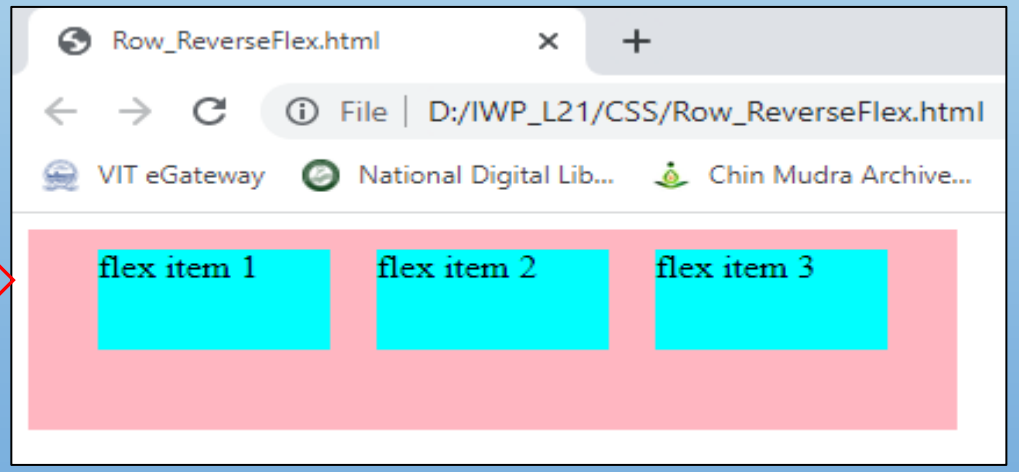


CSS – Flex Justify Content

```
.flex-container {
    display: -webkit-flex;
    display: flex;
    -webkit-justify-content: flex-end;
    justify-content: flex-end;
    width: 400px;
    height: 100px;
    background-color: lightpink;
}
```

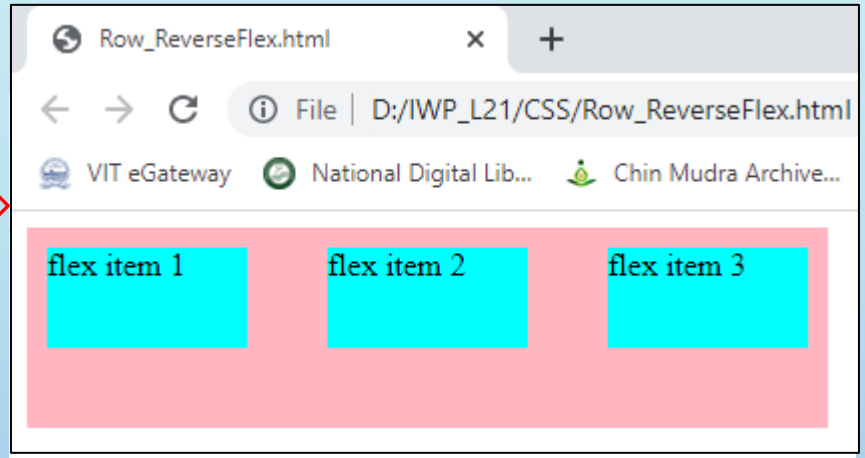



```
.flex-container {
    display: -webkit-flex;
    display: flex;
    -webkit-justify-content: center;
    justify-content: center;
    width: 400px;
    height: 100px;
    background-color: lightpink;
}
```

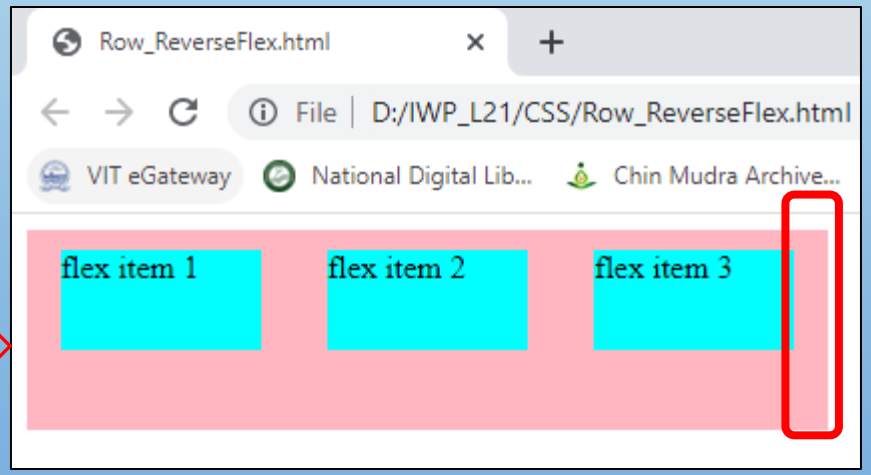


CSS – Flex Justify Content

```
.flex-container {
    display: -webkit-flex;
    display: flex;
    -webkit-justify-content: space-between;
    justify-content: space-between;
    width: 400px;
    height: 100px;
    background-color: lightpink;
}
```

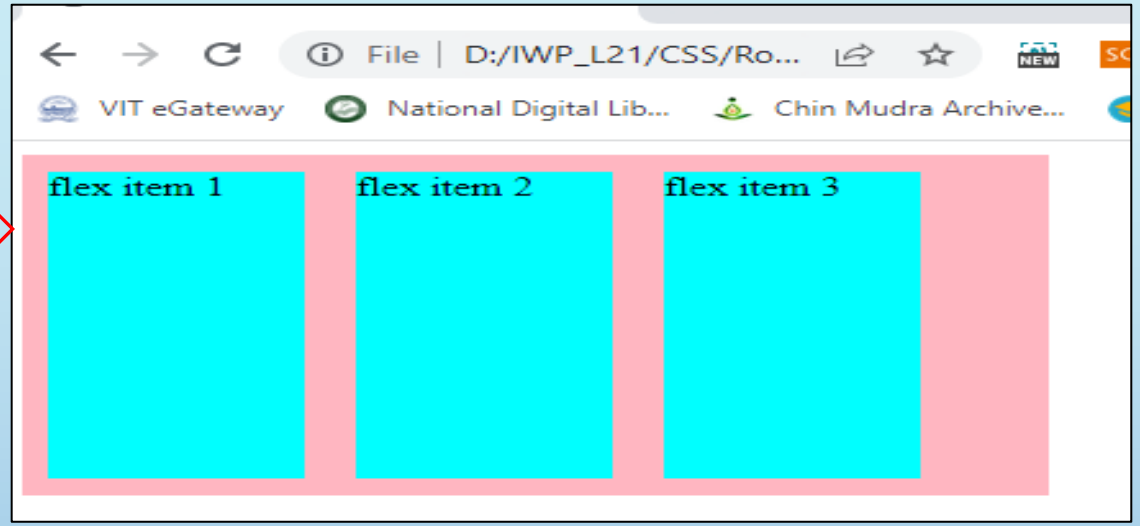



```
.flex-container {
    display: -webkit-flex;
    display: flex;
    -webkit-justify-content: space-around;
    justify-content: space-around;
    width: 400px;
    height: 100px;
    background-color: lightpink;
}
```

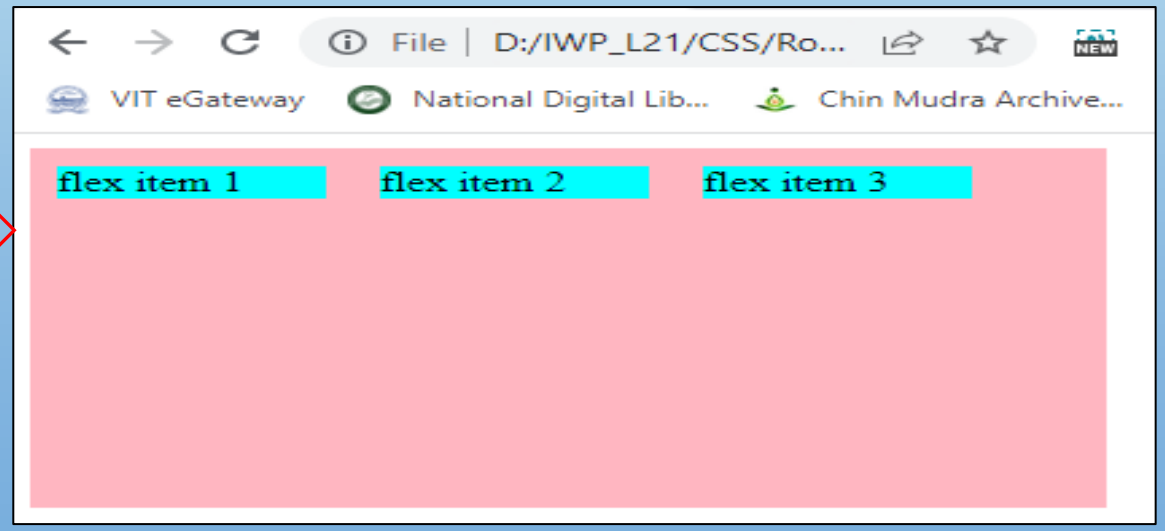


CSS – Flex Align Items

```
.flex-container {
    display: -webkit-flex;
    display: flex;
    -webkit-align-items: stretch;
    align-items: stretch;
    width: 400px;
    height: 200px;
    background-color: lightpink;
}
```

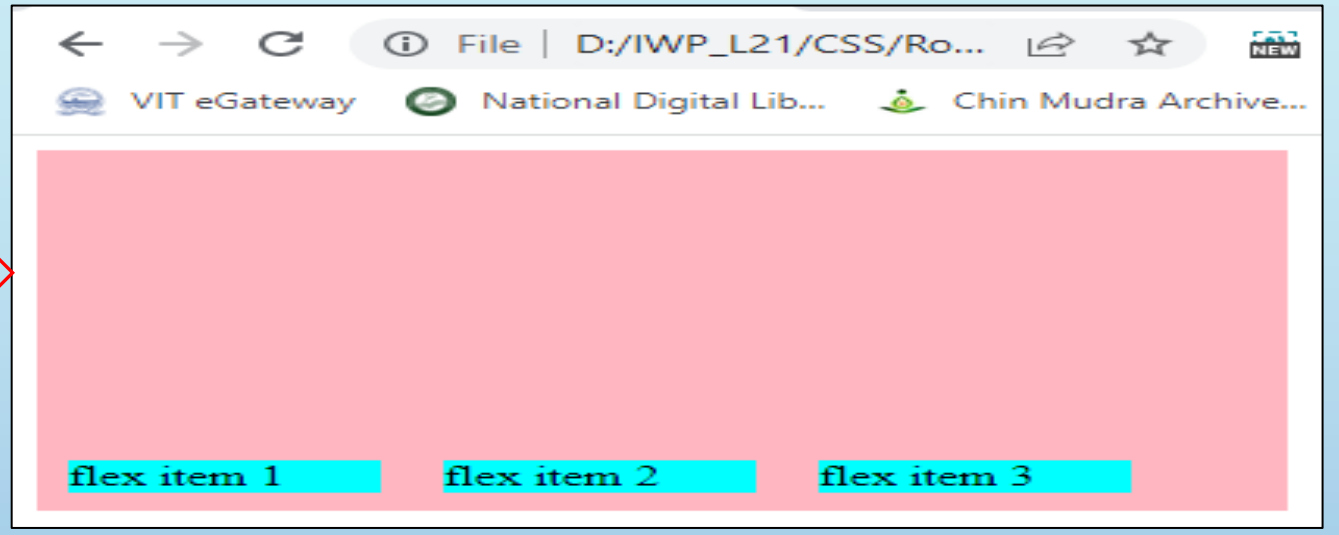



```
.flex-container {
    display: -webkit-flex;
    display: flex;
    -webkit-align-items: flex-start;
    align-items: flex-start;
    width: 400px;
    height: 200px;
    background-color: lightpink;
}
```

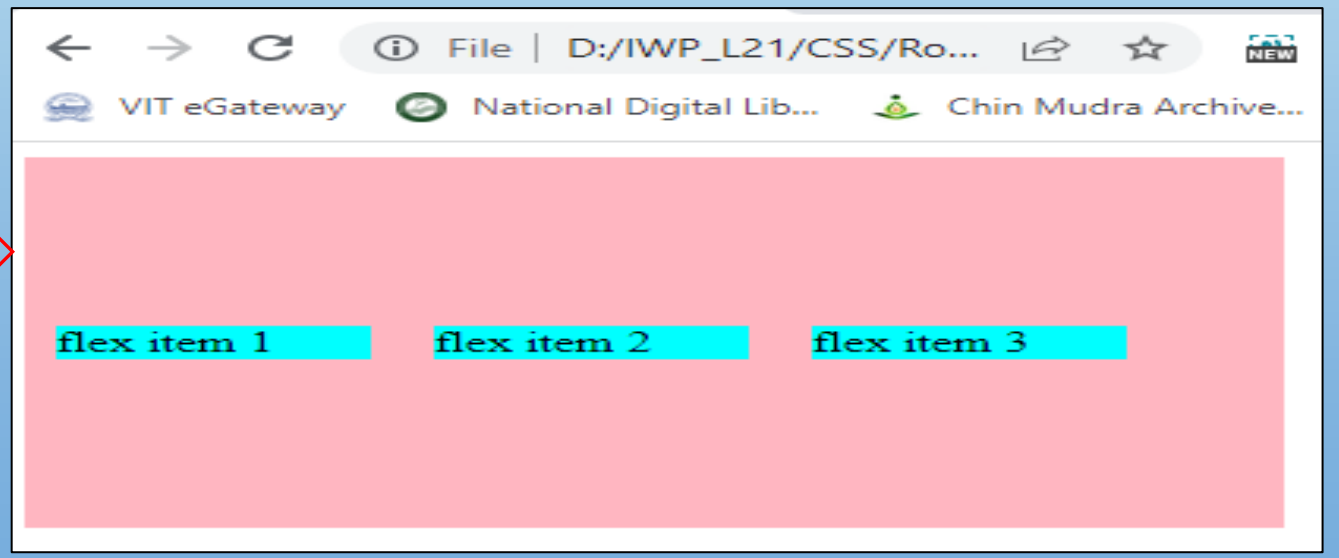


CSS – Flex Align Items

```
.flex-container {
    display: -webkit-flex;
    display: flex;
    -webkit-align-items: flex-end;
    align-items: flex-end;
    width: 400px;
    height: 200px;
    background-color: lightpink;
}
```

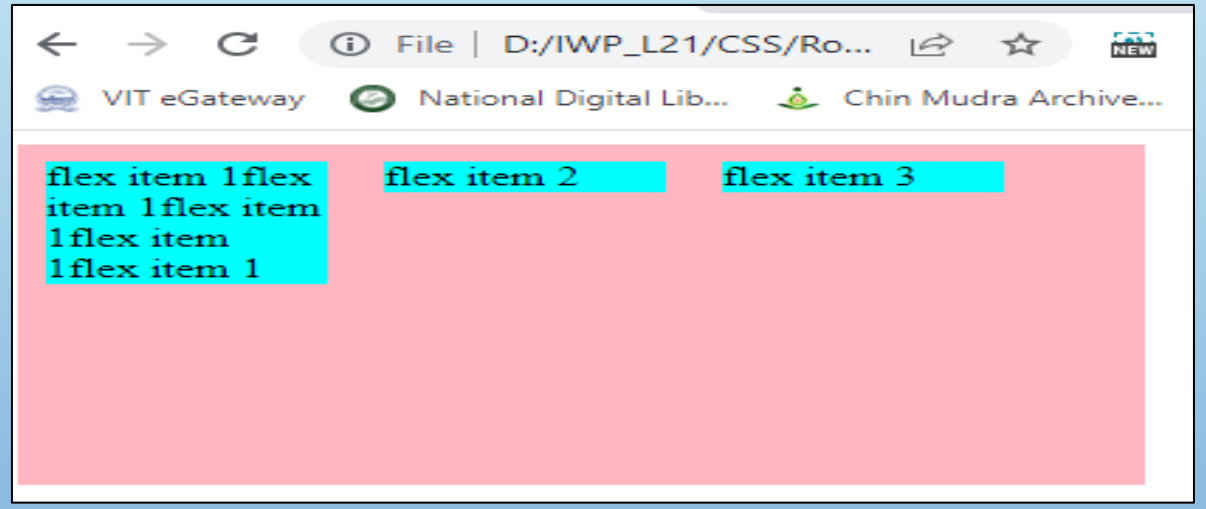



```
.flex-container {
    display: -webkit-flex;
    display: flex;
    -webkit-align-items: center;
    align-items: center;
    width: 400px;
    height: 200px;
    background-color: lightpink;
}
```



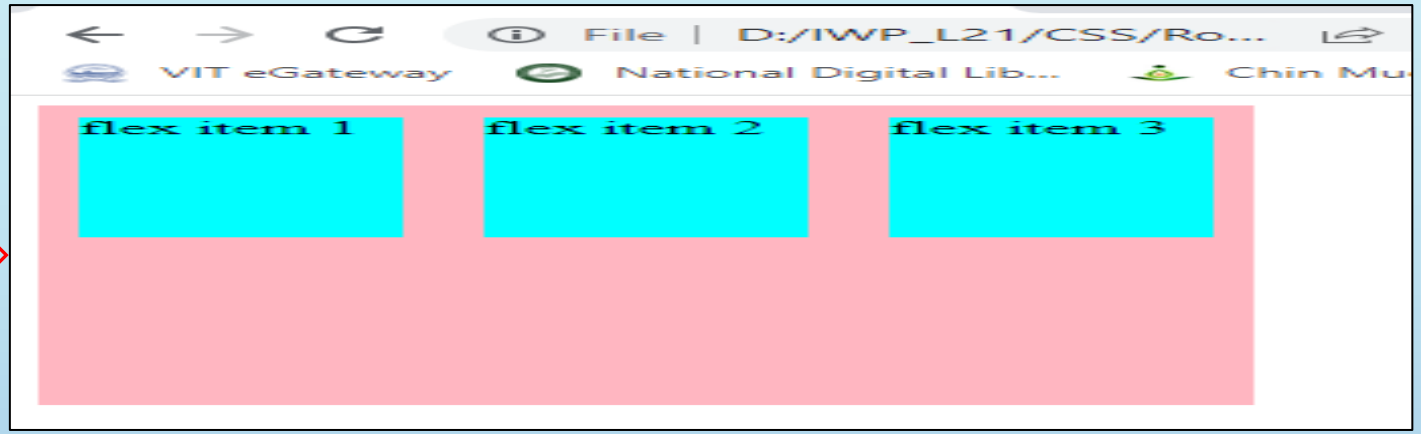
CSS – Flex Align Items

```
.flex-container {
    display: -webkit-flex;
    display: flex;
    -webkit-align-items: baseline;
    align-items: baseline;
    width: 400px;
    height: 200px;
    background-color: lightpink;
}
```

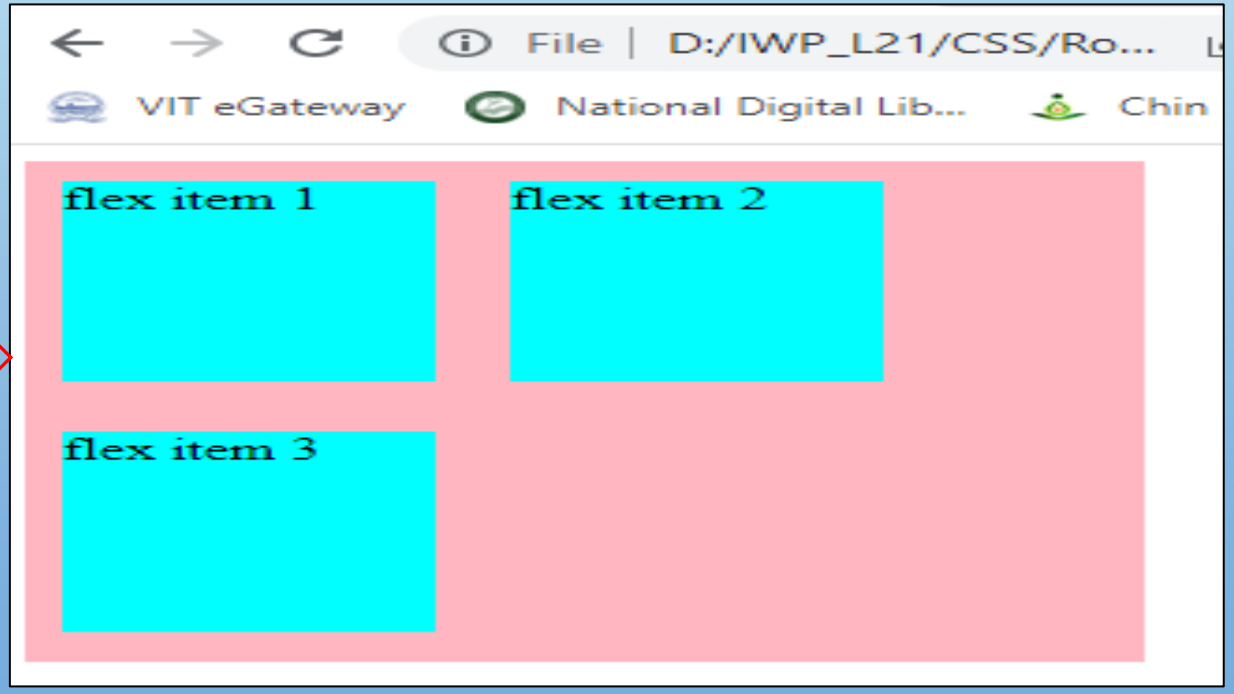


CSS – Flex Wrap

```
.flex-container {
    display: -webkit-flex;
    display: flex;
    -webkit-flex-wrap: nowrap;
    flex-wrap: nowrap;
    width: 300px;
    height: 250px;
    background-color: lightpink;
}
```




```
.flex-container {
    display: -webkit-flex;
    display: flex;
    -webkit-flex-wrap: wrap;
    flex-wrap: wrap;
    width: 300px;
    height: 250px;
    background-color: lightpink;
}
```



CSS – Flex Wrap

```
<!DOCTYPE html>
<html>
<head>
<style>
.flex-container {
    display: flex;
    flex-wrap: wrap-reverse;
    width: 300px;
    height: 250px;
    background-color: lightpink;
}
.flex-item {
    background-color: cyan;
    width: 100px;
    height: 100px;
    margin: 10px;
}
</style>
</head>
<body>
<div class="flex-container">
  <div class="flex-item">flex item 1</div>
  <div class="flex-item">flex item 2</div>
  <div class="flex-item">flex item 3</div>
</div> </body> </html>
```