Lesson:

Flex container properties

PW SKILLS

What we will learn

- What is Flexbox
- Understanding the align-items
- align-content
- justify-content
- gap
- flex-wrap
- flex-direction
- flex-flowtwo main axes of flexbox (main axis and cross axis)

Let us first see how we can make a container flex

```
Unset
.parent {
  display: flex;
}

<div class="parent">
  <div>Child 1</div>
  <div>Child 2</div>
  <div>Child 3</div>
</div>
```

In the code above, we can see we have a div with class parent, we are targeting this class and setting display: flex; now all the elements inside this parent div are flex items.

We can manipulate these flex containers using the different flex properties that are available. We will look at those properties one by one

align-items:

It is used to align the flex items along the cross-axis(vertical) of the container within the flexbox layout.

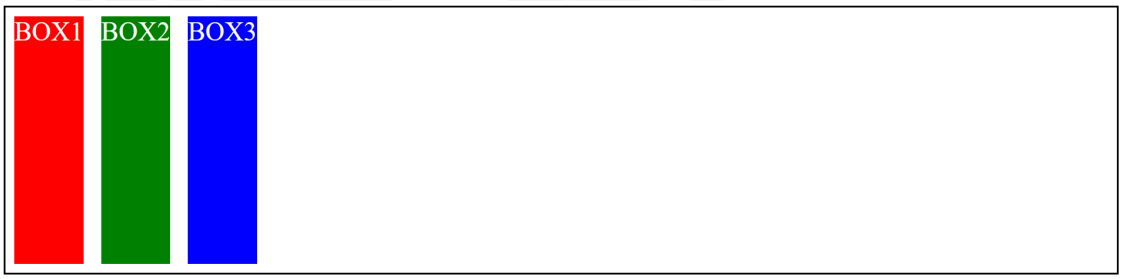
It can have these possible values:

1. **stretch:** This is the default value of the align-items which allows the flex items to stretch themselves to fit the height of the flex container.

Example Code without stretch

```
Unset
  <style>
    .box {
      font-size: 30px;
      color: white;
      margin: 10px;
    }
    .box1 {
      background-color: red;
    }
    .box2 {
      background-color: green;
    }
    .box3 {
      background-color: blue;
    }
    .container {
      height: 300px;
      border: 2px solid black;
      display: flex;
    }
  </style>

  <body>
    <div class="container">
      <div class="box box1">BOX1</div>
      <div class="box box2">BOX2</div>
      <div class="box box3">BOX3</div>
    </div>
  </body>
```

Output

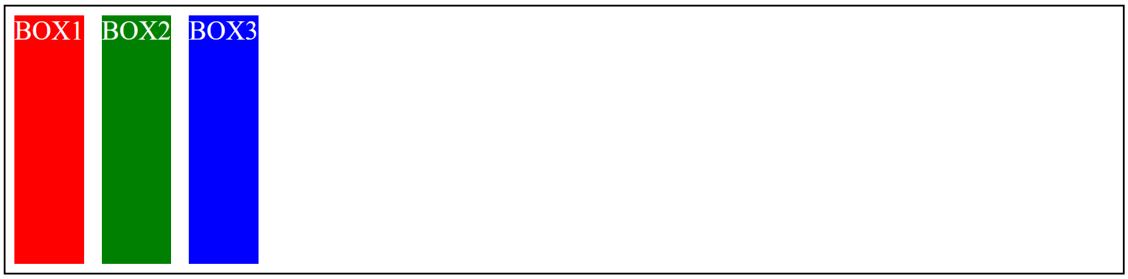


Example Code with align-items:stretch

In the previous example code if we change the .container class with the following code

```
Unset
.container {
      height: 300px;
      border: 2px solid black;
      display: flex;
}
```

The output would look as shown below

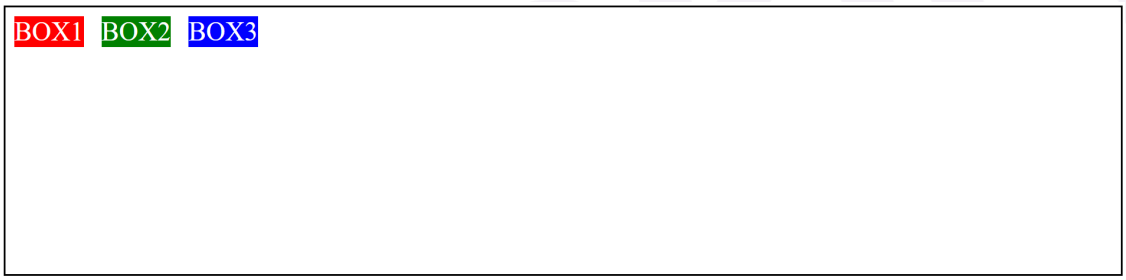


2. **flex-start:** This value is used to align all the flex items to the start of the cross axis.

Ex: If we modify the .container class from the previous code with the following properties

```
Unset
.container {
      height: 300px;
      border: 2px solid black;
      display: flex;
      align-items: flex-start;
}
```

The output would look like this

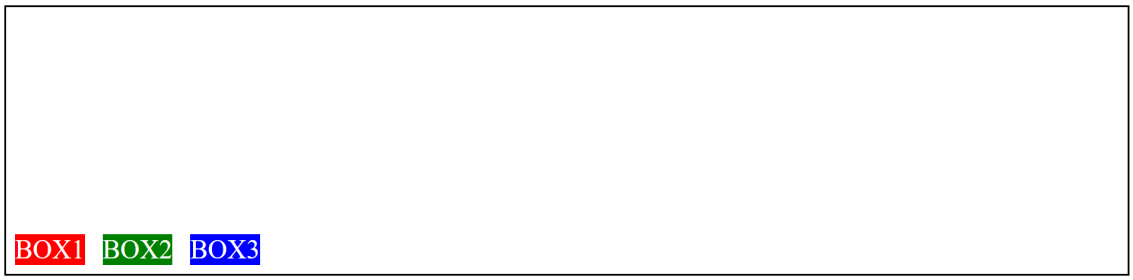


3. **flex-end:** This value is used to align all the flex items to the end of the cross axis.

Ex: If we modify the .container class from the previous code with the following properties

```
Unset
.container {
      height: 300px;
      border: 2px solid black;
      display: flex;
      align-items: flex-end;
}
```

The output would look like this

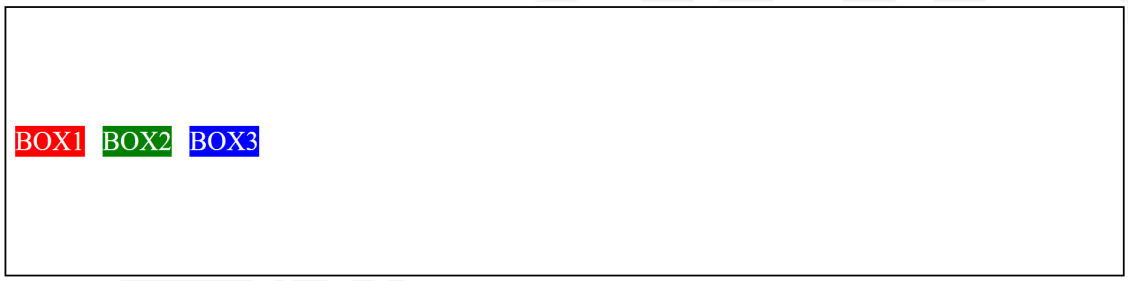


4. **center:** This value aligns all the flex items to the center of the cross axis.

Ex: If we modify the .container class from the previous code with the following properties

```
Unset
.container {
      height: 300px;
      border: 2px solid black;
      display: flex;
      align-items: center;
}
```

The output would look like this



5. **baseline:** This value aligns the flex items at the baseline of the cross-axis. This is useful when the flex items have a baseline to align to.

Ex:

```
    }
    .box3 {
      background-color: blue;
    }
    .container {
      height: 200px;
      border: 2px solid black;
      display: flex;
      margin-bottom: 10px;
    }
    .container1 {
      align-items: flex-start;
    }
    .container2 {
      align-items: baseline;
    }
  </style>
  <body>
    <div class="container container1">
      <img
        width="100px"

src="https://www.pw.live//version14/assets/img/logo.png"
        alt="logo"
      />
      <div class="box box1">BOX1</div>
      <div class="box box2">BOX2</div>
      <div class="box box3">BOX3</div>
    </div>
    <div class="container container2">
      <img
        width="100px"
src="https://www.pw.live//version14/assets/img/logo.png"
        alt="logo"
      />
      <div class="box box1">BOX4</div>
      <div class="box box2">BOX5</div>
      <div class="box box3">BOX6</div>
    </div>
  </body>
```

Output for the above would look like this

justify-content:

It is used to align the flex items along with the main axis of the container within the flexbox layout.

It can have these possible values-

1. **flex-start:** This is the default alignment value of the flex-items. It positions the flex items at the start of the main axis in the flex container.

Ex:

```
Unset
<style>
    .box {
      height: 100px;
      width: 100px;
      font-size: 30px;
      color: white;
      margin: 10px;
    }
    .box1 {
      background-color: red;
    }
    .box2 {
      background-color: green;
    }
    .box3 {
      background-color: blue;
    }
    .container {
      border: 2px solid black;
      display: flex;
    }
```

```
</style>
<body>
  <div class="container">
    <div class="box box1">BOX1</div>
    <div class="box box2">BOX2</div>
    <div class="box box3">BOX3</div>
  </div>
</body>
```

Output

Example Code with justify-content:flex-start

Unset

```
<style>
    .box {
      height: 100px;
      width: 100px;
      font-size: 30px;
      color: white;
      margin: 10px;
    }
    .box1 {
      background-color: red;
    }
    .box2 {
      background-color: green;
    }
    .box3 {
      background-color: blue;
    }
    .container {
      border: 2px solid black;
      display: flex;
      justify-content: flex-start;
    }
```

```
  </style>
  <body>
    <div class="container">
      <div class="box box1">BOX1</div>
      <div class="box box2">BOX2</div>
      <div class="box box3">BOX3</div>
    </div>
  </body>
```

Output: The output with and without flex start property is the same because that is the default value of the justify content.

2. flex-end: It positions the flex items at the end of the main axis in the flex container.

Ex:

```
Unset
  <style>
      .box {
        height: 100px;
        width: 100px;
        font-size: 30px;
        color: white;
        margin: 10px;
      }
      .box1 {
        background-color: red;
      }
      .box2 {
        background-color: green;
      }
```

```
    .box3 {
      background-color: blue;
    }
    .container {
      border: 2px solid black;
      display: flex;
      justify-content: flex-end;
    }
  </style>
  <body>
    <div class="container">
      <div class="box box1">BOX1</div>
      <div class="box box2">BOX2</div>
      <div class="box box3">BOX3</div>
    </div>
  </body>
```

Output:

3. center: All the flex items will be in the center of the main-axis in the flex container.

Ex:

```
Unset
<style>
    .box {

      height: 100px;
      width: 100px;
      font-size: 30px;
      color: white;
      margin: 10px;
    }
    .box1 {
      background-color: red;
    }
    .box2 {
      background-color: green;
    }
```

```
    .box3 {
      background-color: blue;
    }
    .container {
      border: 2px solid black;
      display: flex;
      justify-content: center;
    }
  </style>
  <body>
    <div class="container">
      <div class="box box1">BOX1</div>
      <div class="box box2">BOX2</div>
      <div class="box box3">BOX3</div>
    </div>
  </body>
```

Output:

4. **space-between:** All the flex items are evenly distributed within the flex container in which the first flex item will be aligned in the start of the main axis and the last one in the end of the main axis.

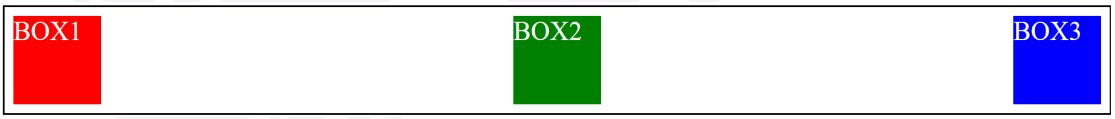
Ex:

Unset

```
<style>
    .box {
      height: 100px;
      width: 100px;
      font-size: 30px;
      color: white;
      margin: 10px;
    }
    .box1 {
      background-color: red;
    }
```

```
    }
    .box2 {
      background-color: green;
    }
    .box3 {
      background-color: blue;
    }
    .container {
      border: 2px solid black;
      display: flex;
      justify-content: space-between;
    }
  </style>
  <body>
     <div class="container">
       <div class="box box1">BOX1</div>
       <div class="box box2">BOX2</div>
       <div class="box box3">BOX3</div>
     </div>
   </body>
```

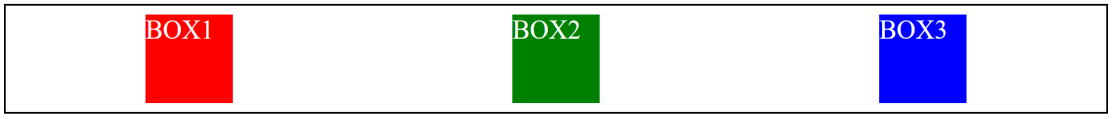
Output:



5. **space-around:** All the flex items are evenly distributed within the container, in which all the flex items have an equal space around each other.

```
Unset
<style>
    .box {
      height: 100px;
      width: 100px;
      font-size: 30px;
      color: white;
      margin: 10px;
    }
    .box1 {
      background-color: red;
    }
    .box2 {
      background-color: green;
   }
   .box3 {
     background-color: blue;
   }
   .container {
     border: 2px solid black;
     display: flex;
     justify-content: space-around;
   }
 </style>
 <body>
   <div class="container">
     <div class="box box1">BOX1</div>
     <div class="box box2">BOX2</div>
     <div class="box box3">BOX3</div>
   </div>
 </body>
```

Output:

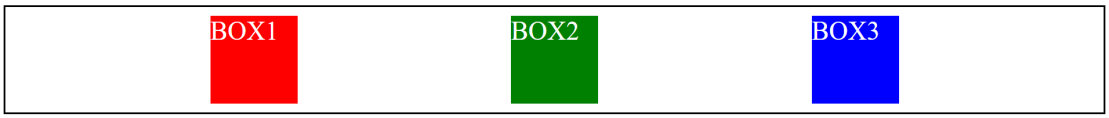


6. space-evenly: All the flex-items will have an equal distribution of space between them.

Output:

```
Unset
 <style>
   .box {
     height: 100px;
     width: 100px;
     font-size: 30px;
     color: white;
     margin: 10px;
   }
   .box1 {
     background-color: red;
   }
   .box2 {
     background-color: green;
   }
   .box3 {
     background-color: blue;
   }
    .container {
      border: 2px solid black;
      display: flex;
      justify-content: space-evenly;
    }
 </style>
 <body>
   <div class="container">
     <div class="box box1">BOX1</div>
     <div class="box box2">BOX2</div>
     <div class="box box3">BOX3</div>
   </div>
 </body>
```

Output:



align-content:

This property is used to align a flex container's lines within it when there is extra space in the cross axis, similar to that of the justify-content property which works on the main-axis. There should be more than one line to function the align content.

It can have these possible values-

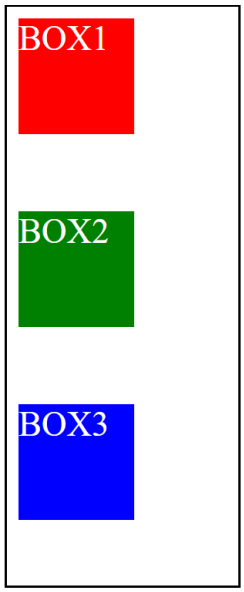
1. **stretch:** This is the default value of the align-content in which the flex items will stretch themselves to fill the whole cross axis space.

Example without align-content:stretch

```
Unset
 <style>
    .box {
      width: 100px;
      height: 100px;
      font-size: 30px;
      color: white;
      margin: 10px;
    }
    .box1 {
      background-color: red;
    }
    .box2 {
      background-color: green;

   }
   .box3 {
     background-color: blue;
   }
   .container {
     height: 500px;
     width: 200px;
     border: 2px solid black;
     display: flex;
     flex-wrap: wrap;
   }
```

```
  </style>
  <body>
    <div class="container">
      <div class="box box1">BOX1</div>
      <div class="box box2">BOX2</div>
      <div class="box box3">BOX3</div>
    </div>
  </body>
```

Output for the above code would look like this



Example with align-content:stretch

Unset

```
<style>
  .box {
    width: 100px;
    height: 100px;
    font-size: 30px;
    color: white;
    margin: 10px;
  }

  .box1 {
    background-color: red;
  }
```

```
  .box2 {
    background-color: green;
  }

  .box3 {
    background-color: blue;
  }

   .container {
     height: 500px;
     width: 200px;
     border: 2px solid black;
     display: flex;
     flex-wrap: wrap;
     align-content: stretch;
   }
</style>

 <body>
   <div class="container">
     <div class="box box1">BOX1</div>
     <div class="box box2">BOX2</div>
     <div class="box box3">BOX3</div>
   </div>
 </body>
```

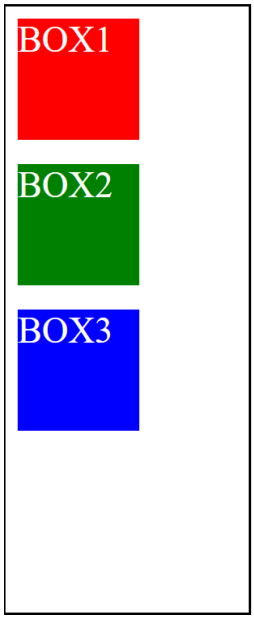
Output: The output with and without stretch value are the same because the stretch is the default value of the align-content.

1. flex-start: This value aligns all the flex items to the start of the cross-axis of the flex container.

Ex:

```
Unset
<style>
    .box {
      width: 100px;
      height: 100px;

      font-size: 30px;
      color: white;
      margin: 10px;
    }
    .box1 {
      background-color: red;
    }
    .box2 {
      background-color: green;
    }
    .box3 {
      background-color: blue;
    }
    .container {
      height: 500px;
      width: 200px;
      border: 2px solid black;
      display: flex;
      flex-wrap: wrap;
      align-content: flex-start;
    }
  </style>
  <body>
    <div class="container">
      <div class="box box1">BOX1</div>
      <div class="box box2">BOX2</div>
      <div class="box box3">BOX3</div>
    </div>
  </body>
```

Output

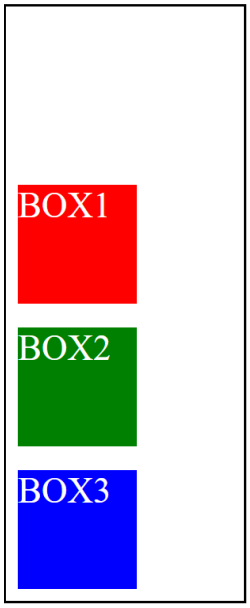


3. flex-end: This value aligns all the flex items to the end of the cross axis of the flex container.

Ex:

```
Unset
<style>
    .box {
      width: 100px;
      height: 100px;
      font-size: 30px;
      color: white;

     margin: 10px;
   }
   .box1 {
     background-color: red;
   }
   .box2 {
     background-color: green;
   }
   .box3 {
     background-color: blue;
   }
```

Output for the above code would look like this



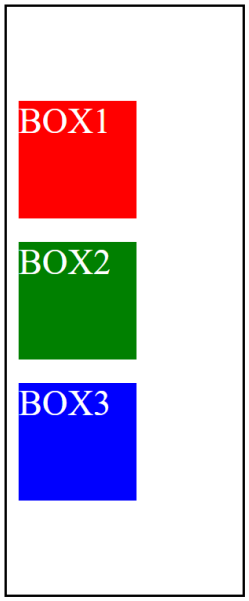
4. center: This value aligns all the flex items to the center of the cross axis of the flex container.

Ex:

```
Unset
  <style>
    .box {
      width: 100px:
      height: 100px;
      font-size: 30px;
      color: white;
      margin: 10px;
    }
    .box1 {
      background-color: red;
    }
    .box2 {
      background-color: green;
    }
    .box3 {
      background-color: blue;
    }
```

```
    .container {
      height: 500px;
      width: 200px;
      border: 2px solid black;
      display: flex;
      flex-wrap: wrap;
      align-content: center;
    }
  </style>
  <body>
    <div class="container">
      <div class="box box1">BOX1</div>
      <div class="box box2">BOX2</div>
      <div class="box box3">BOX3</div>
    </div>
  </body>
```

Output for the above code would look like this

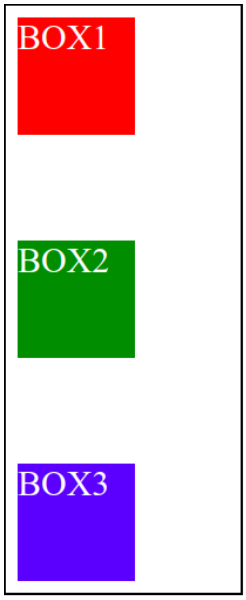


5. **space-between:** All the flex items are evenly distributed within the flex container in which the first flex item will be aligned at the start of the cross axis and the last one at the end of the cross axis.

Ex:

```
Unset
<style>
    .box {
      width: 100px;
      height: 100px;
      font-size: 30px;
      color: white;
      margin: 10px;
    }
    .box1 {
      background-color: red;
    }
    .box2 {
      background-color: green;
    }
    .box3 {
      background-color: blue;
    }
    .container {
      height: 500px;
      width: 200px;
      border: 2px solid black;
      display: flex;
      flex-wrap: wrap;
      align-content: space-between;
    }
</style>
<body>
  <div class="container">
    <div class="box box1">BOX1</div>
    <div class="box box2">BOX2</div>
    <div class="box box3">BOX3</div>
  </div>
</body>
```

Output:



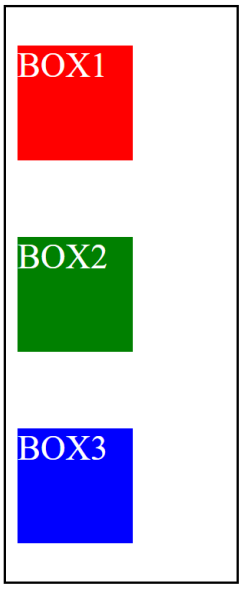
6. **space-around:** All the flex items are evenly distributed within the container on the cross axis, in which all the flex items have an equal space around each other.

Ex:

```
Unset
 <style>
    .box {
      width: 100px;
      height: 100px;
      font-size: 30px;
      color: white;
      margin: 10px;
    }
    .box1 {
      background-color: red;
    }
    .box2 {
      background-color: green;
    }
    .box3 {
      background-color: blue;
    }
```

```
    .container {
      height: 500px;
      width: 200px;
      border: 2px solid black;
      display: flex;
      flex-wrap: wrap;
      align-content: space-around;
    }
  </style>
  <body>
    <div class="container">
      <div class="box box1">BOX1</div>
      <div class="box box2">BOX2</div>
      <div class="box box3">BOX3</div>
    </div>
  </body>
```

Output:



Example Code without flex-direction: row

```
Unset
<style>
  .box {
    width: 100px;
    height: 100px;
    color: white;
    font-size: 35px;
    margin: 10px;
  }

 .box1 {
   background-color: red;
 }

 .box2 {
   background-color: green;
 }

 .box3 {
   background-color: blue;
 }

 .container {
   display: flex;
   border: 2px solid black;
 }
</style>

<body>
  <div class="container">
    <div class="box box1">BOX1</div>
    <div class="box box2">BOX2</div>
    <div class="box box3">BOX3</div>
  </div>
</body>
```

Output without flex-direction: row

Example Code with flex-direction: row

```
Unset
<style>
  .box {
    width: 100px;
    height: 100px;
    color: white;
    font-size: 35px;
   margin: 10px;
 }

 .box1 {
   background-color: red;
 }

 .box2 {
   background-color: green;
 }

 .box3 {
   background-color: blue;
 }
  .container {
    display: flex;
    flex-direction: row;
    border: 2px solid black;
  }
</style>
```

```
<body>
  <div class="container">
    <div class="box box1">BOX1</div>
    <div class="box box2">BOX2</div>
    <div class="box box3">BOX3</div>
  </div>
</body>
```

Output with flex-direction: row; We can clearly see that the output without and with flex-direction: row are the same because the row is the default direction for flex container.

2. **row-reverse:** In this, the elements are aligned horizontally in a row from right to left.

Ex:

Unset

```
<style>
  .box {
    width: 100px;
    height: 100px;
    color: white;
    font-size: 35px;
    margin: 10px;
  }

  .box1 {
    background-color: red;
  }

  .box2 {
    background-color: green;
  }

  .box3 {
    background-color: blue;
  }
```

```
 .container {
   display: flex;
   flex-direction: row-reverse;
   border: 2px solid black;
 }
</style>

<body>
  <div class="container">
    <div class="box box1">BOX1</div>
    <div class="box box2">BOX2</div>
    <div class="box box3">BOX3</div>
  </div>
</body>
```

Output

3. column: In this, the elements are aligned vertically in a column from the top to bottom.

When the flex-direction will be a column, then the main axis will become vertical from top to bottom and the cross axis will be horizontal from left to right.

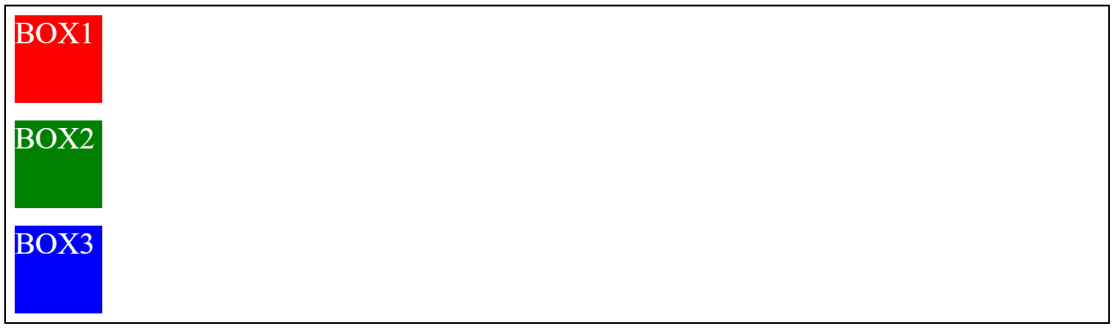
Ex:

Unset

```
<style>
  .box {
    width: 100px;
    height: 100px;
    color: white;
    font-size: 35px;
    margin: 10px;
  }
```

```
  .box1 {
    background-color: red;
  }

  .box2 {
    background-color: green;
  }

  .box3 {
    background-color: blue;
  }


  .container {
    display: flex;
    flex-direction: column;
    border: 2px solid black;
  }
</style>

<body>
  <div class="container">
    <div class="box box1">BOX1</div>
    <div class="box box2">BOX2</div>
    <div class="box box3">BOX3</div>
  </div>
</body>
```

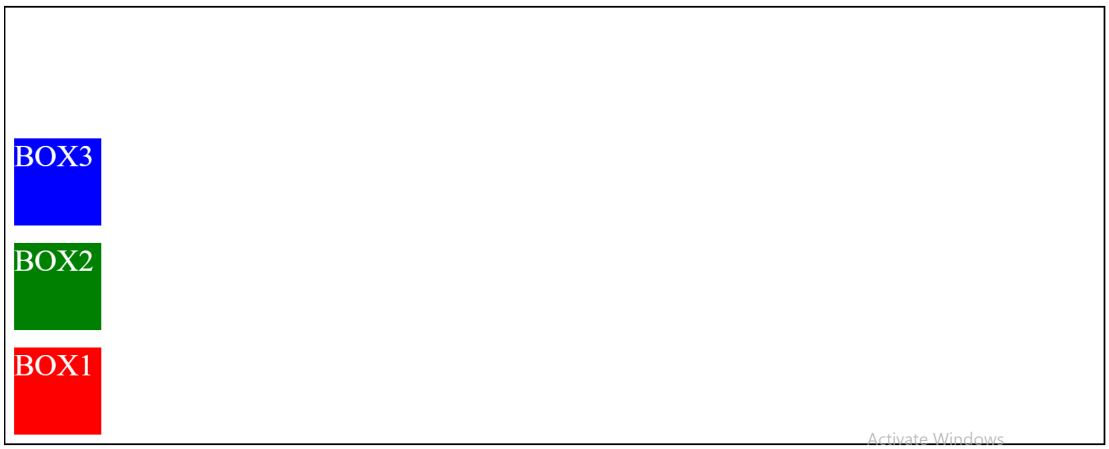
Output



Unset

```
<style>
  .box {
    width: 100px;
    height: 100px;
    color: white;
    font-size: 35px;
    margin: 10px;
  }

  .box1 {
    background-color: red;
  }

  .box2 {
    background-color: green;
  }

  .box3 {
    background-color: blue;
  }

 .container {
   height: 500px;
   display: flex;
   flex-direction: column-reverse;
   border: 2px solid black;
 }

<body>
  <div class="container">
    <div class="box box1">BOX1</div>
    <div class="box box2">BOX2</div>
    <div class="box box3">BOX3</div>
  </div>
</body>
```

Output



Gap property:

This property defines the amount of space between the rows and columns of a flex container.

When we want to give a gap between rows we use row-gap property and when we want to give a gap between columns we use column-gap property.

The gap is a shorthand property for defining the row-gap and column-gap.

Syntax

- row-gap: 10px;
- column-gap: 10px;
- gap: 10px;
  This is similar to row-gap:10px and column-gap:10px
- gap: 10px 20px
  This is similar to row-gap:10px and column-gap:20px

Example Code for row and column gap

```
Unset
<style>
  .box {
    height: 50px;
    width: 50px;
    font-size: 30px;
    color: white;
    background-color: gray;
  }
```

```
  .container {
    border: 2px solid black;
    margin-bottom: 10px;
    display: flex;
  }
   .container1 {
     column-gap: 20px;
   }

   .container2 {
     flex-direction: column;
     row-gap: 20px;
   }
</style>

<body>
  <div class="container container1">
    <div class="box">1</div>
    <div class="box">2</div>
  </div>

  <div class="container container2">
    <div class="box">3</div>
    <div class="box">4</div>
  </div>
</body>
```

Output

1 2

3

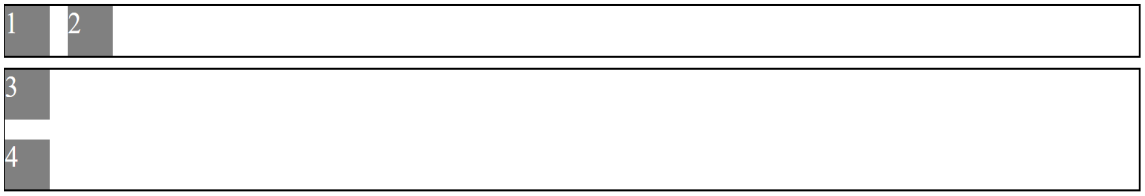
4

Note: Flex items properties will be covered in further classes.

Ex for gap:

```
Unset
<style>
  .box {
    height: 50px;
    width: 50px;
    font-size: 30px;
    color: white;
    background-color: gray;
  }

  .container {
    border: 2px solid black;
    margin-bottom: 10px;
    display: flex;
    gap: 20px;
  }

  .container2 {
    flex-direction: column;
  }
</style>

<body>
  <div class="container container1">
    <div class="box">1</div>
    <div class="box">2</div>
  </div>

  <div class="container container2">
    <div class="box">3</div>
    <div class="box">4</div>
  </div>
</body>
```

Output: Flexbox is a one-dimensional layout, that's why we cannot visualize row and column gap together because it will require a 2D structure like 'Grid' which we will learn in further classes.



flex-wrap:

This property is used to define whether the flex items should wrap themself to fit the container width or not if the total required width of flex items is more than the flex container width.

Flex wrap can have three possible values-

1. **nowrap:** This one is the default value of the flex-wrap, it does not allow the flex items to wrap to fit the flex container width. This will place all the flex items in one line even if they do not fit the container width.

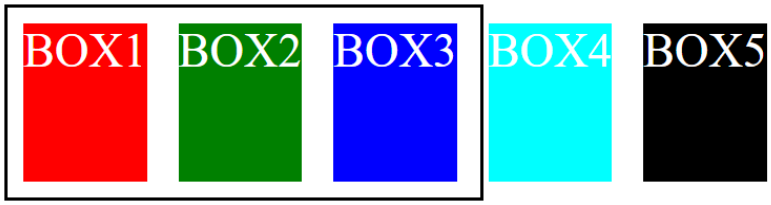
Example code without flex-wrap

```
Unset
<style>
  .box {
    width: 100px;
    height: 100px;
    font-size: 30px;
    color: white;
    margin: 10px;
  }

  .box1 {
    background-color: red;
  }

  .box2 {
    background-color: green;
  }

  .box3 {
```

```
  .container {
    border: 2px solid black;
    width: 300px;
    display: flex;
  }
</style>

 <body>
   <div class="container">
     <div class="box box1">BOX1</div>
     <div class="box box2">BOX2</div>
     <div class="box box3">BOX3</div>
     <div class="box box4">BOX4</div>
     <div class="box box5">BOX5</div>
   </div>
 </body>
```

Output without flex-wrap

As we can see in the output that the flex container has a width of 300px and the total width required for flex items is 500px but the flex items are overflowing from the container because the default value of flex-wrap is no-wrap always.



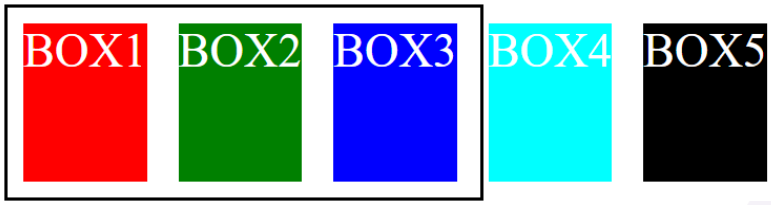
Example code with flex-wrap:nowrap

```
Unset
<style>
  .box {
    width: 100px;
    height: 100px;
    font-size: 30px;
    color: white;
    margin: 10px;
  }
  .box1 {
    background-color: red;
  }

  .box2 {
    background-color: green;
  }

  .box3 {
    background-color: blue;
  }
  .box4 {
    background-color: aqua;
  }

  .box5 {
    background-color: black;
  }

 .container {
    border: 2px solid black;
    width: 300px;
    display: flex;
    flex-wrap: nowrap;
  }
</style>
```

```
<body>
  <div class="container">
    <div class="box box1">BOX1</div>
    <div class="box box2">BOX2</div>
    <div class="box box3">BOX3</div>
    <div class="box box4">BOX4</div>
    <div class="box box5">BOX5</div>
  </div>
</body>
```

Example code with flex-wrap:nowrap

We can clearly see that the output with and without flex-wrap: nowrap; is the same because this one is the default value for a flex container.



2. **wrap:** It allows the flex items to wrap to the next line once the current line is full. The flex items can wrap themself into multiple lines if required to fit them inside the width of the flex container

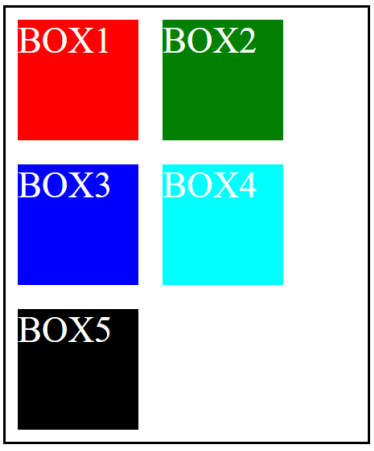
Ex:

Unset

```
<style>
  .box {
    width: 100px;
    height: 100px;
    font-size: 30px;
    color: white;
    margin: 10px;
  }
  .box1 {
    background-color: red;
  }

  .box2 {
    background-color: green;
  }

  .box3 {
```

```
    background-color: blue;
  }

  .box4 {
    background-color: aqua;
  }

  .box5 {
    background-color: black;
  }

  .container {
    border: 2px solid black;
    width: 300px;
    display: flex;
    flex-wrap: wrap;
  }
</style>

<body>
  <div class="container">
    <div class="box box1">BOX1</div>
    <div class="box box2">BOX2</div>
    <div class="box box3">BOX3</div>
    <div class="box box4">BOX4</div>
    <div class="box box5">BOX5</div>
  </div>
</body>
```

Output



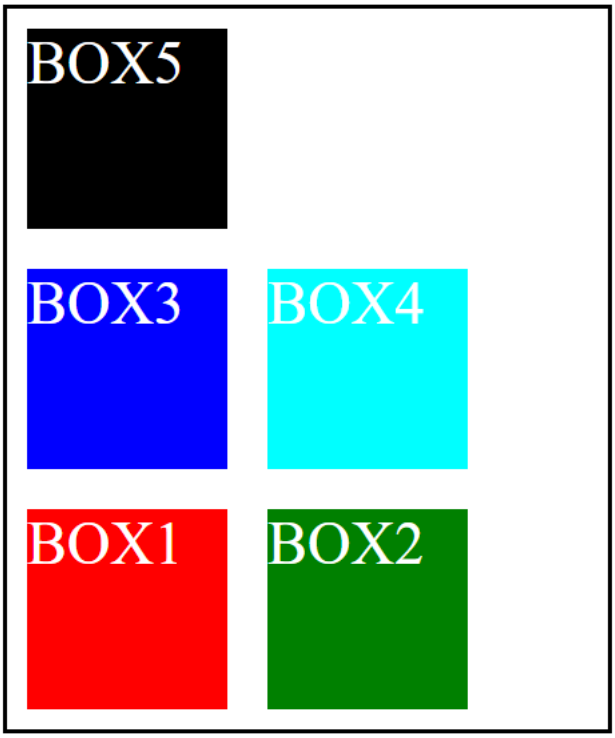
3. **wrap-reverse:** It allows the flex items to wrap to the next line once the current line is full but in the reverse order. The flex items can wrap themself into multiple lines if required to fit them inside the width of the flex container.

Ex:

```
.box2 {
  background-color: green;
}

.box3 {
  background-color: blue;
}

.box4 {
  background-color: aqua;
}

.box5 {
  background-color: black;
}

.container {
  border: 2px solid black;
  width: 300px;
  display: flex;
```

```
    flex-wrap: wrap-reverse;
  }
</style>

<body>
  <div class="container">
    <div class="box box1">BOX1</div>
    <div class="box box2">BOX2</div>
    <div class="box box3">BOX3</div>
    <div class="box box4">BOX4</div>
    <div class="box box5">BOX5</div>
  </div>
</body>
```

Output



flex-wrap:

It is a shorthand property for setting both flex-direction and flex-wrap properties in one line.

Syntax:

```
Unset
flex-flow: flex-direction flex-wrap
```

Ex:

```
Unset
<style>
  .box {
    width: 100px;
    height: 100px;
    font-size: 30px;
    color: white;
    margin: 10px;
  }

  .box1 {
    background-color: red;
  }

  .box2 {
    background-color: green;
  }

  .box3 {
    background-color: blue;

  }

  .box4 {
    background-color: aqua;
  }

  .box5 {
    background-color: black;
  `
  .container {
    border: 2px solid black;
    height: 300px;
    display: flex;
    flex-flow: column wrap;
  }
</style>
```

```
<body>
  <div class="container">
    <div class="box box1">BOX1</div>
    <div class="box box2">BOX2</div>
    <div class="box box3">BOX3</div>
    <div class="box box4">BOX4</div>
    <div class="box box5">BOX5</div>
  </div>
</body>
```