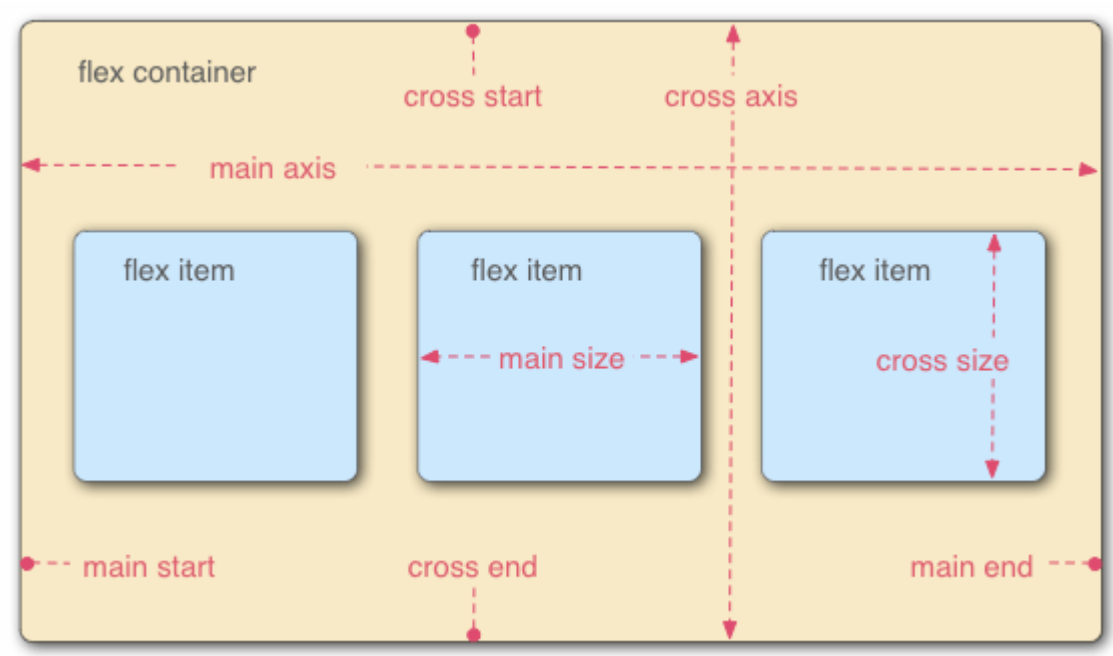
CSS Flexbox Layout Module

The Flexible Box Layout Module, makes it easier to design flexible responsive layout structure without using float or positioning.



Flexbox Elements

To start using the Flexbox model, you need to first define a flex container.

The element above represents a flex container (the blue area) with three flex items.

Example

A flex container with three flex items:

```
<div class="flex-container">
  <div>1</div>
  <div>2</div>
  <div>3</div>
```

```
</div>
```

Parent Element (Container)

This is a flex **container** (the blue area) with three flex **items**:

The flex container becomes flexible by setting the `display` property to `flex`:

The flex container properties are:

- `flex-direction`
- `flex-wrap`
- `flex-flow`
- `justify-content`
- `align-items`
- `align-content`

The flex-direction Property

1.flex-direction:column

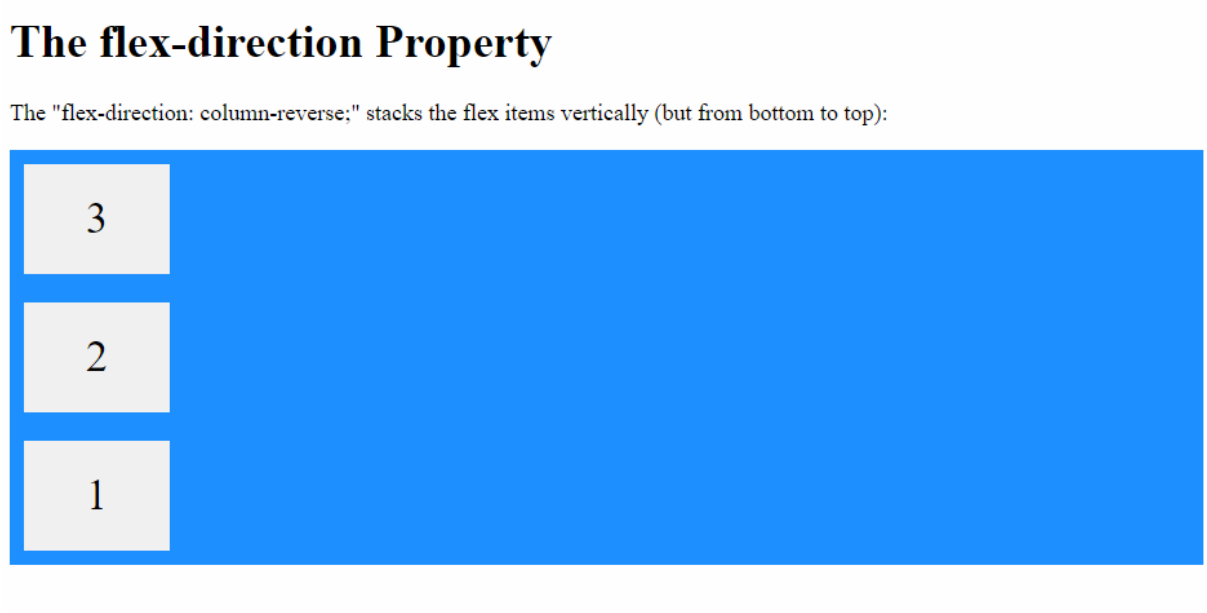
The `flex-direction` property defines in which direction the container wants to stack the flex items.

Example

The `column` value stacks the flex items vertically (from top to bottom):

```
.flex-container {
  display: flex;
  flex-direction: column;
}
```

2.flex-direction:column-reverse;

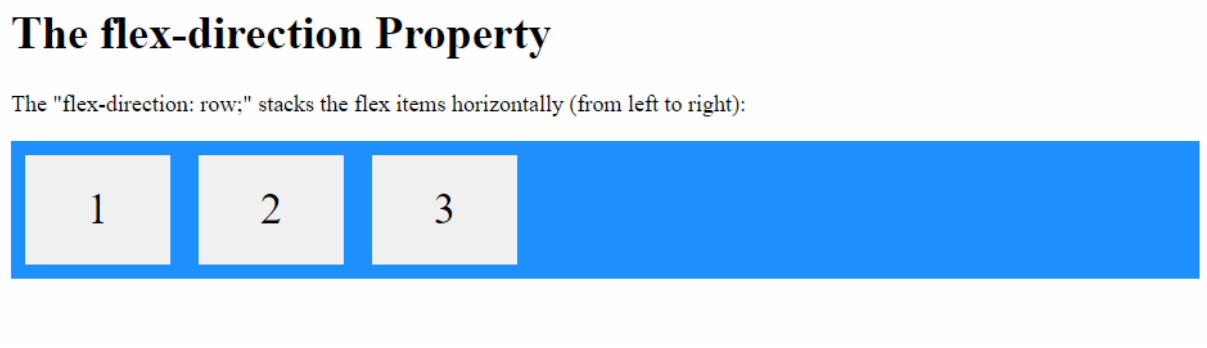


Example

The `column-reverse` value stacks the flex items vertically (but from bottom to top):

```
.container {
  display: flex;
  flex-direction: column-reverse;
}
```

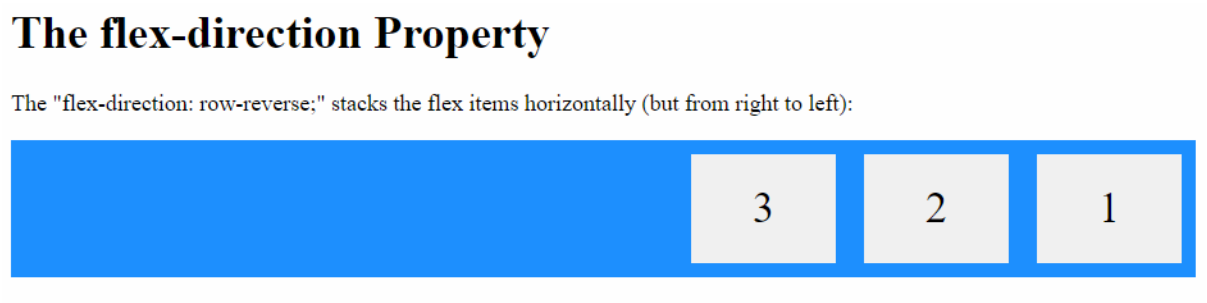
3.flex-direction:row;



The `row` value stacks the flex items horizontally (from left to right):

```
.container {
  display: flex;
  flex-direction: row;
}
```

4.flex-direction:row;



Example

The `row-reverse` value stacks the flex items horizontally (but from right to left):

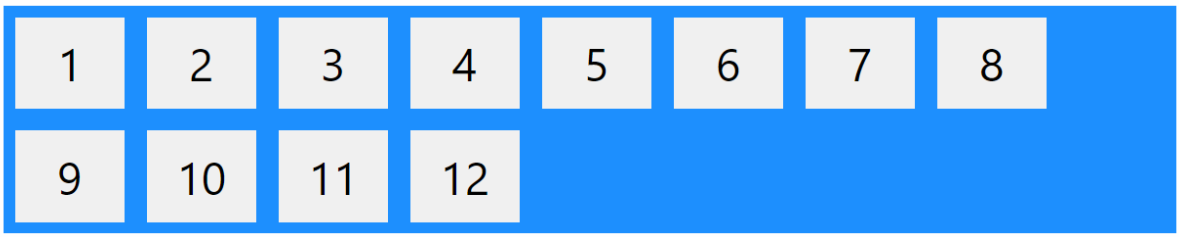
```
.container {
  display: flex;
  flex-direction: row-reverse;
}
```

The flex-wrap Property

1.flex-wrap:wrap;

The `flex-wrap` property specifies whether the flex items should wrap or not.

The examples below have 12 flex items, to better demonstrate the `flex-wrap` property.

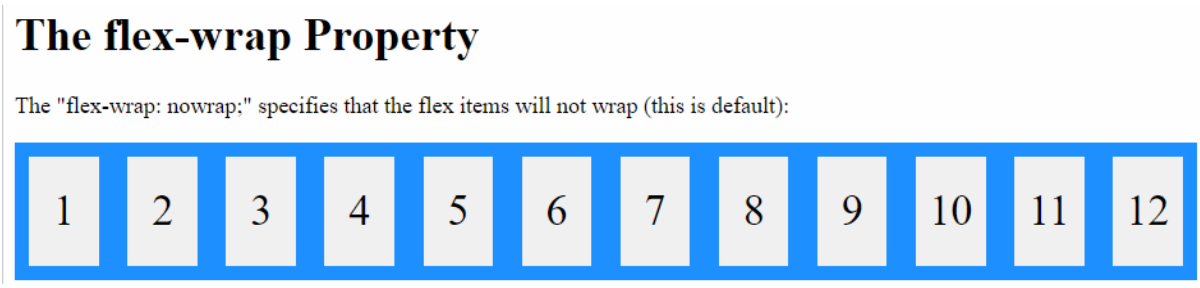


Example

The `wrap` value specifies that the flex items will wrap if necessary:

```
.container {
  display: flex;
  flex-wrap: wrap;
}
```

2.flex-wrap:nowrap(default);

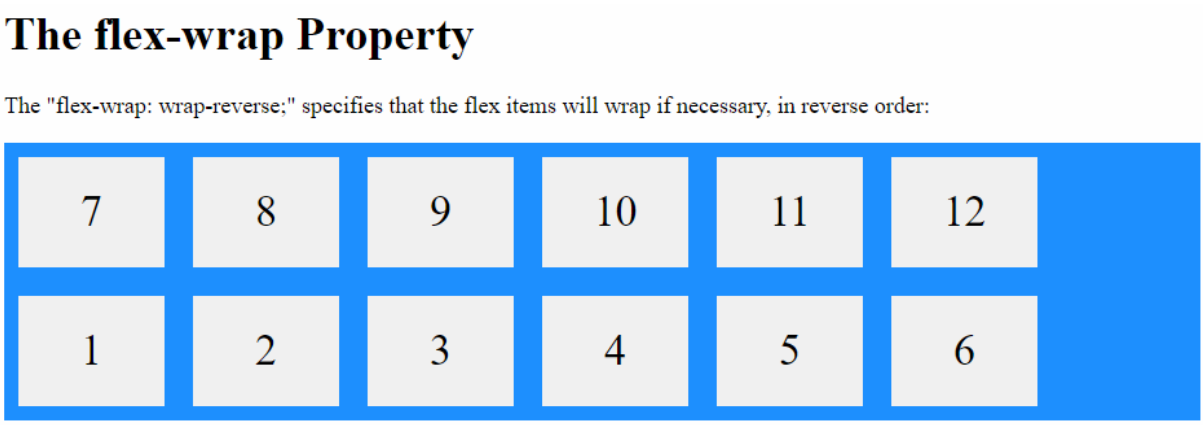


Example

The `nowrap` value specifies that the flex items will not wrap (this is default):

```
.container {
  display: flex;
  flex-wrap: nowrap;
}
```

3.flex-wrap:wrap-reverse;



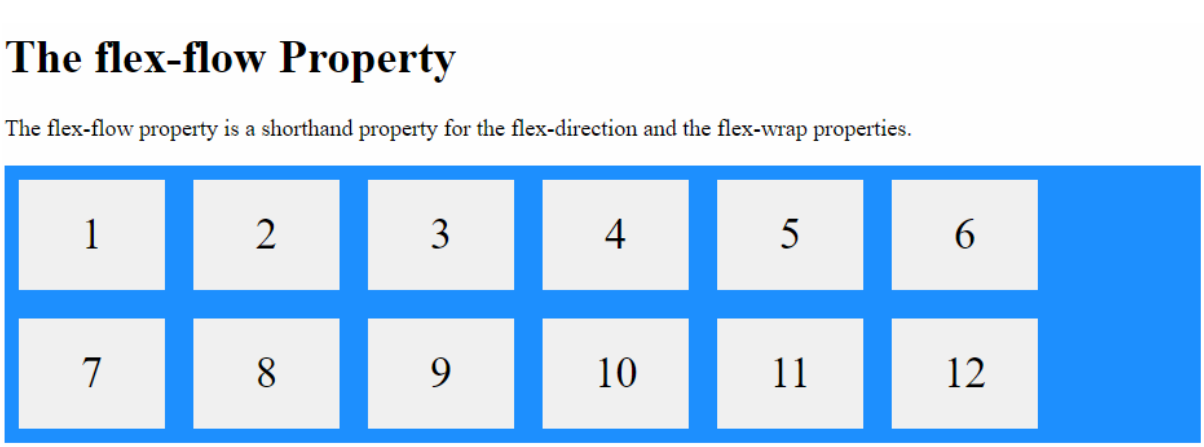
Example

The `wrap-reverse` value specifies that the flexible items will wrap if necessary, in reverse order:

```
.container {
  display: flex;
  flex-wrap: wrap-reverse;
}
```

The flex-flow Property

The `flex-flow` property is a shorthand property for setting both the `flex-direction` and `flex-wrap` properties.



Example

```
.container {
  display: flex;
  flex-flow: row wrap;
}
```

The justify-content Property

1.justify-content:center;

The `justify-content` property is used to align the flex items:

Example

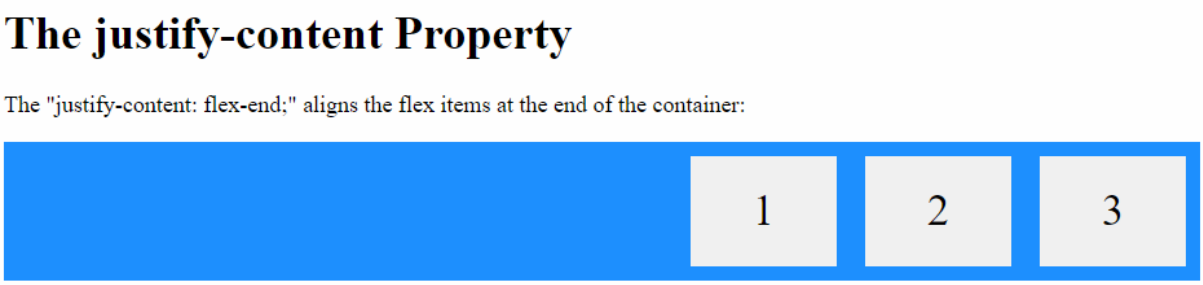
The `center` value aligns the flex items at the center of the container:

```
.container {
  display: flex;
  justify-content: center;
}
```

2.justify-content:start;


```
.container {
  display: flex;
  justify-content: flex-start;
}
```

3.justify-content:end;



Example

The `flex-end` value aligns the flex items at the end of the container:

```
.flex-container {
  display: flex;
  justify-content: flex-end;
}
```

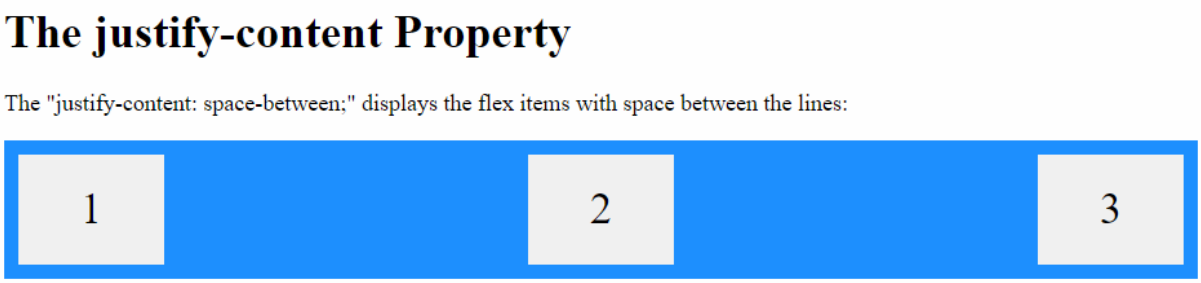
4.justify-content:space-around;

Example

The `space-around` value displays the flex items with space before, between, and after the lines:

```
.container {
  display: flex;
  justify-content: space-around;
}
```

5.justify-content:space-between;



Example

The `space-between` value displays the flex items with space between the lines:

```
.container {
  display: flex;
  justify-content: space-between;
}
```

6.justify-content:space-evenly;

Example

The `space-between` value aligns the flex items at the evenly of the container:

```
.flex-container {
  display: flex;
  justify-content: space-evenly;
}
```

The align-items Property

1.align-items:center;

The `align-items` property is used to align the flex items.

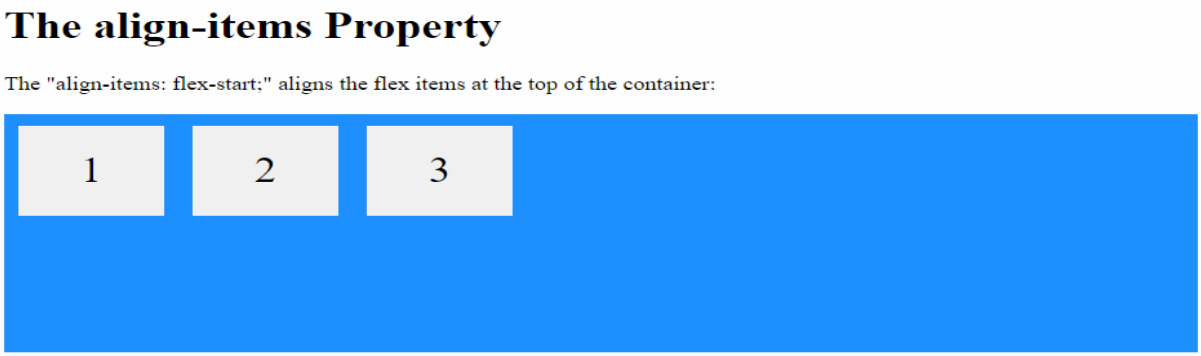
In these examples we use a 200 pixels high container, to better demonstrate the `align-items` property.

Example

The `center` value aligns the flex items in the middle of the container:

```
.container {
  display: flex;
  height: 200px;
  align-items: center;
}
```

2.align-items:start(default);



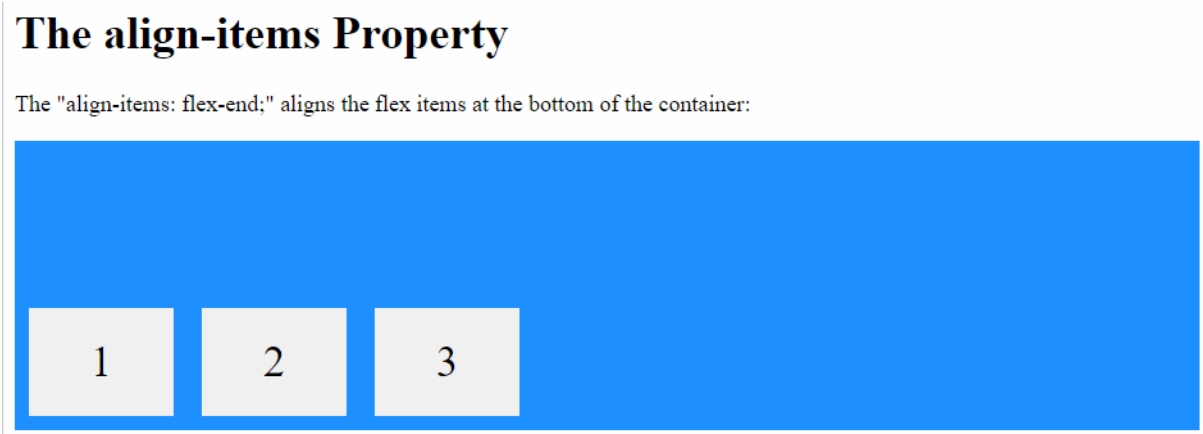
Example

The `flex-start` value aligns the flex items at the top of the container:

```
.container {
  display: flex;
  height: 200px;
  align-items: flex-start;
```

```
}
```

2.align-items:end;



Example

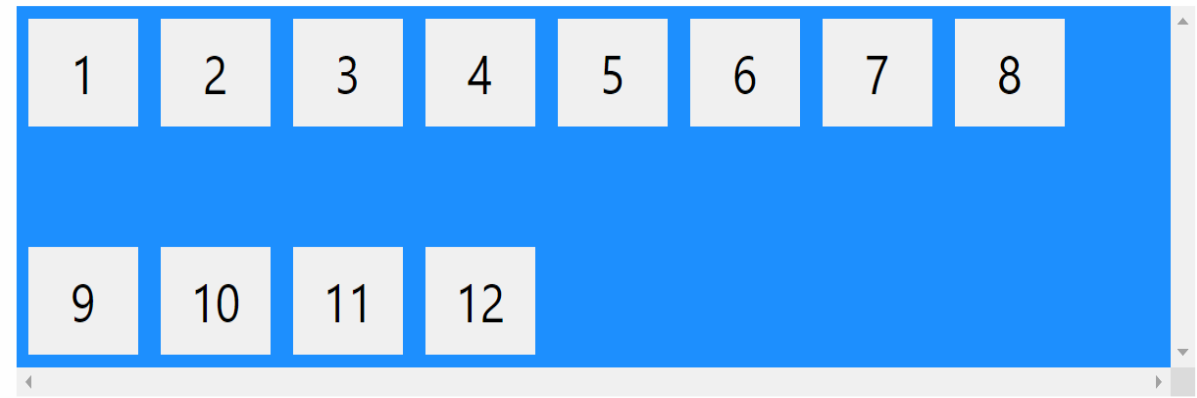
The `flex-end` value aligns the flex items at the bottom of the container:

```
.container {
  display: flex;
  height: 200px;
  align-items: flex-end;
}
```

The align-content Property

1.align-content:space-between;

The `align-content` property is used to align the flex lines.



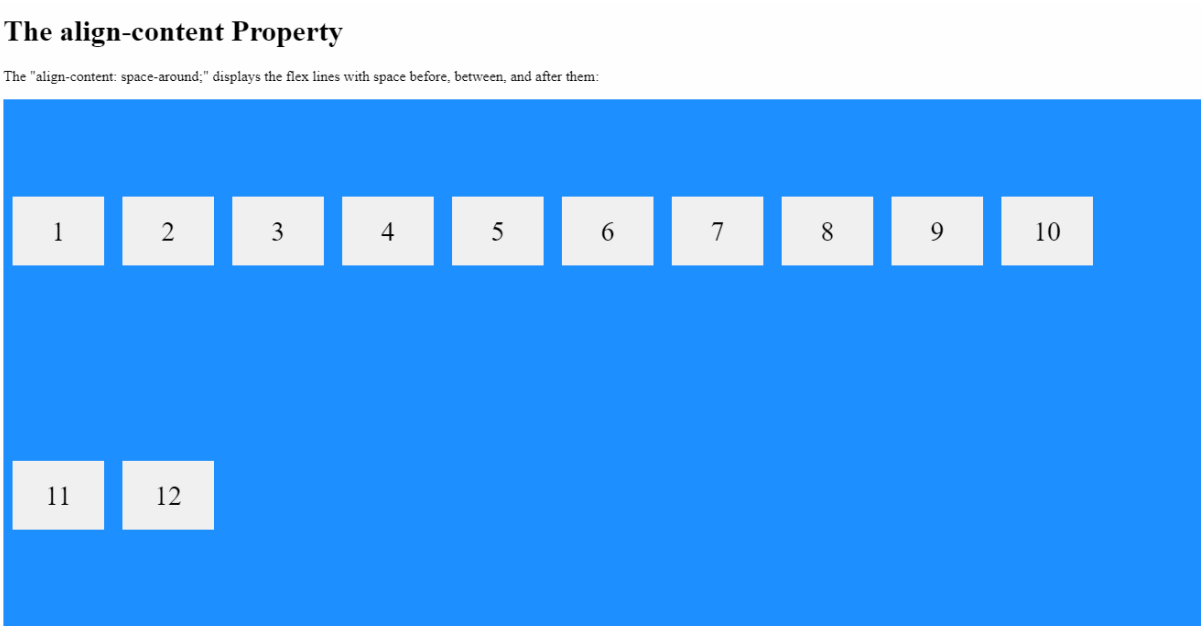
In these examples we use a 600 pixels high container, with the `flex-wrap` property set to `wrap`, to better demonstrate the `align-content` property.

Example

The `space-between` value displays the flex lines with s space between them:

```
.container {
  display: flex;
  height: 600px;
  flex-wrap: wrap;
  align-content: space-between;
}
```

2.align-content:space-around;



Example

The `space-around` value displays the flex lines with around space between them:

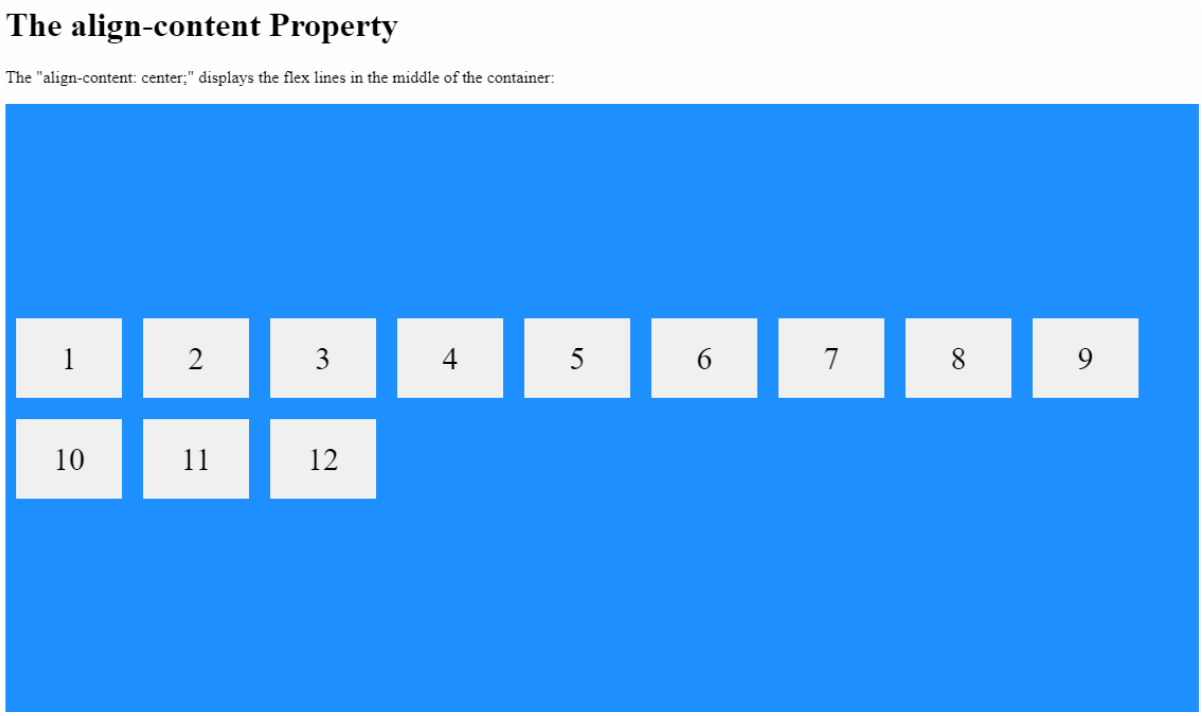
```
.container {
  display: flex;
  height: 600px;
  flex-wrap: wrap;
  align-content: space-around;
}
```

3.align-content:space-center;

Example

The `center` value displays the flex lines with center them:

```
.container {
  display: flex;
  height: 600px;
  flex-wrap: wrap;
  align-content: center;
}
```

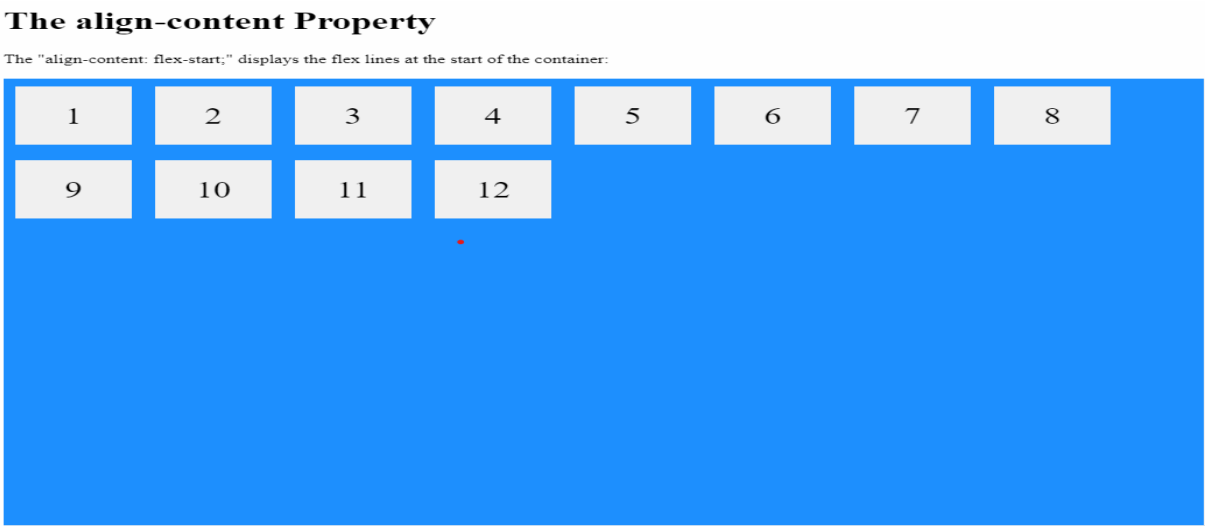


4.align-content:start;

Example

The `start` value displays the flex lines with start of the container:

```
.container {
  display: flex;
  height: 600px;
  flex-wrap: wrap;
  align-content: start;
}
```



4.align-content:end;

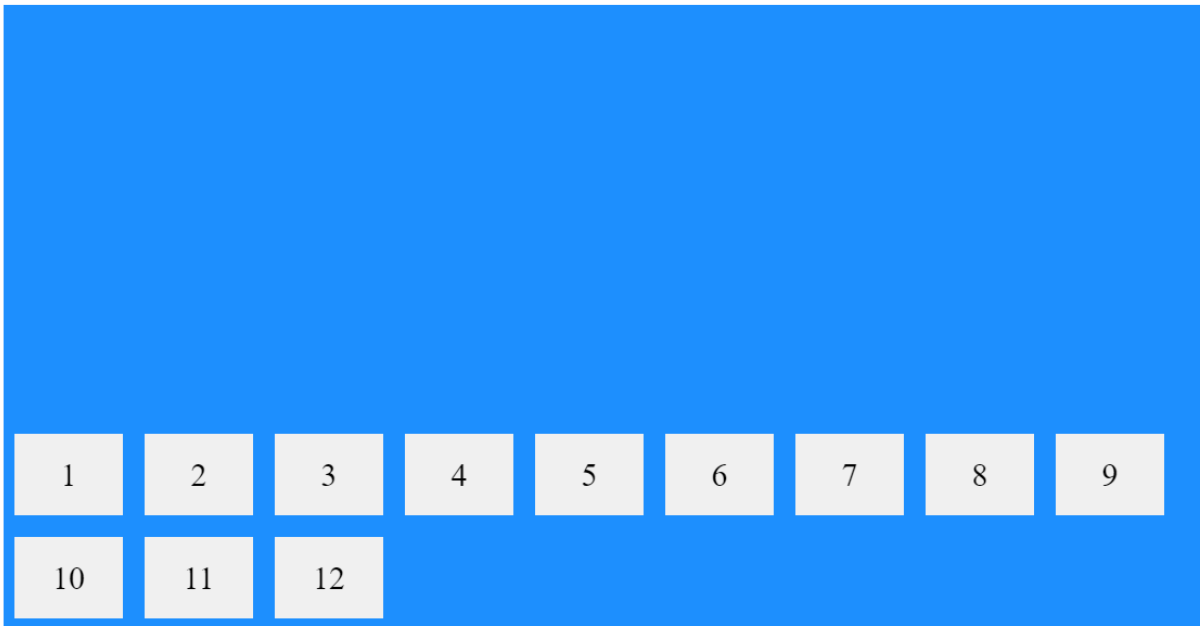
Example

The `end` value displays the flex lines at the end of the container:

```
.flex-container {
  display: flex;
  height: 600px;
  flex-wrap: wrap;
  align-content: end;
}
```

The align-content Property

The "align-content: flex-end;" displays the flex lines at the end of the container:



Perfect Centering

In the following example we will solve a very common style problem: perfect centering.

SOLUTION: Set both the `justify-content` and `align-items` properties to `center`, and the flex item will be perfectly centered:

Example

```
.flex-container {
  display: flex;
  height: 300px;
  justify-content: center;
  align-items: center;
}
```

The CSS Flexbox Container Properties

The following table lists all the CSS Flexbox Container properties:

Property	Description
align-content	Modifies the behavior of the flex-wrap property. It is similar to align-items, but instead of aligning flex items, it aligns flex lines
align-items	Vertically aligns the flex items when the items do not use all available space on the cross-axis
display	Specifies the type of box used for an HTML element
flex-direction	Specifies the direction of the flexible items inside a flex container
flex-flow	A shorthand property for flex-direction and flex-wrap
flex-wrap	Specifies whether the flex items should wrap or not, if there is not enough room for them on one flex line
justify-content	Horizontally aligns the flex items when the items do not use all available space on the main-axis

Child Elements (Items)

The direct child elements of a flex container automatically becomes flexible (flex) items.

The element above represents four blue flex items inside a grey flex container.

Example

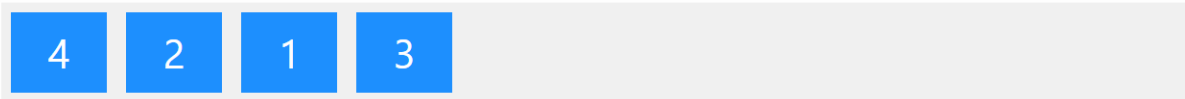
```
<div class="flex-container">
  <div>1</div>
  <div>2</div>
  <div>3</div>
  <div>4</div>
</div>
```

The flex item properties are:

- order
- flex-grow
- flex-shrink
- flex-basis
- flex
- align-self

The order Property

The `order` property specifies the order of the flex items.



The first flex item in the code does not have to appear as the first item in the layout.

The order value must be a number, default value is 0.

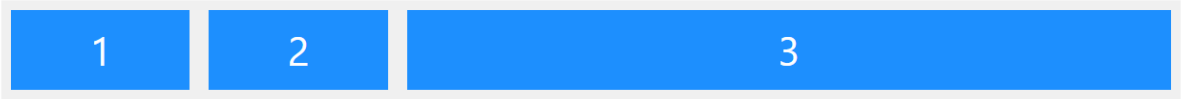
Example

The *order* property can change the order of the flex items:

```
<div class="flex-container">
  <div style="order: 3">1</div>
  <div style="order: 2">2</div>
  <div style="order: 4">3</div>
  <div style="order: 1">4</div>
</div>
```

The flex-grow Property

The `flex-grow` property specifies how much a flex item will grow relative to the rest of the flex items.



The value must be a number, default value is 0.

Example

Make the third flex item grow eight times faster than the other flex items:

```
<div class="flex-container">
  <div style="flex-grow: 1">1</div>
  <div style="flex-grow: 1">2</div>
  <div style="flex-grow: 8">3</div>
</div>
```

The flex-shrink Property

The `flex-shrink` property specifies how much a flex item will shrink relative to the rest of the flex items.

The value must be a number, default value is 1.

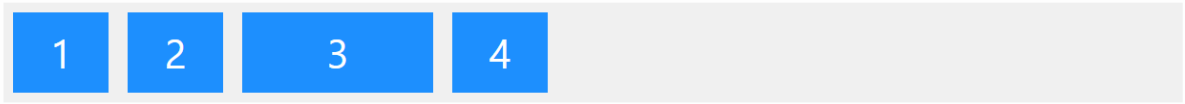
Example

Do not let the third flex item shrink as much as the other flex items:

```
<div class="flex-container">
  <div>1</div>
  <div>2</div>
  <div style="flex-shrink: 0">3</div>
  <div>4</div>
  <div>5</div>
  <div>6</div>
  <div>7</div>
  <div>8</div>
  <div>9</div>
  <div>10</div>
</div>
```

The flex-basis Property

The `flex-basis` property specifies the initial length of a flex item.



Example

Set the initial length of the third flex item to 200 pixels:

```
<div class="flex-container">
  <div>1</div>
  <div>2</div>
  <div style="flex-basis: 200px">3</div>
  <div>4</div>
</div>
```

The flex Property

The `flex` property is a shorthand property for the `flex-grow`, `flex-shrink`, and `flex-basis` properties.

Example

Make the third flex item not growable (0), not shrinkable (0), and with an initial length of 200 pixels:

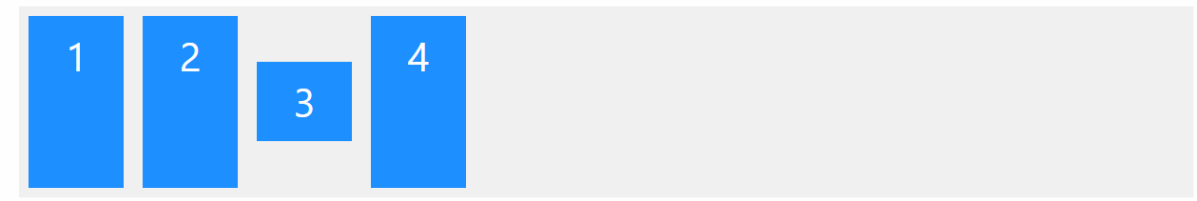
```
<div class="flex-container">
  <div>1</div>
  <div>2</div>
  <div style="flex: 0 0 200px">3</div>
  <div>4</div>
</div>
```

The align-self Property

1.align-self:center;

The `align-self` property specifies the alignment for the selected item inside the flexible container.

The `align-self` property overrides the default alignment set by the container's `align-items` property.



In these examples we use a 200 pixels high container, to better demonstrate the `align-self` property:

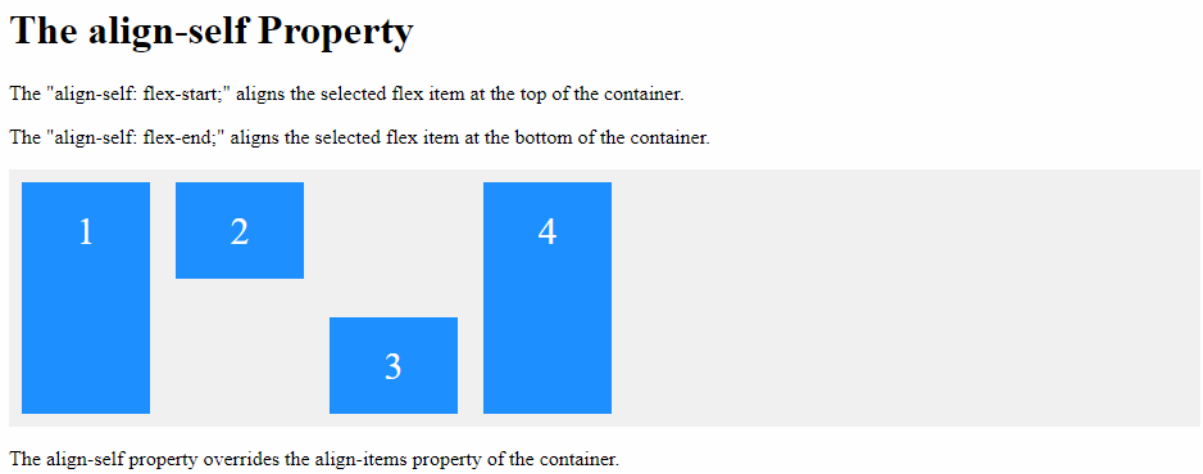
Example

Align the third flex item in the middle of the container:

```
<div class="flex-container">
  <div>1</div>
  <div>2</div>
  <div style="align-self: center">3</div>
  <div>4</div>
</div>
```

2.align-self:start;

3.align-self:end



Example

Align the second flex item at the top of the container, and the third flex item at the bottom of the container:

```
<div class="flex-container">
  <div>1</div>
  <div style="align-self: flex-start">2</div>
  <div style="align-self: flex-end">3</div>
  <div>4</div>
</div>
```

The CSS Flexbox Items Properties

The following table lists all the CSS Flexbox Items properties:

Property	Description
align-self	Specifies the alignment for a flex item (overrides the flex container's align-items property)
flex	A shorthand property for the flex-grow, flex-shrink, and the flex-basis properties
flex-basis	Specifies the initial length of a flex item
flex-grow	Specifies how much a flex item will grow relative to the rest of the flex items inside the same container
flex-shrink	Specifies how much a flex item will shrink relative to the rest of the flex items inside the same container
order	Specifies the order of the flex items inside the same container

CSS Media Queries

The `@media` rule, introduced in CSS2, made it possible to define different style rules for different media types.

Media queries in CSS3 extended the CSS2 media types idea: Instead of looking for a type of device, they look at the capability of the device.

Media queries can be used to check many things, such as:

- width and height of the viewport

- orientation of the viewport (landscape or portrait)
- resolution

Using media queries are a popular technique for delivering a tailored style sheet to desktops, laptops, tablets, and mobile phones (such as iPhone and Android phones).

Media Query Syntax

A media query consists of a media type and can contain one or more media features, which resolve to either true or false.

```
@media   (media feature) and (media feature) {
  CSS-Code;
}
```

CSS Common Media Features

Here are some commonly used media features:

Value	Description
orientation	Orientation of the viewport. Landscape or portrait
max-height	Maximum height of the viewport
min-height	Minimum height of the viewport
height	Height of the viewport (including scrollbar)
max-width	Maximum width of the viewport
min-width	Minimum width of the viewport
width	Width of the viewport (including scrollbar)

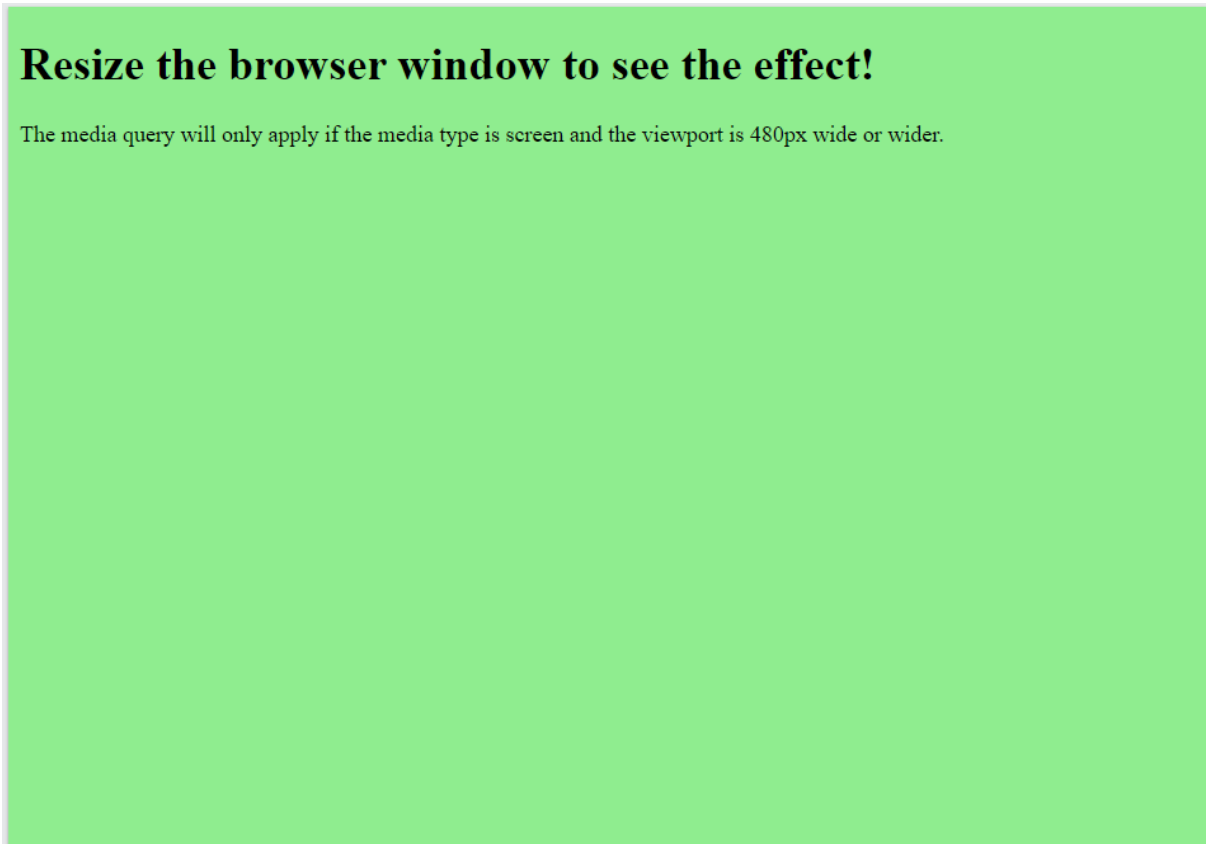
Media Queries Simple Examples

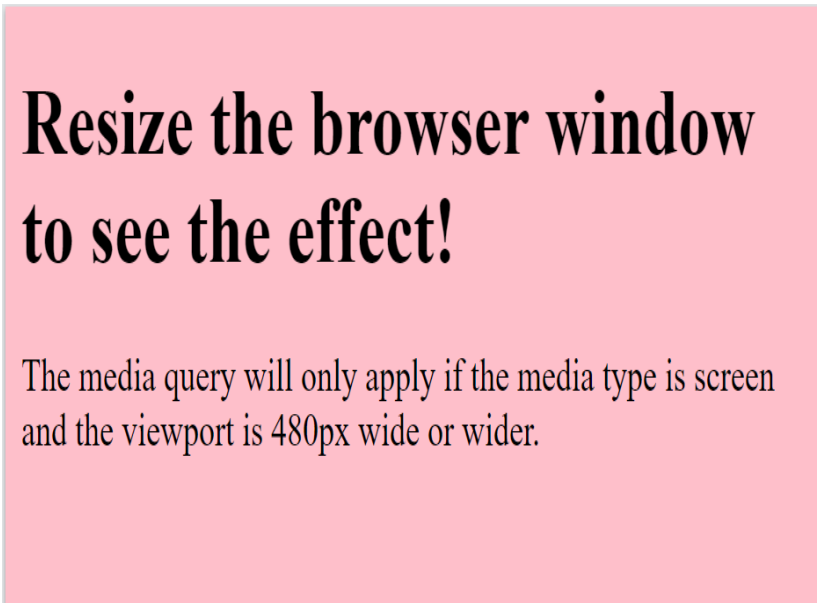
One way to use media queries is to have an alternate CSS section right inside your style sheet.

The following example changes the background-color to lightgreen if the viewport is 480 pixels wide or wider (if the viewport is less than 480 pixels, the background-color will be pink):

```
<!DOCTYPE html>
<html>
<head>
<style>
body {
  background-color: pink;
}

@media screen and (min-width: 480px) {
  body {
    background-color: lightgreen;
  }
}
</style>
</head>
<body>

<h1>Resize the browser window to see the effect!</h1>
```

```
<p>The media query will only apply if the media type is screen and the viewport is 480px wide or wider.</p>


</body>

</html>
```





What are Pseudo-classes?

A pseudo-class is used to define a special state of an element.

For example, it can be used to:

- Style an element when a user mouses over it
- Style visited and unvisited links differently
- Style an element when it gets focus

Anchor Pseudo-classes

1.link

2.visited

3.hover

4.active

Syntax

The syntax of pseudo-classes:

```
selector:pseudo-class {
  property: value;
```

```
}

<!DOCTYPE html>
<html>
<head>
<style>
/* unvisited link */
a:link {
  color: red;
}

/* visited link */
a:visited {
  color: green;
}

/* mouse over link */
a:hover {
  color: hotpink;
}

/* selected link */
a:active {
  color: blue;
}
</style>
</head>
<body>

<h2>Styling a link depending on state</h2>

<p><b><a href="default.asp" target="_blank">This is a
link</a></b></p>
<p><b>Note:</b> a:hover MUST come after a:link and a:visited in the
CSS definition in order to be effective.</p>
<p><b>Note:</b> a:active MUST come after a:hover in the CSS
definition in order to be effective.</p>

</body>
</html>
```