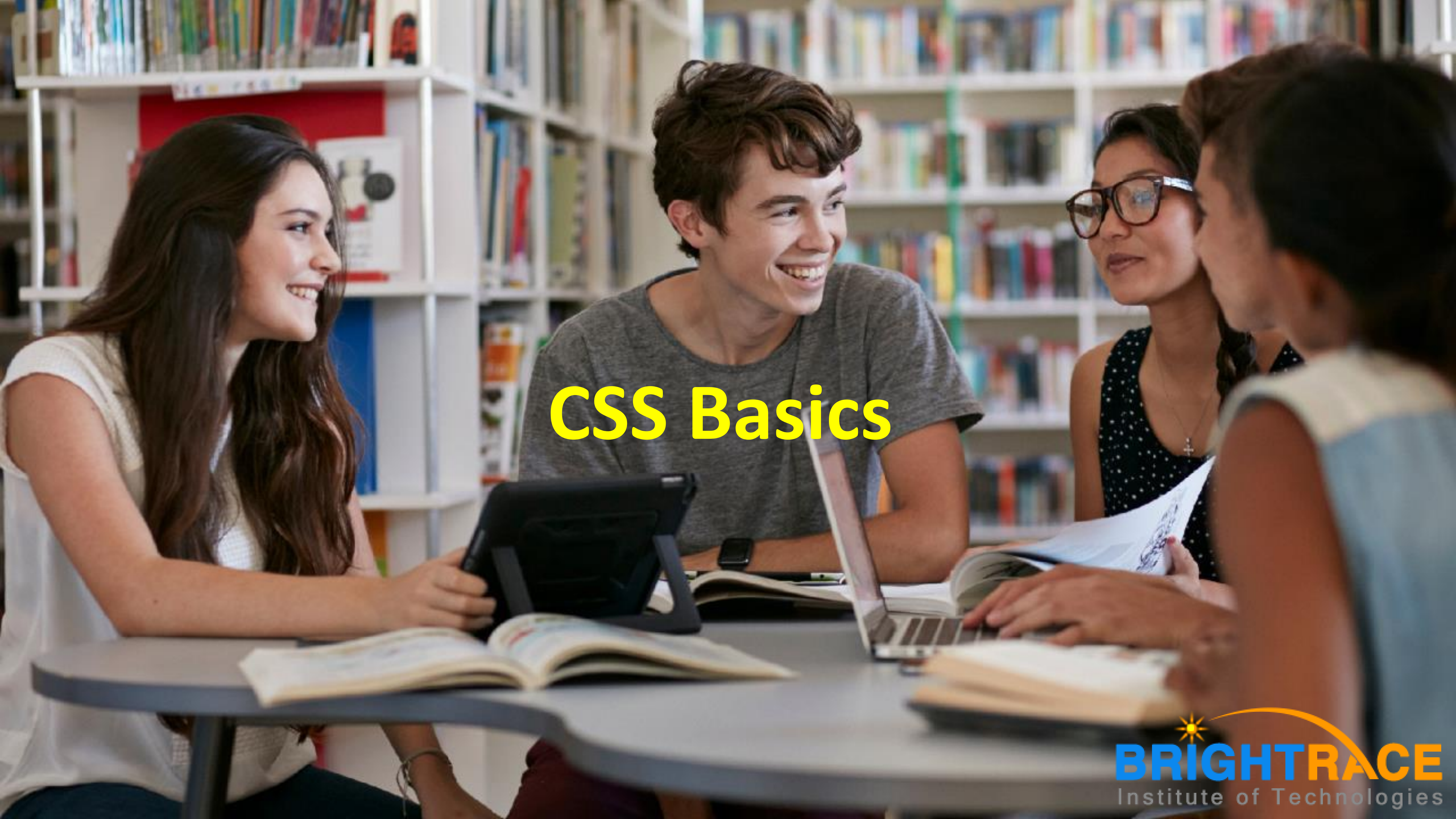


CSS Basics

BRIGHTRACE
Institute of Technologies

Agenda

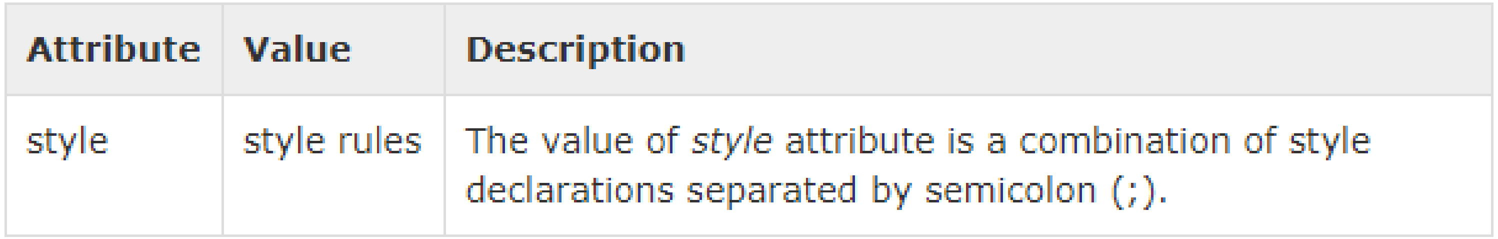
- Basics of CSS
  - Definition of CSS
  - Using inline CSS
  - Color
  - Using Internal CSS
  - Using Ids and Classes
  - Creating External CSS
  - Linking to External CSS
  - Inefficient Selectors
  - Efficient Selectors
  - HTML Element State
  - The CSS Box Model
  - Fonts
  - Tables
  - Margins
  - Padding
  - Scrollbars
- Main CSS3 Specific Properties
  - Opacity
  - Alpha Color Space
  - Shadow
  - Rounded Corners
  - Border Images
  - Gradient
  - 2D transform
  - 3D transform
  - Animation
  - Box Sizing
- CSS Flexbox
  - Flexbox layout Module
  - Flexbox Elements
  - Parent Elements (Container)
  - Child Elements (Items)

Definition of CSS

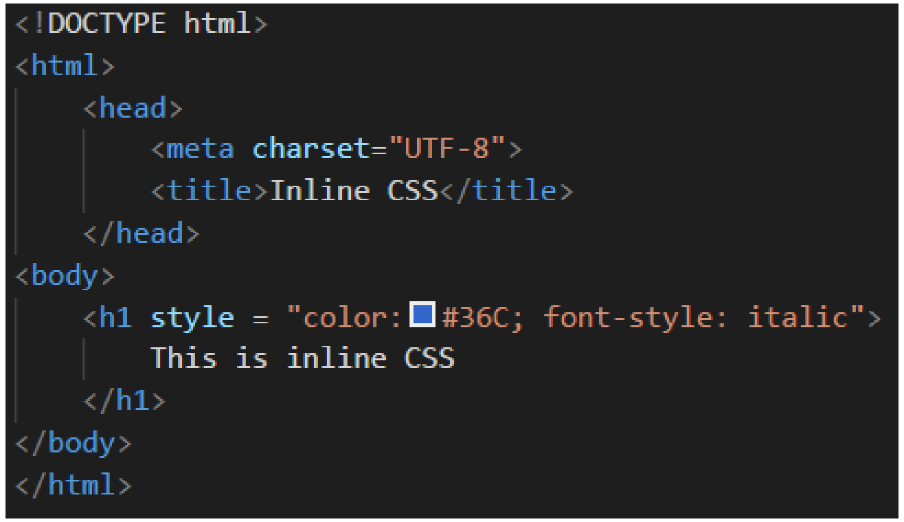
- CSS is the acronym for "Cascading Style Sheet“ and provides HTML with layout and design.
- It is used to control the style of a web document in a simple and easy way.
- Along with making things pretty and aesthetically pleasing, CSS also provides a general structure to HTML.
- CSS can be implemented in three different ways to our HTML:
  - Inline
  - Internal
  - External

Using inline CSS

- An inline style may be used to apply a unique style for a single element.
- To use inline styles, add the style attribute to the relevant element. The style attribute can contain any CSS property.
- The example below shows using inline css in a html.



Attribute	Value	Description
style	style rules	The value of *style* attribute is a combination of style declarations separated by semicolon (;).

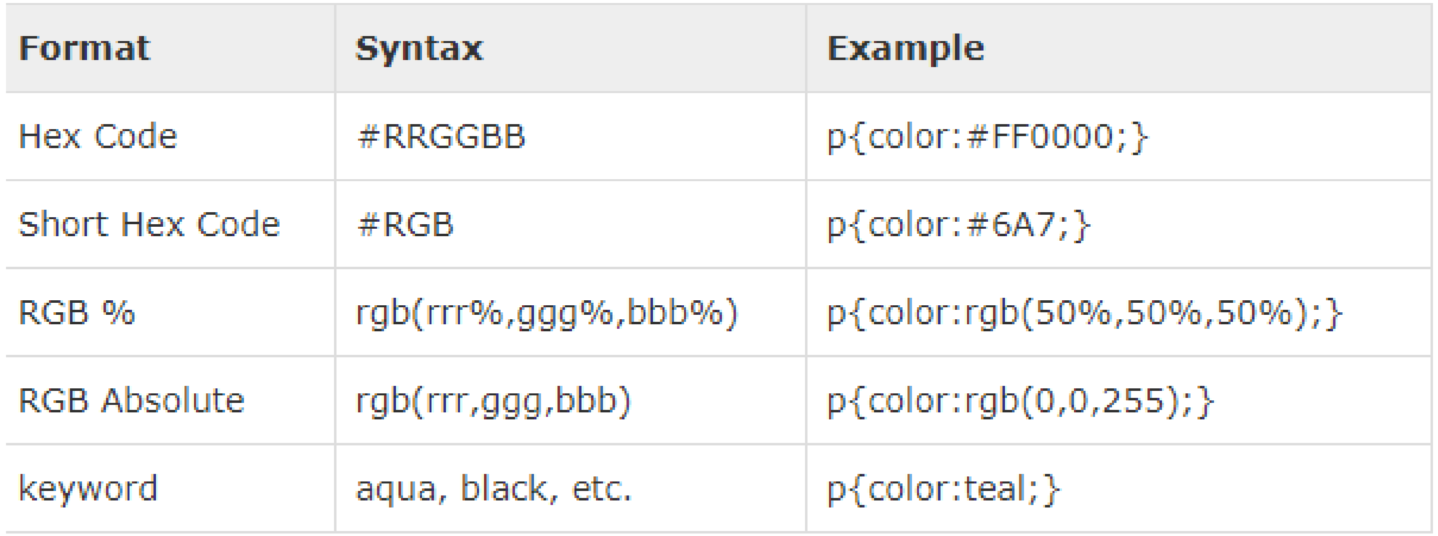


```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="UTF-8">
        <title>Inline CSS</title>
    </head>
<body>
    <h1 style = "color: #36C; font-style: italic">
        This is inline CSS
    </h1>
</body>
</html>
```

Using inline CSS cont.

CSS Color

- You can specify your color values in various formats. Following table lists all the possible formats



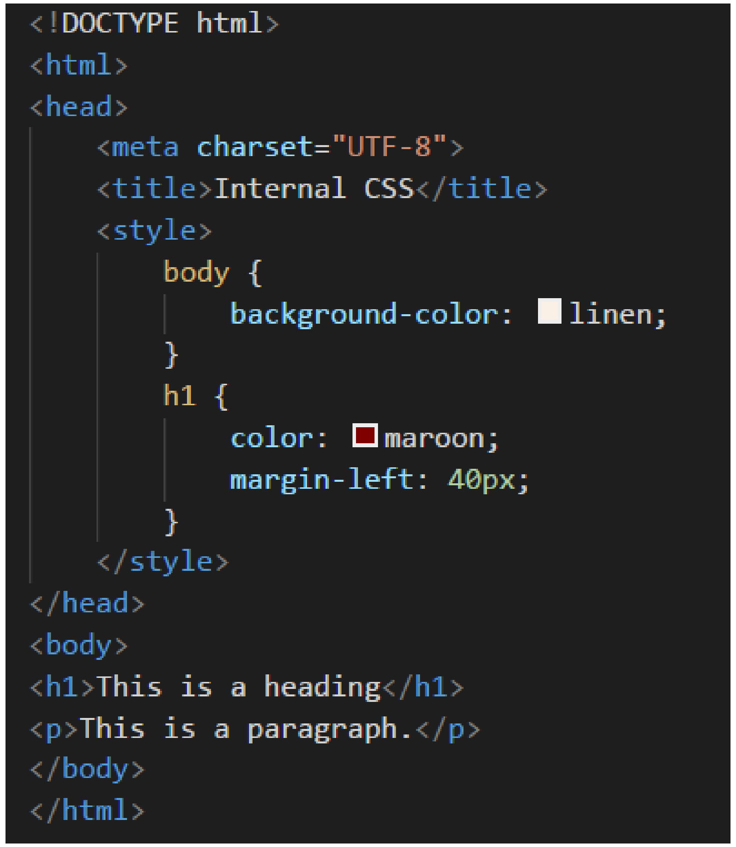
Format	Syntax	Example
Hex Code	#RRGGBB	p{color:#FF0000;}
Short Hex Code	#RGB	p{color:#6A7;}
RGB %	rgb(rrr%,ggg%,bbb%)	p{color:rgb(50%,50%,50%);}
RGB Absolute	rgb(rrr,ggg,bbb)	p{color:rgb(0,0,255);}
keyword	aqua, black, etc.	p{color:teal;}

Using Internal CSS

- As you have already noticed, ***Inline CSS*** is good for adding slight changes or specifying colors for different text elements, but it starts to get a little 'wordy' and messy.
- It can be used if one single page has a unique style.
- They are defined within the <style> element, inside the <head> section of an HTML page.
- When we add CSS to HTML either; externally or in the head section, we can use selectors which allow us to 'select' or 'point' to a specific element of our HTML.
- CSS can use HTML elements as selectors, such as the paragraph, body, h1, anchor, em and strong tags.
- If we referred to these elements as selectors in our CSS we would be styling every paragraph, body, h1, anchor, em and strong element in our HTML.

Using Internal CSS cont.

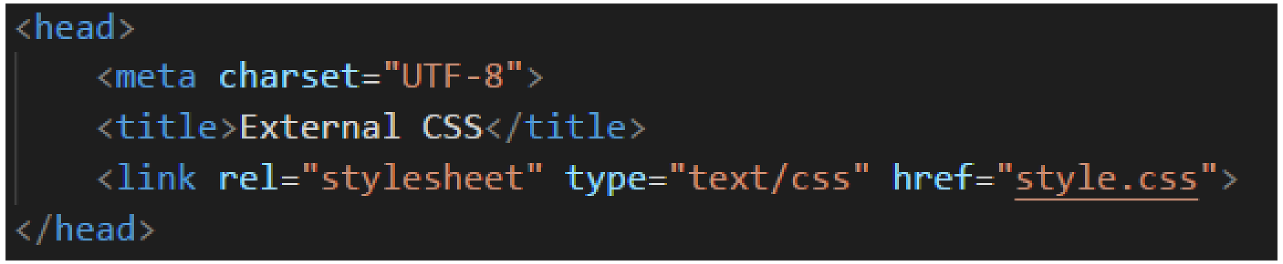
- Rules defined using this syntax will be applied to all the elements available in the document as shown in the example below.



```
<!DOCTYPE html>
<html>
<head>
    <meta charset="UTF-8">
    <title>Internal CSS</title>
    <style>
        body {
            background-color: ⬜linen;
        }
        h1 {
            color: 🟥maroon;
            margin-left: 40px;
        }
    </style>
</head>
<body>
<h1>This is a heading</h1>
<p>This is a paragraph.</p>
</body>
</html>
```

Using External CSS

- With an external style sheet, you can change the look of an entire website by changing just one file.
- To add an external CSS to the HTML, we need to tell the HTML all about it- what relation it has to the HTML, the type of file, its location and name.
- Each page must include a reference to the external style sheet file inside the <link> element. The <link> element goes inside the <head> section.



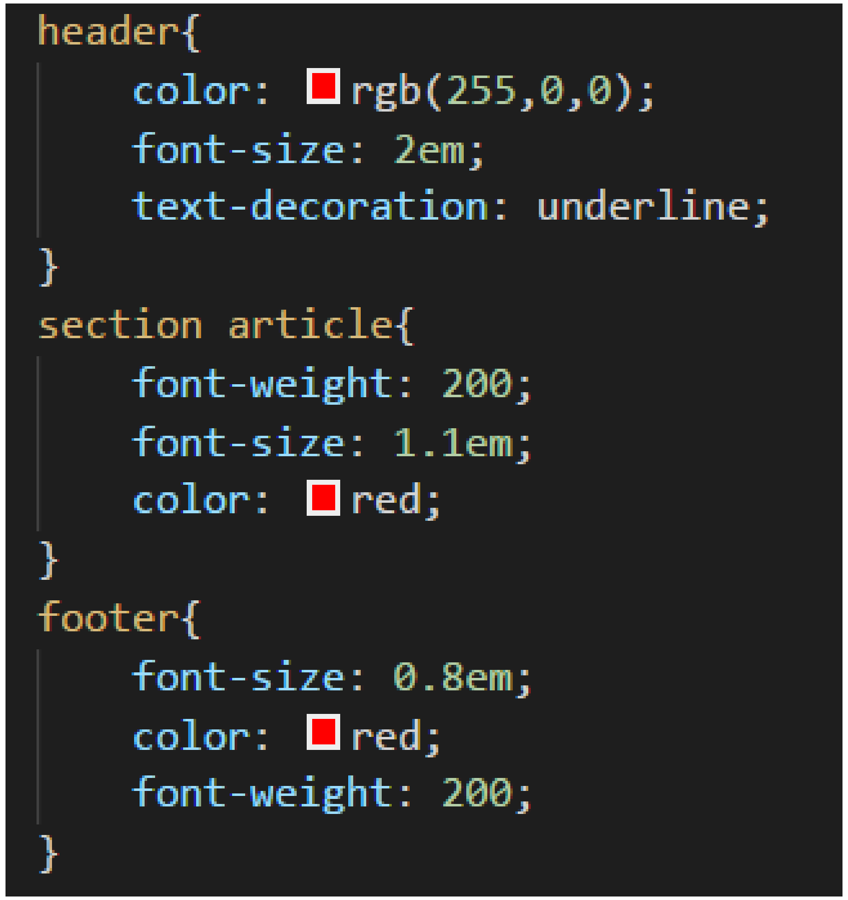
```
<head>
    <meta charset="UTF-8">
    <title>External CSS</title>
    <link rel="stylesheet" type="text/css" href="style.css">
</head>
```

- In the above example, we use a “rel” value to tell HTML the relation of the CSS file to the HTML, a “type” value to tell the HTML the type of file and a “href” value telling the HTML where the file is located and its name.

Using External CSS cont.

Creating CSS file:

- An ***external style sheet*** can be written in any text editor.
- The file should not contain any html tags.
- The style sheet file must be saved with a .css extension.
- The following is an example of css file.

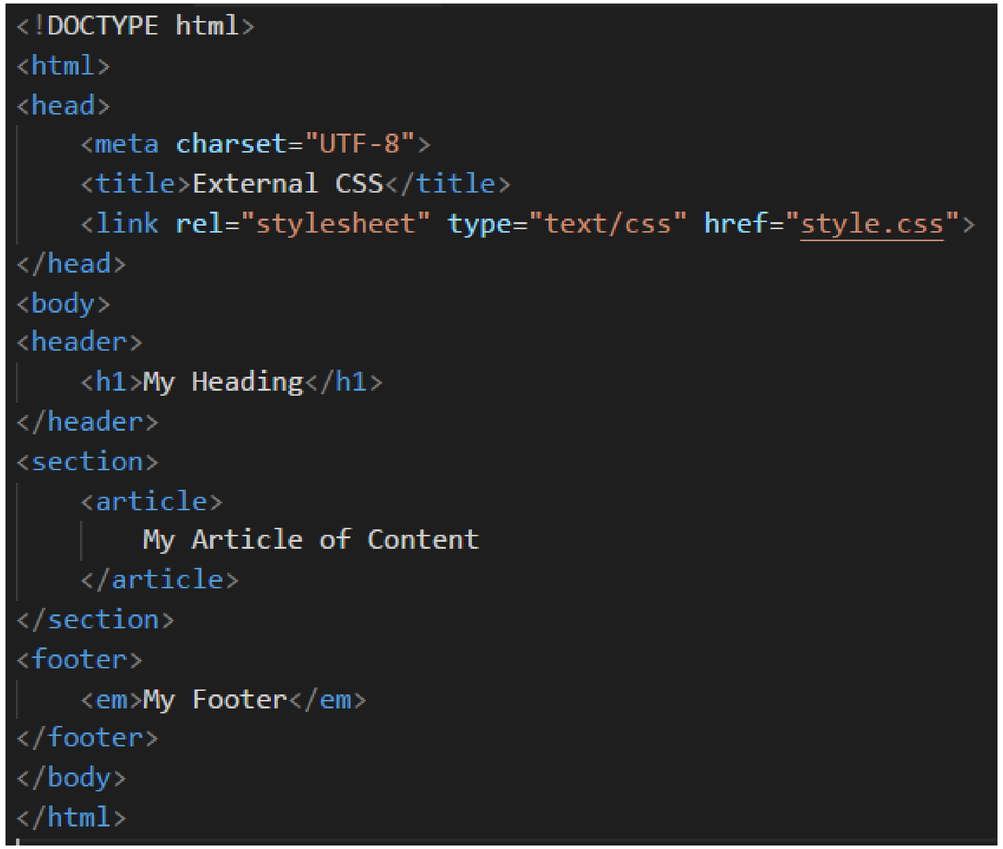


```
header{
    color: 🟥rgb(255,0,0);
    font-size: 2em;
    text-decoration: underline;
}
section article{
    font-weight: 200;
    font-size: 1.1em;
    color: 🟥red;
}
footer{
    font-size: 0.8em;
    color: 🟥red;
    font-weight: 200;
}
```

Using External CSS cont.

Linking to External CSS:

- If our style sheet (CSS) were located in the same directory (or folder) as our HTML file, we would add the tag to the head section of the HTML document, like shown below.
- The CSS will tell the browser to render the styles for our HTML in the same way

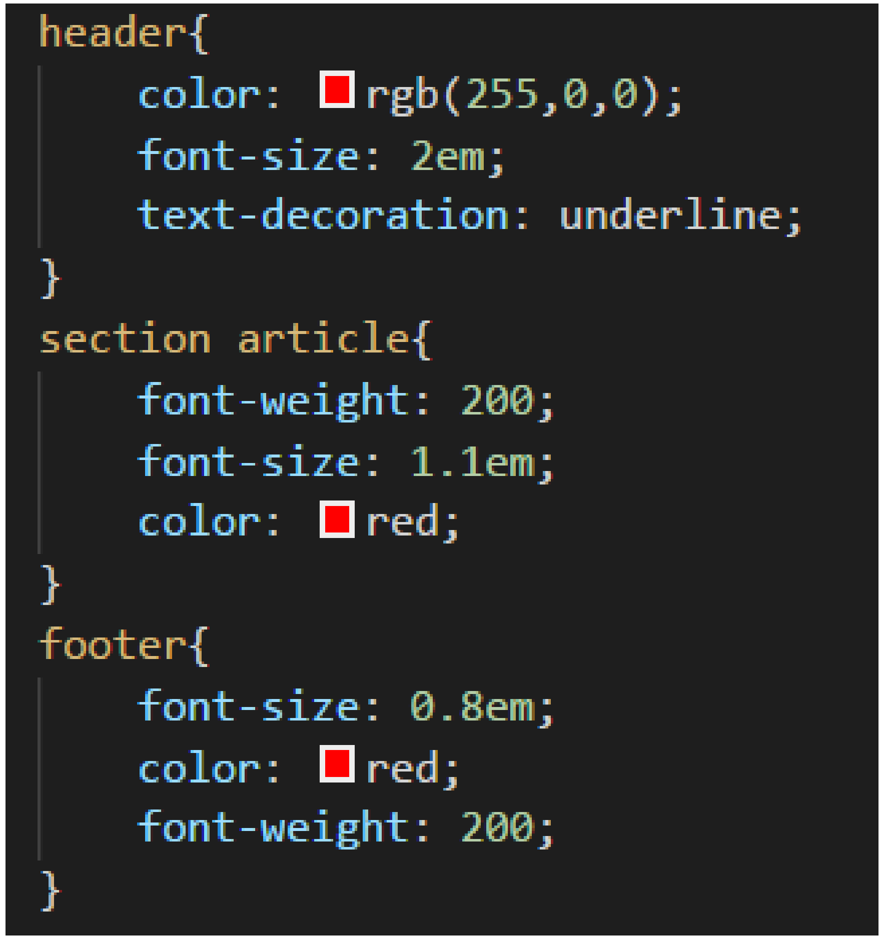


```
<!DOCTYPE html>
<html>
<head>
    <meta charset="UTF-8">
    <title>External CSS</title>
    <link rel="stylesheet" type="text/css" href="style.css">
</head>
<body>
<header>
    <h1>My Heading</h1>
</header>
<section>
    <article>
        My Article of Content
    </article>
</section>
<footer>
    <em>My Footer</em>
</footer>
</body>
</html>
```

Using External CSS cont.

Inefficient Selectors

- The advantages of using external CSS include:
  - the ability to completely separate the HTML from the CSS
  - to reduce individual file size and length
  - make things more readable and use effective selectors (meaning selectors that target multiple elements where necessary).
- If you observe this example, CSS is repeating and applying the same styling to different elements which is called ***Inefficient Selectors***.

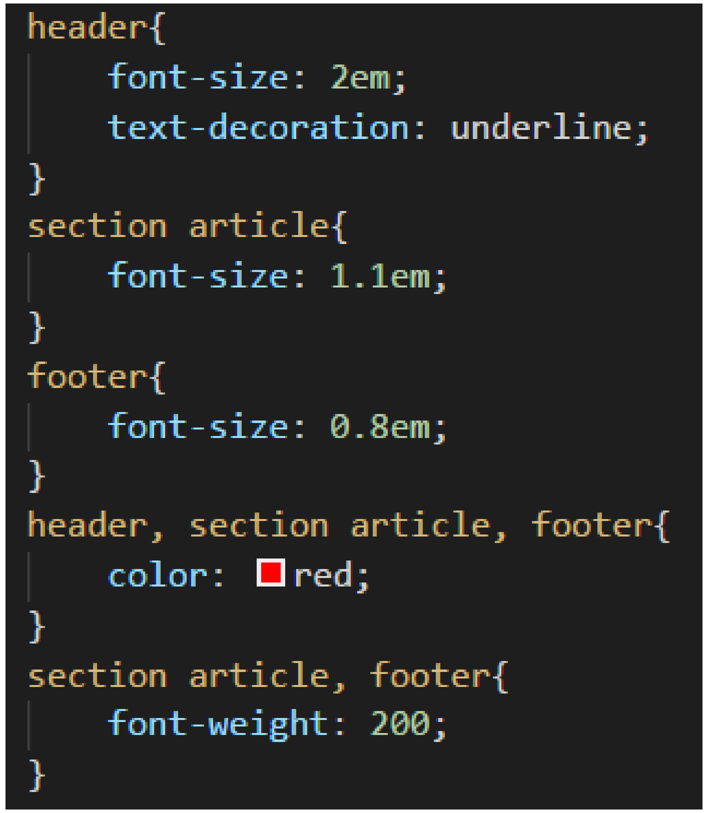


```css
header{
    color: 🟥rgb(255,0,0);
    font-size: 2em;
    text-decoration: underline;
}
section article{
    font-weight: 200;
    font-size: 1.1em;
    color: 🟥red;
}
footer{
    font-size: 0.8em;
    color: 🟥red;
    font-weight: 200;
}
```

Using External CSS cont.

Efficient Selectors

- Rather than having a large portion of CSS repeating and applying the same styling to different elements, we could 'join' the selectors together to create a smaller file size or to simply be more efficient with our use of CSS.
- We can target or select multiple elements by separating the selectors with a comma in the CSS.
- Lets look in to an example of ***efficient selectors***:



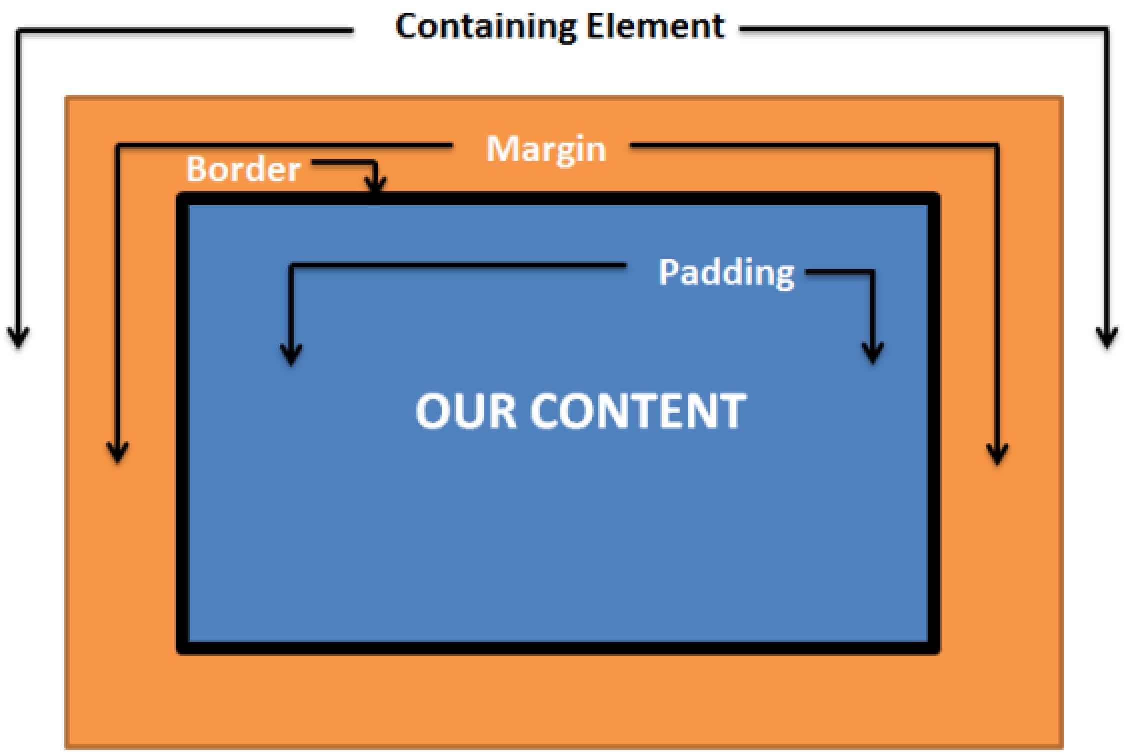
```css
header{
    font-size: 2em;
    text-decoration: underline;
}
section article{
    font-size: 1.1em;
}
footer{
    font-size: 0.8em;
}
header, section article, footer{
    color: 🟥red;
}
section article, footer{
    font-weight: 200;
}
```

The CSS Box Model

- All HTML elements can be considered as boxes.
- One of the fundamental understandings of CSS is the ***Box Model***.
- It allows us to add a border around elements, and to define space between elements.
- It is essentially a box that wraps around every HTML element.
- It consists of: margins, borders, padding, and the actual content.

The CSS Box Model cont.

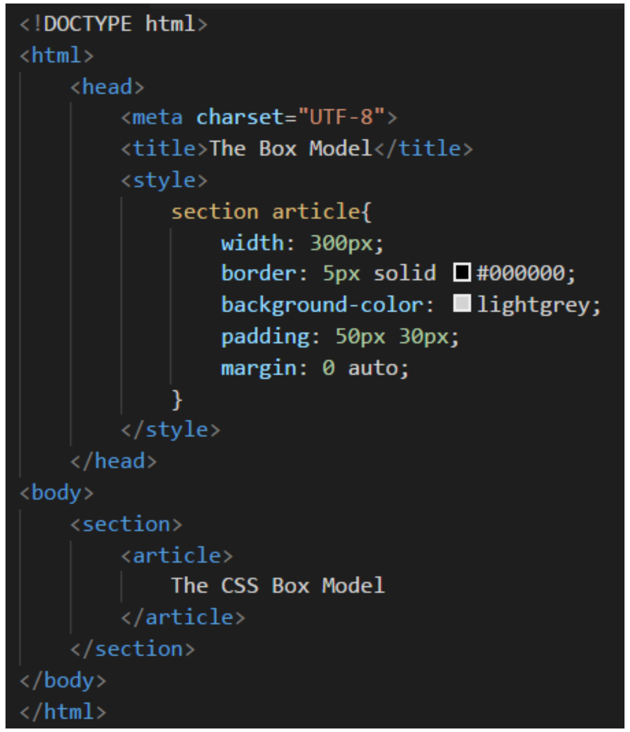
- The image illustrates the box model:
- The ***padding*** is the area that separates the content from the border. It is transparent.
- The ***border*** is the area that separates the padding from the margin.
- The ***margin*** is the area that separates our box from surrounding elements. It is transparent
- The ***content*** - The content of the box, where text and images appear.



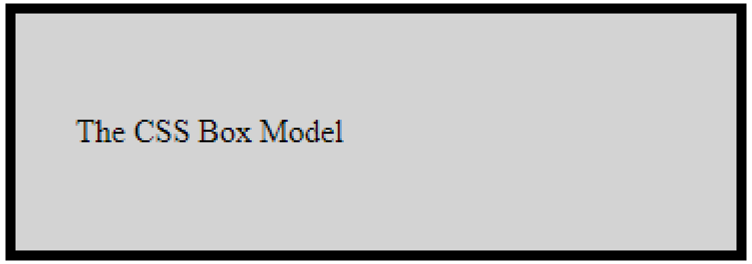
The CSS Box Model cont.

Let's look in to an example:

- We have set the padding for the top and bottom of the element to 50px and the left and right to 30px.
- The margin has been set to 0px for the top and bottom and the left and right margin is set to auto.
- When we set a value of auto in our margins, it will basically 'center' the element within the containing or parent element.
- The border is set to 5px wide for each side, which is solid and has a color of black.
- The rendered output is also shown.



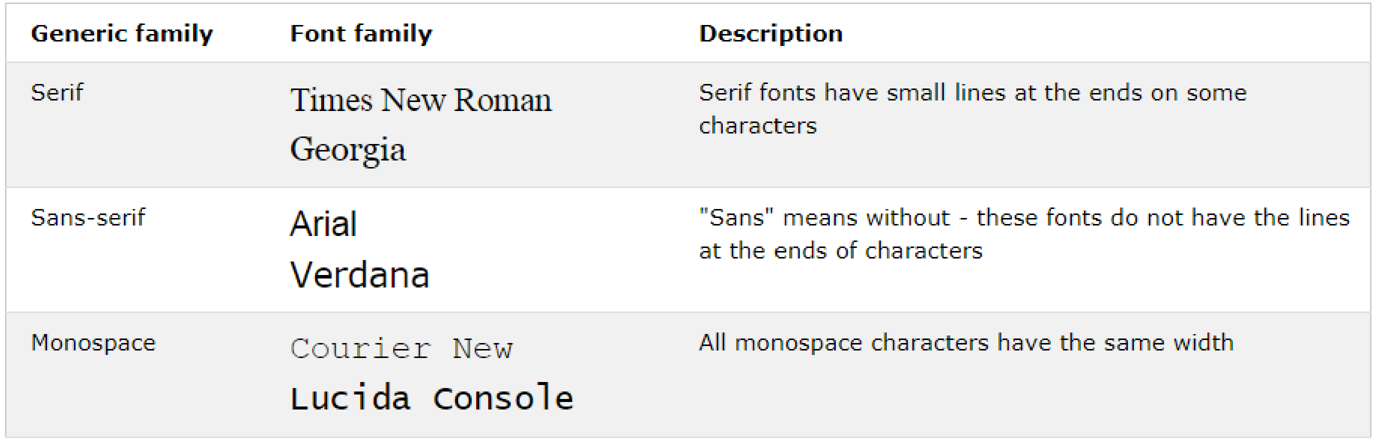
```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="UTF-8">
        <title>The Box Model</title>
        <style>
            section article{
                width: 300px;
                border: 5px solid □#000000;
                background-color: ■lightgrey;
                padding: 50px 30px;
                margin: 0 auto;
            }
        </style>
    </head>
<body>
    <section>
        <article>
            The CSS Box Model
        </article>
    </section>
</body>
</html>
```



The CSS Box Model

CSS Fonts

- Using ***CSS font*** properties, we can change the font-family of our text.
- In CSS, there are two types of font family names:
  - ***generic family*** - a group of font families with a similar look (like "Serif" or "Monospace")
  - ***font family*** - a specific font family (like "Times New Roman" or "Arial")



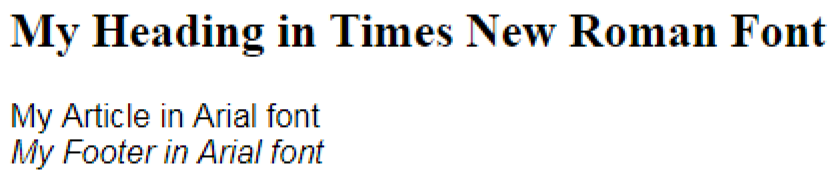
Generic family	Font family	Description
Serif	Times New Roman Georgia	Serif fonts have small lines at the ends on some characters
Sans-serif	Arial Verdana	"Sans" means without - these fonts do not have the lines at the ends of characters
Monospace	Courier New Lucida Console	All monospace characters have the same width

CSS Fonts cont.

- We can specify multiple font-families for any given element.
- The font-family property should hold several font names as a "fallback" system. If the browser does not support the first font, it tries the next font, and so on.
- Start with the font you want, and end with a generic family, to let the browser pick a similar font in the generic family, if no other fonts are available.
- These font-families are separated with a comma and the proceeding fonts are referred to as fallback fonts.

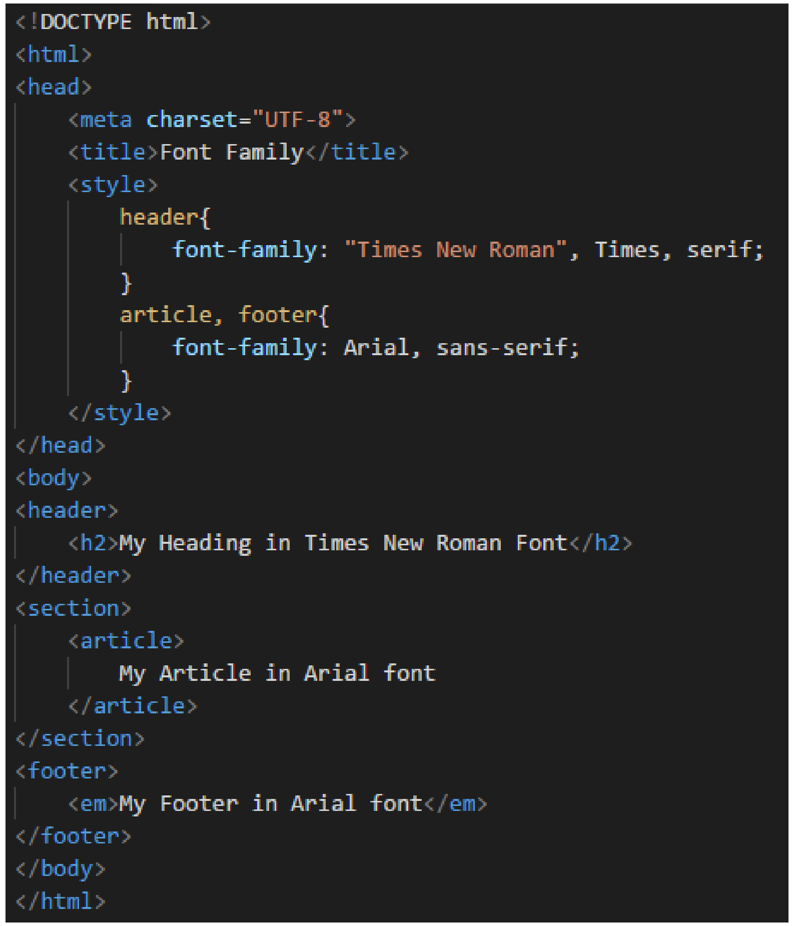
CSS Fonts cont.

- In the example we are saying that our header should have a font-family of Times New Roman.
- If Times New Roman is not located on the users system, we will try Times.
- If Times is not located on the user's system, we will 'fall-back' to a generic serif font.
- Rendered output is shown here.



My Heading in Times New Roman Font

My Article in Arial font
My Footer in Arial font



```
<!DOCTYPE html>
<html>
<head>
    <meta charset="UTF-8">
    <title>Font Family</title>
    <style>
        header{
            font-family: "Times New Roman", Times, serif;
        }
        article, footer{
            font-family: Arial, sans-serif;
        }
    </style>
</head>
<body>
<header>
    <h2>My Heading in Times New Roman Font</h2>
</header>
<section>
    <article>
        My Article in Arial font
    </article>
</section>
<footer>
    <em>My Footer in Arial font</em>
</footer>
</body>
</html>
```

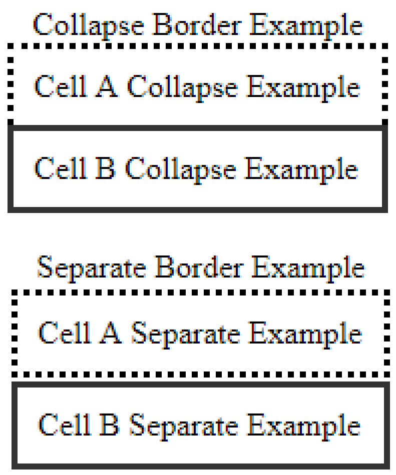
CSS Tables

- Let’s see how to set different properties of an HTML table using CSS.
- You can set following properties of a table:
    - The **border-collapse** specifies whether the browser should control the appearance of the adjacent borders that touch each other or whether each cell should maintain its style.
    - The **border-spacing** specifies the width that should appear between table cells.
    - The **caption-side** captions are presented in the <caption> element. By default, these are rendered above the table in the document. You use the *caption-side* property to control the placement of the table caption.
    - The **empty-cells** specifies whether the border should be shown if a cell is empty.
    - The **table-layout** allows browsers to speed up layout of a table by using the first width properties it comes across for the rest of a column rather than having to load the whole table before rendering it.

CSS Tables cont.

The border-collapse Property

- This property can have two values:
  - ***Collapse***
  - ***separate***.
- The example shows the use both the values.

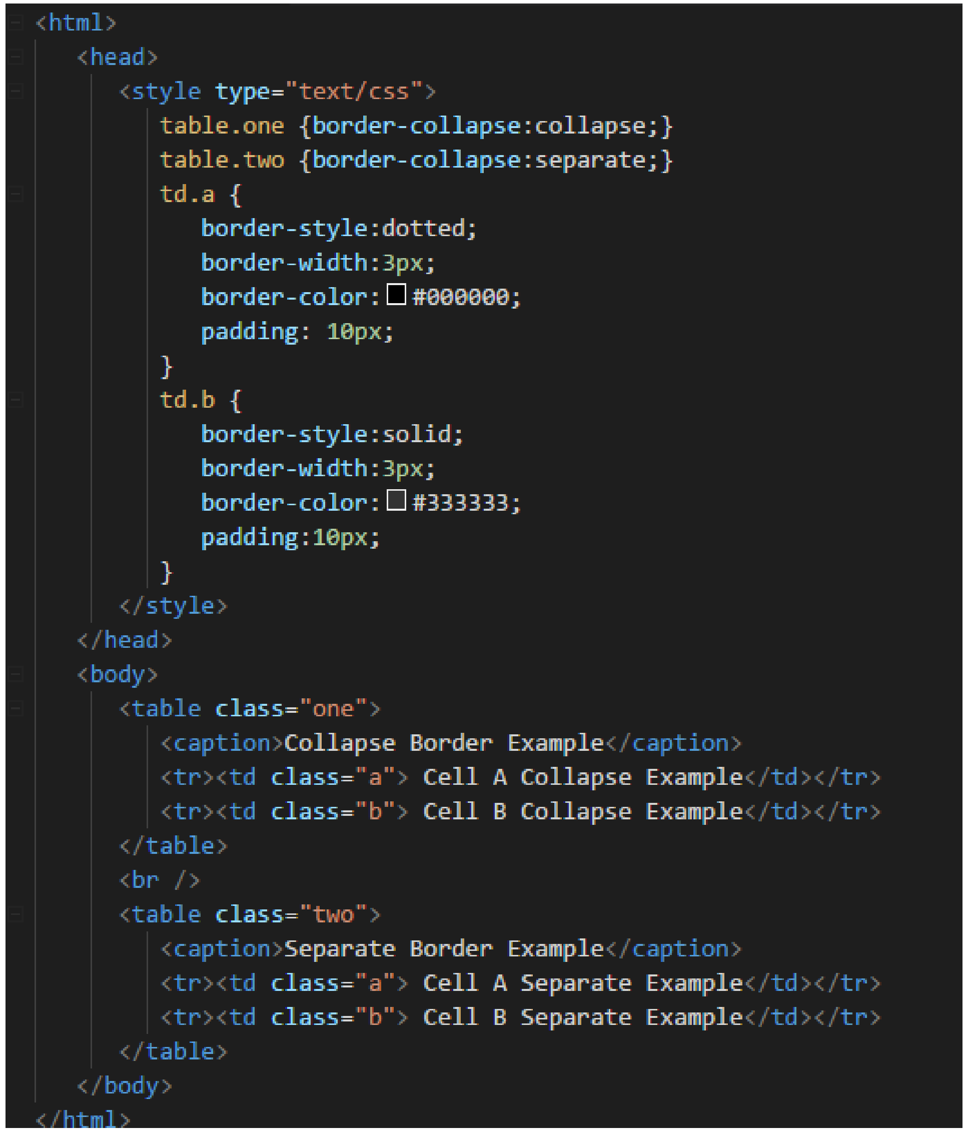


Collapse Border Example

Cell A Collapse Example
Cell B Collapse Example

Separate Border Example

Cell A Separate Example
Cell B Separate Example

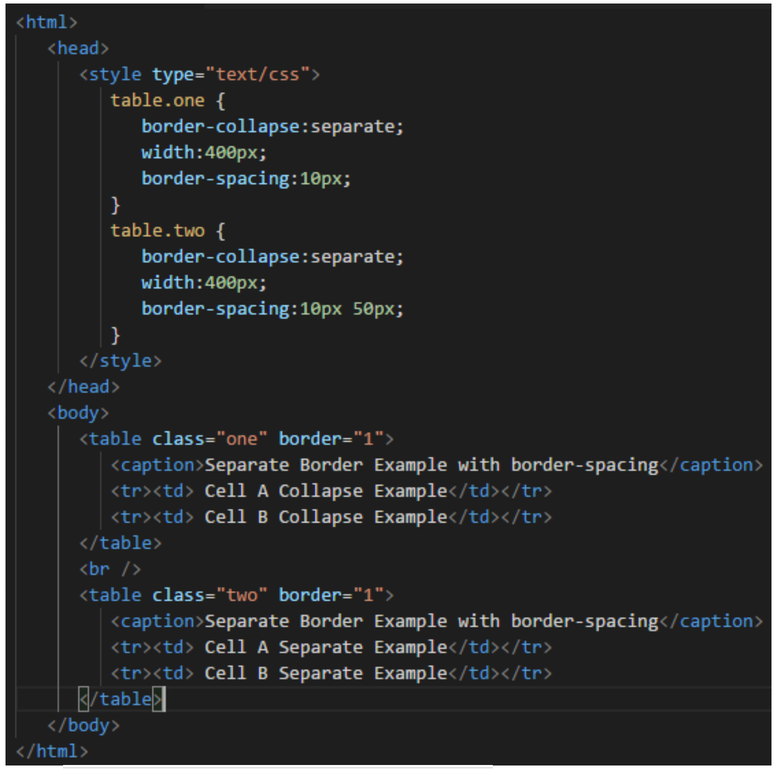


```html
<html>
   <head>
      <style type="text/css">
         table.one {border-collapse:collapse;}
         table.two {border-collapse:separate;}
         td.a {
            border-style:dotted;
            border-width:3px;
            border-color:□#000000;
            padding: 10px;
         }
         td.b {
            border-style:solid;
            border-width:3px;
            border-color:□#333333;
            padding:10px;
         }
      </style>
   </head>
   <body>
      <table class="one">
         <caption>Collapse Border Example</caption>
         <tr><td class="a"> Cell A Collapse Example</td></tr>
         <tr><td class="b"> Cell B Collapse Example</td></tr>
      </table>
      <br />
      <table class="two">
         <caption>Separate Border Example</caption>
         <tr><td class="a"> Cell A Separate Example</td></tr>
         <tr><td class="b"> Cell B Separate Example</td></tr>
      </table>
   </body>
</html>
```

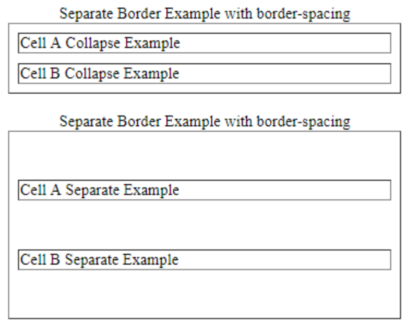
CSS Tables cont.

The border-spacing Property

- The border-spacing property specifies the distance that separates adjacent cells'.
- It can take either one or two values which should be units of length.
- If we provide one value, it will applies to both vertical and horizontal borders.
- If we specify two values, in which case, the first refers to the horizontal spacing and the second to the vertical spacing.
- Let see an example.



```
<html>
   <head>
      <style type="text/css">
         table.one {
            border-collapse:separate;
            width:400px;
            border-spacing:10px;
         }
         table.two {
            border-collapse:separate;
            width:400px;
            border-spacing:10px 50px;
         }
      </style>
   </head>
   <body>
      <table class="one" border="1">
         <caption>Separate Border Example with border-spacing</caption>
         <tr><td> Cell A Collapse Example</td></tr>
         <tr><td> Cell B Collapse Example</td></tr>
      </table>
      <br />
      <table class="two" border="1">
         <caption>Separate Border Example with border-spacing</caption>
         <tr><td> Cell A Separate Example</td></tr>
         <tr><td> Cell B Separate Example</td></tr>
      </table>
   </body>
</html>
```



Separate Border Example with border-spacing

Cell A Collapse Example
Cell B Collapse Example

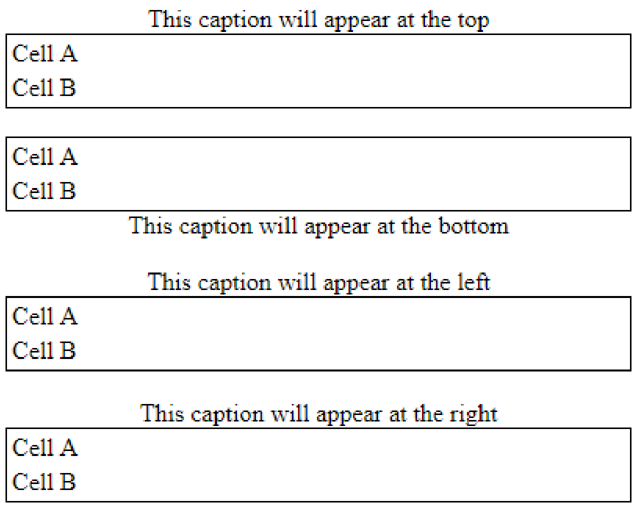
Separate Border Example with border-spacing

Cell A Separate Example
Cell B Separate Example

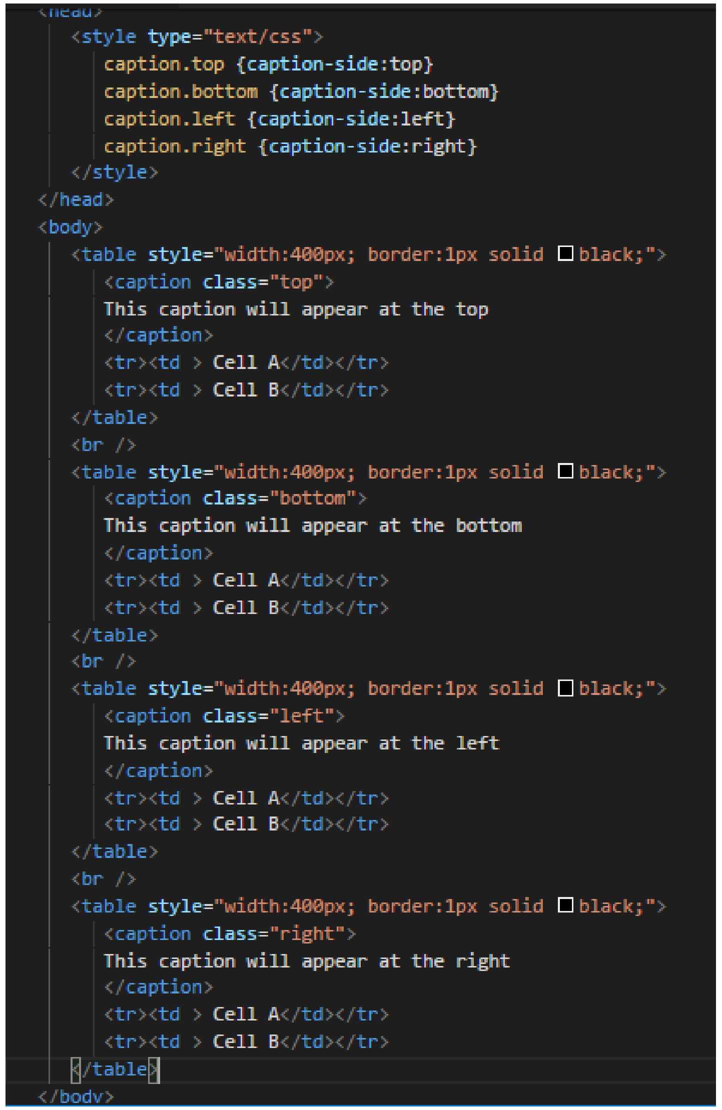
CSS Tables cont.

The caption-side Property

- The ***caption-side*** property allows you to specify where the content of a <caption> element should be placed in relationship to the table. The table that follows lists the possible values.
- The example uses each value.



```
This caption will appear at the top
Cell A
Cell B

Cell A
Cell B
This caption will appear at the bottom

This caption will appear at the left
Cell A
Cell B

This caption will appear at the right
Cell A
Cell B
```

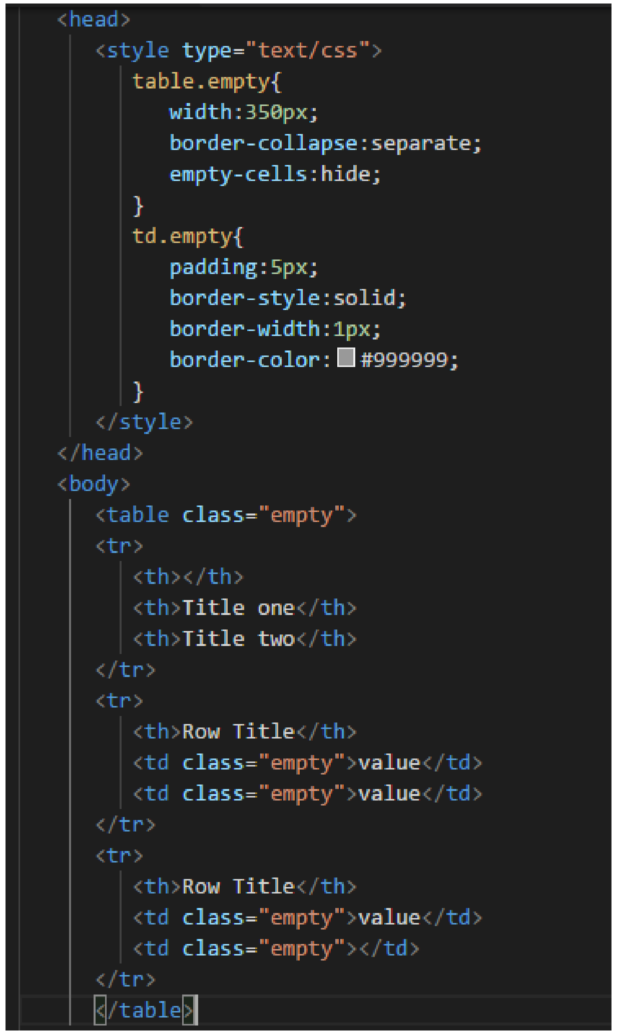


```
<head>
    <style type="text/css">
        caption.top {caption-side:top}
        caption.bottom {caption-side:bottom}
        caption.left {caption-side:left}
        caption.right {caption-side:right}
    </style>
</head>
<body>
    <table style="width:400px; border:1px solid □black;">
        <caption class="top">
        This caption will appear at the top
        </caption>
        <tr><td > Cell A</td></tr>
        <tr><td > Cell B</td></tr>
    </table>
    <br />
    <table style="width:400px; border:1px solid □black;">
        <caption class="bottom">
        This caption will appear at the bottom
        </caption>
        <tr><td > Cell A</td></tr>
        <tr><td > Cell B</td></tr>
    </table>
    <br />
    <table style="width:400px; border:1px solid □black;">
        <caption class="left">
        This caption will appear at the left
        </caption>
        <tr><td > Cell A</td></tr>
        <tr><td > Cell B</td></tr>
    </table>
    <br />
    <table style="width:400px; border:1px solid □black;">
        <caption class="right">
        This caption will appear at the right
        </caption>
        <tr><td > Cell A</td></tr>
        <tr><td > Cell B</td></tr>
    </table>
</body>
```

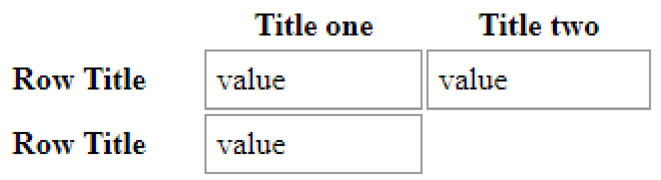
CSS Tables cont.

The empty-cells Property

- The ***empty-cells*** property indicates whether a cell without any content should have a border displayed.
- This property can have one of the three values - ***show, hide*** or ***inherit***.
- Here is an example of the ***empty-cells*** property used to ***hide borders of empty cells*** in the <table> element.



```
<head>
  <style type="text/css">
    table.empty{
      width:350px;
      border-collapse:separate;
      empty-cells:hide;
    }
    td.empty{
      padding:5px;
      border-style:solid;
      border-width:1px;
      border-color: #999999;
    }
  </style>
</head>
<body>
  <table class="empty">
  <tr>
    <th></th>
    <th>Title one</th>
    <th>Title two</th>
  </tr>
  <tr>
    <th>Row Title</th>
    <td class="empty">value</td>
    <td class="empty">value</td>
  </tr>
  <tr>
    <th>Row Title</th>
    <td class="empty">value</td>
    <td class="empty"></td>
  </tr>
  </table>
```

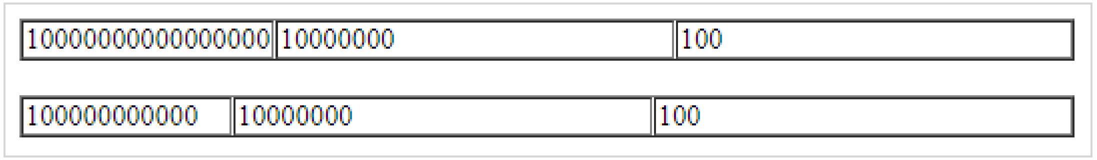


	Title one	Title two
Row Title	value	value
Row Title	value	

CSS Tables cont.

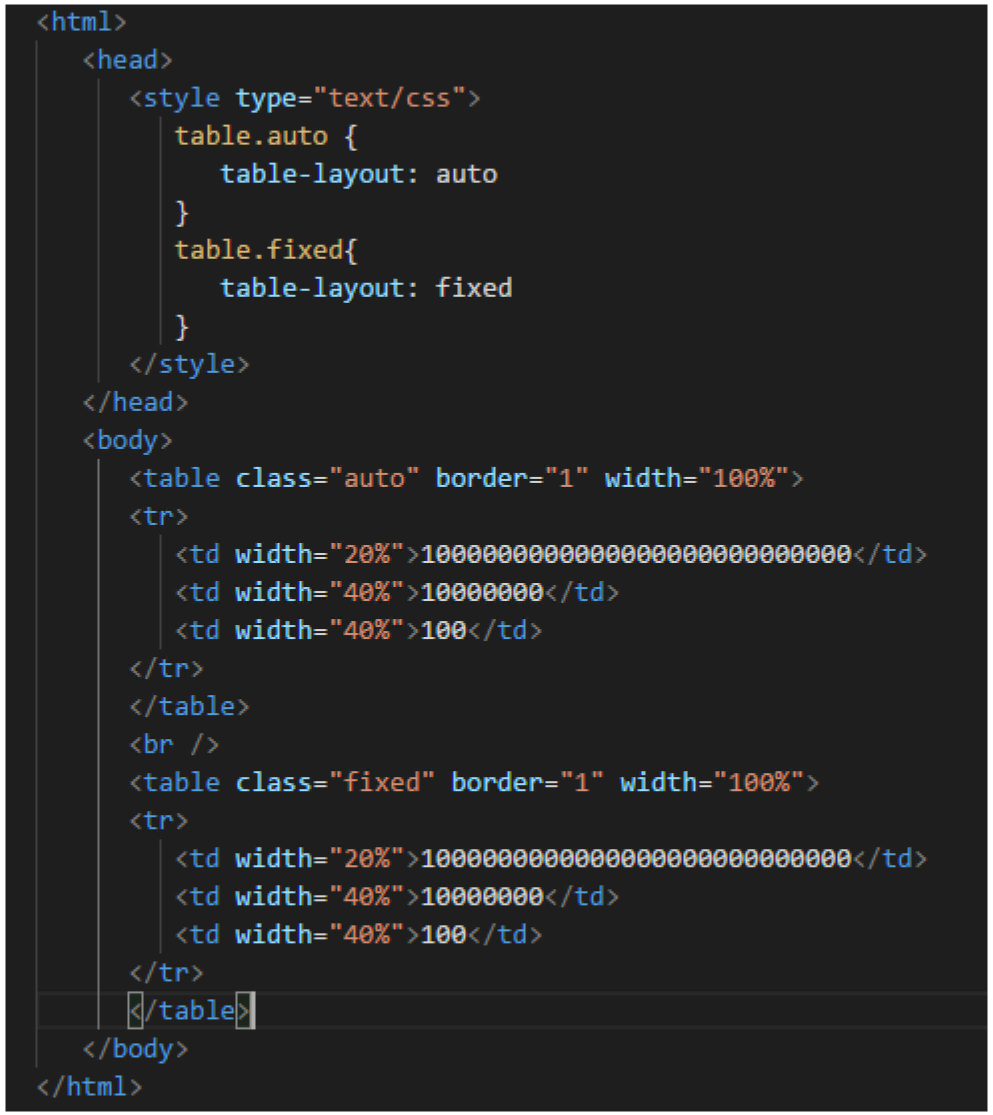
The table-layout Property

- The ***table-layout*** property is supposed to help you control how a browser should render or layout a table.
- This property can have one of the three values: ***fixed, auto*** or ***inherit***.
- The example shows here the difference between these properties.



10000000000000000	10000000	100

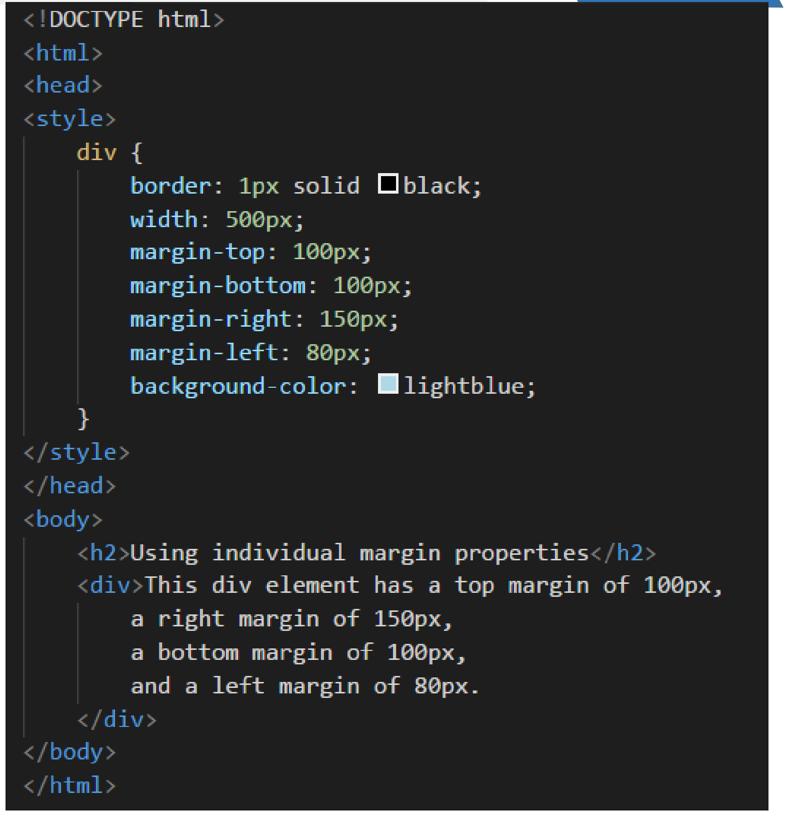
100000000000	10000000	100



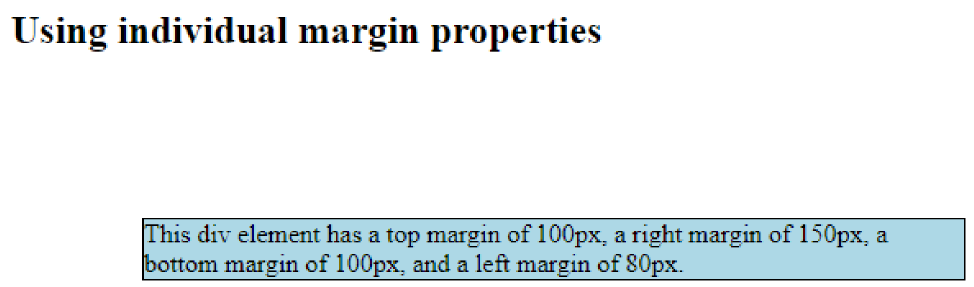
```
<html>
    <head>
        <style type="text/css">
            table.auto {
                table-layout: auto
            }
            table.fixed{
                table-layout: fixed
            }
        </style>
    </head>
    <body>
        <table class="auto" border="1" width="100%">
        <tr>
            <td width="20%">1000000000000000000000000000</td>
            <td width="40%">10000000</td>
            <td width="40%">100</td>
        </tr>
        </table>
        <br />
        <table class="fixed" border="1" width="100%">
        <tr>
            <td width="20%">1000000000000000000000000000</td>
            <td width="40%">10000000</td>
            <td width="40%">100</td>
        </tr>
        </table>
    </body>
</html>
```

CSS Margins

- The CSS margin properties are used to create space around elements, outside of any defined borders.
- CSS has properties for specifying the margin for each side of an element:
    - margin-top
    - margin-right
    - margin-bottom
    - margin-left
- All the margin properties can have the following values:
    - auto - the browser calculates the margin
    - length - specifies a margin in px, pt, cm, etc.
    - % - specifies a margin in % of the width of the containing element
    - inherit - specifies that the margin should be inherited from the parent element
- The example sets different margins for all four sides of a <div> element:



```
<!DOCTYPE html>
<html>
<head>
<style>
    div {
        border: 1px solid ■black;
        width: 500px;
        margin-top: 100px;
        margin-bottom: 100px;
        margin-right: 150px;
        margin-left: 80px;
        background-color: □lightblue;
    }
</style>
</head>
<body>
    <h2>Using individual margin properties</h2>
    <div>This div element has a top margin of 100px,
        a right margin of 150px,
        a bottom margin of 100px,
        and a left margin of 80px.
    </div>
</body>
</html>
```



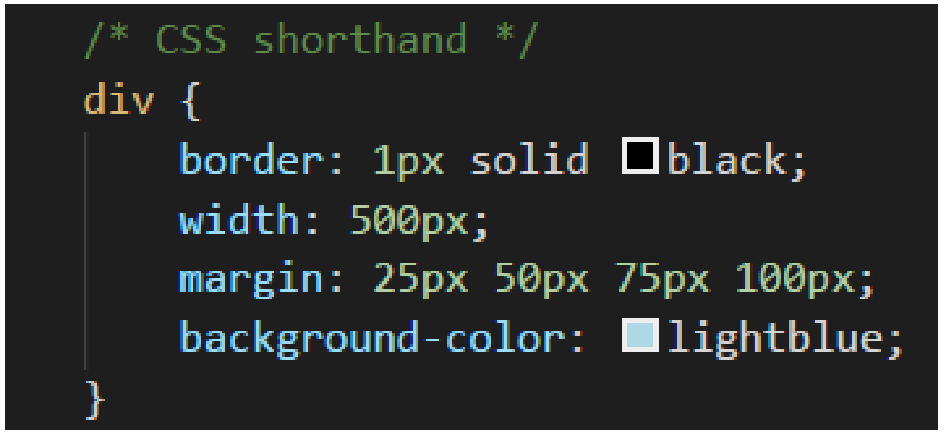
Using individual margin properties

This div element has a top margin of 100px, a right margin of 150px, a bottom margin of 100px, and a left margin of 80px.

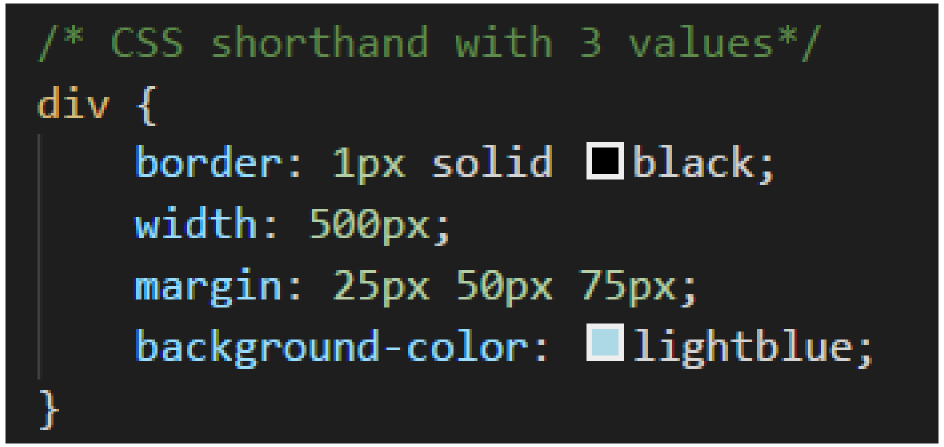
CSS Margins

Margin - Shorthand Property

- To shorten the code, it is possible to specify all the margin properties in one property.
- If the margin property has four values:
  - margin: 25px 50px 75px 100px;
  - top margin is 25px
  - right margin is 50px
  - bottom margin is 75px
  - left margin is 100px
- The example is shown here.
- If the margin property has three values:
  - margin: 25px 50px 75px;
  - top margin is 25px
  - right and left margins are 50px
  - bottom margin is 75px



```css
/* CSS shorthand */
div {
    border: 1px solid ■black;
    width: 500px;
    margin: 25px 50px 75px 100px;
    background-color: □lightblue;
}
```

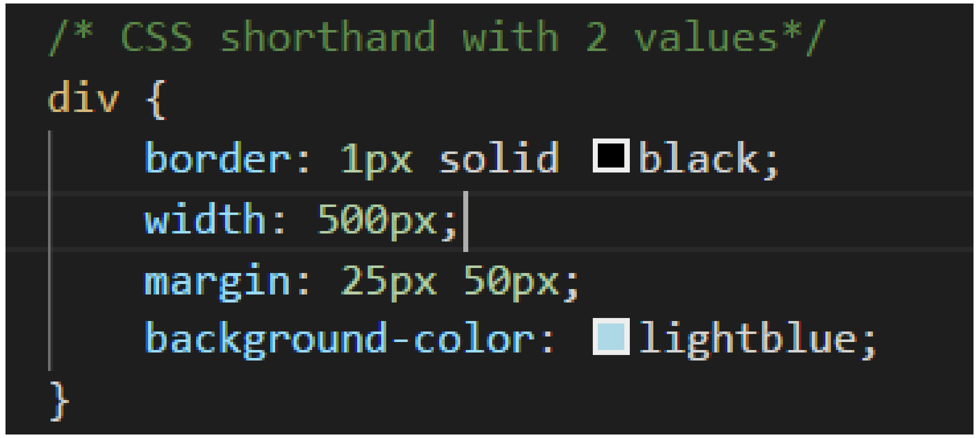


```css
/* CSS shorthand with 3 values*/
div {
    border: 1px solid ■black;
    width: 500px;
    margin: 25px 50px 75px;
    background-color: □lightblue;
}
```

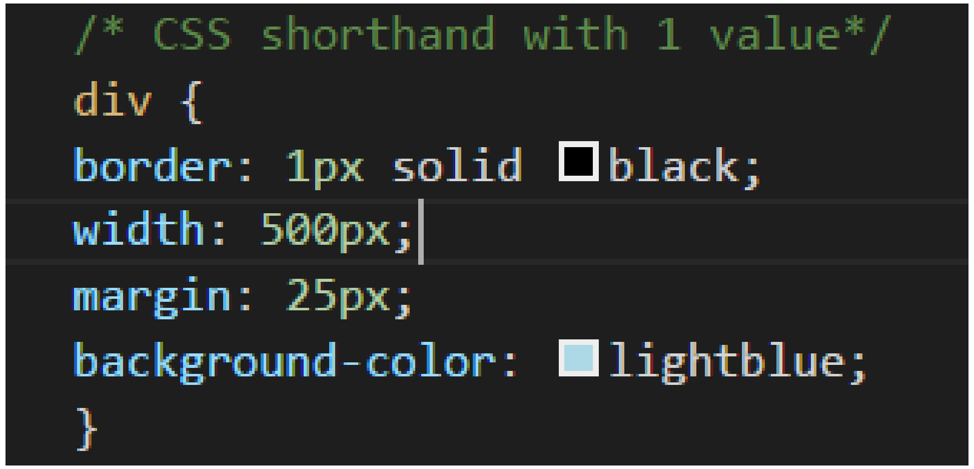
CSS Margins

Margin - Shorthand Property

- If the margin property has two values:
  - margin: 25px 50px;
  - top and bottom margins are 25px
  - right and left margins are 50px
- If the margin property has one value:
  - margin: 25px;
  - all four margins are 25px



```css
/* CSS shorthand with 2 values*/
div {
    border: 1px solid ■black;
    width: 500px;
    margin: 25px 50px;
    background-color: ■lightblue;
}
```

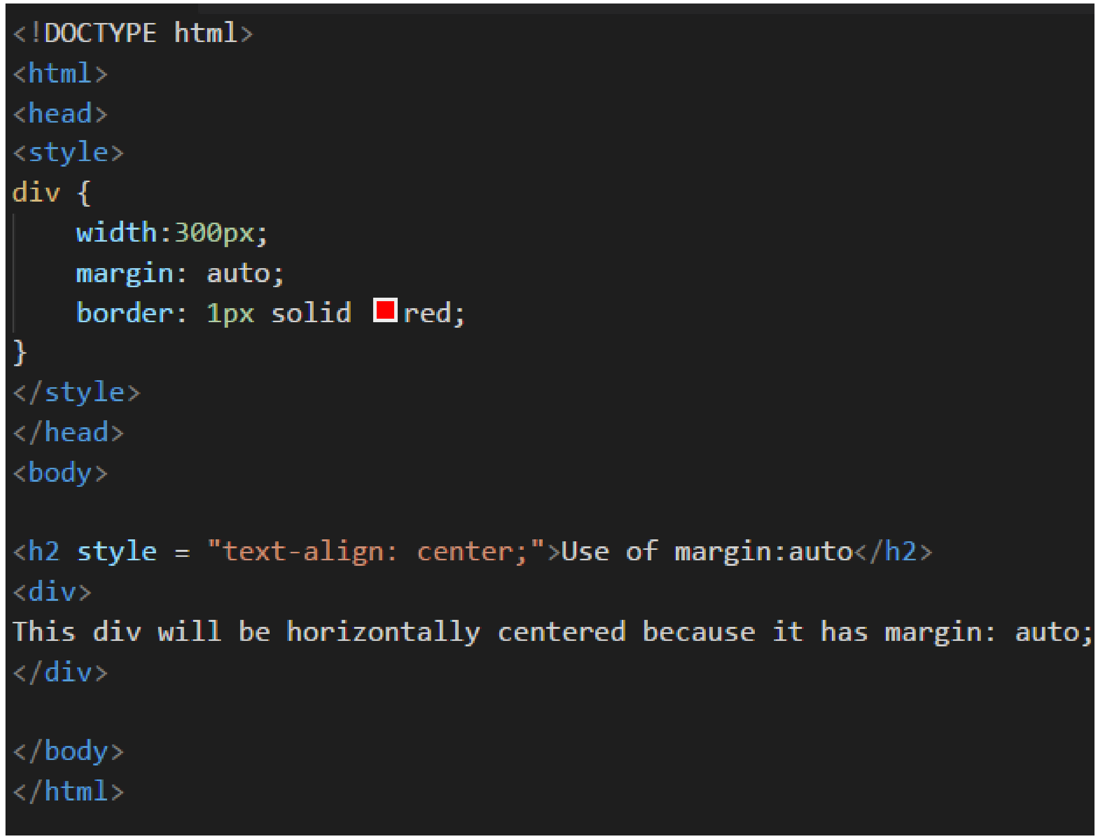


```css
/* CSS shorthand with 1 value*/
div {
border: 1px solid ■black;
width: 500px;
margin: 25px;
background-color: ■lightblue;
}
```

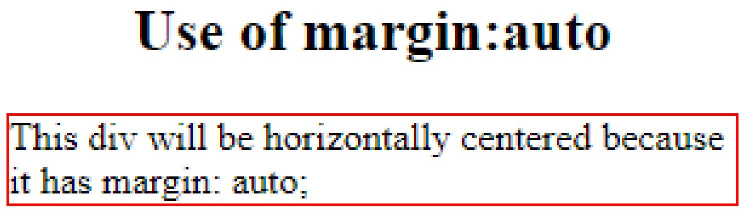
CSS Margins

The Auto Value

- You can set the margin property to auto to horizontally center the element within its container.
- The element will then take up the specified width, and the remaining space will be split equally between the left and right margins:
- Look in to the example shown here.



```
<!DOCTYPE html>
<html>
<head>
<style>
div {
    width:300px;
    margin: auto;
    border: 1px solid 🟥red;
}
</style>
</head>
<body>

<h2 style = "text-align: center;">Use of margin:auto</h2>
<div>
This div will be horizontally centered because it has margin: auto;
</div>

</body>
</html>
```



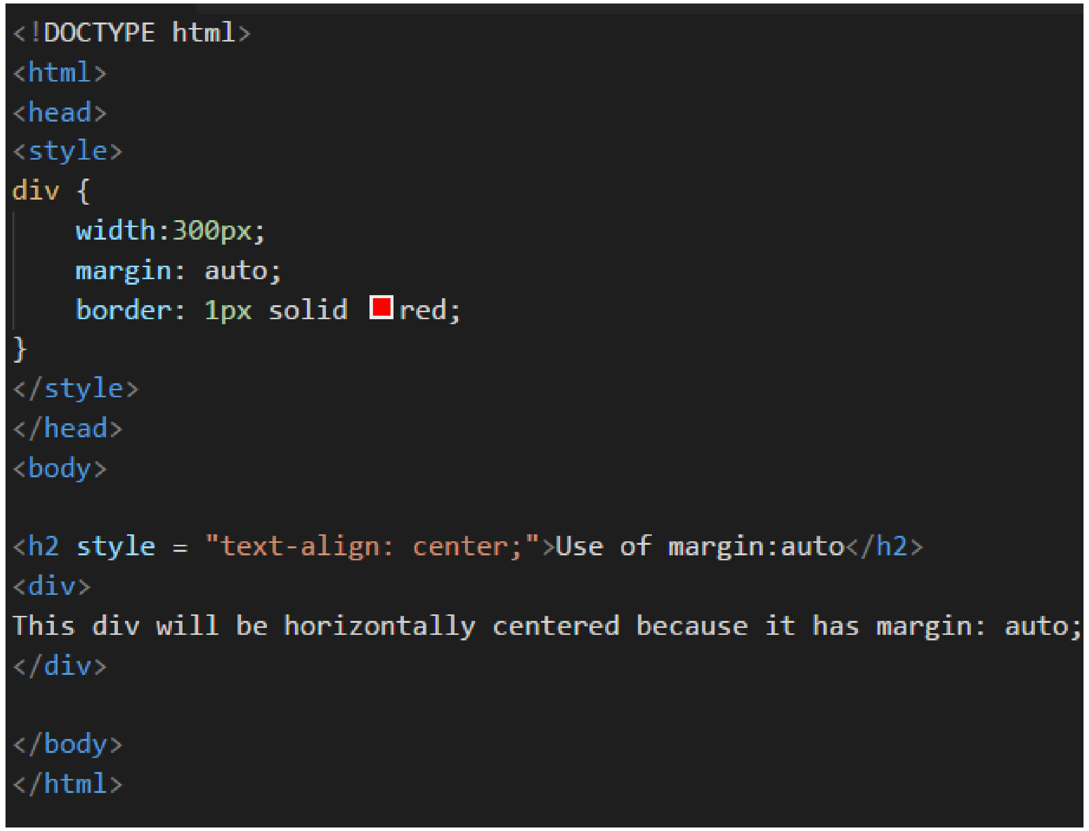
Use of margin:auto

This div will be horizontally centered because it has margin: auto;

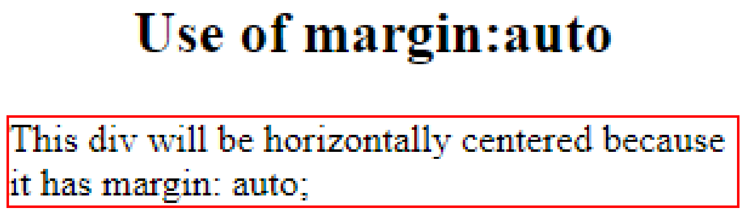
CSS Margins

The Auto Value

- You can set the margin property to auto to horizontally center the element within its container.
- The element will then take up the specified width, and the remaining space will be split equally between the left and right margins:
- Look in to the example shown here.



```
<!DOCTYPE html>
<html>
<head>
<style>
div {
    width:300px;
    margin: auto;
    border: 1px solid 🟥red;
}
</style>
</head>
<body>

<h2 style = "text-align: center;">Use of margin:auto</h2>
<div>
This div will be horizontally centered because it has margin: auto;
</div>

</body>
</html>
```



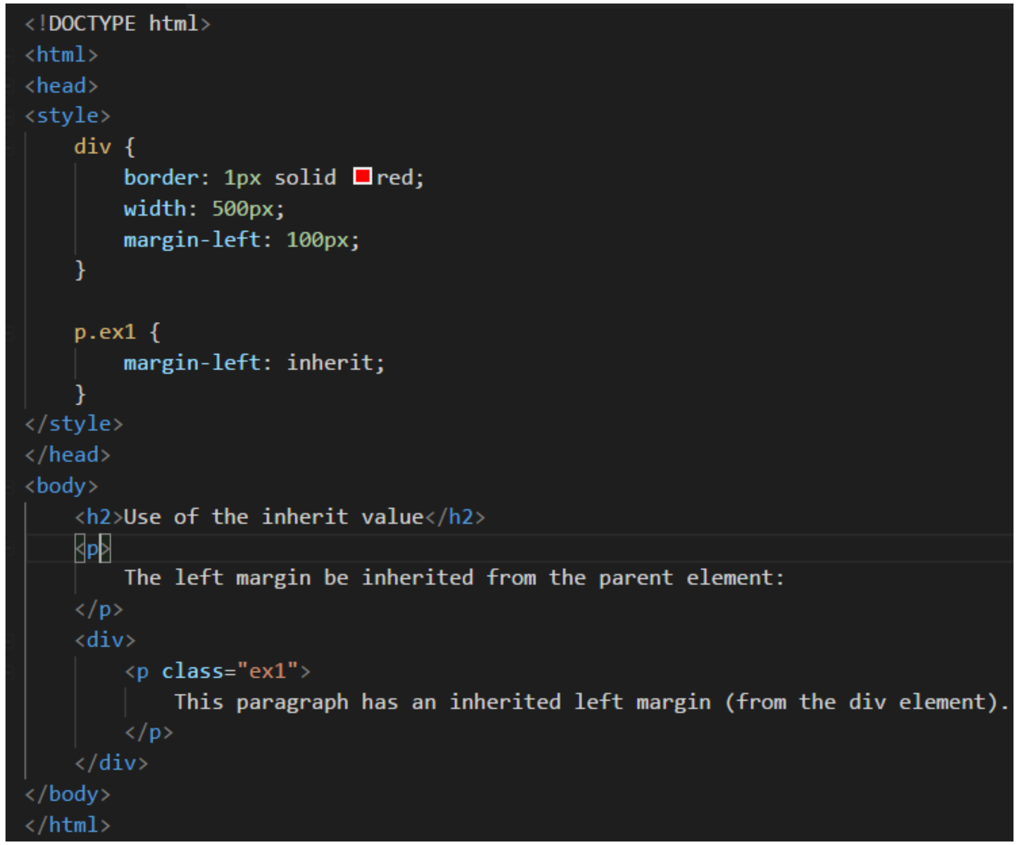
Use of margin:auto

This div will be horizontally centered because it has margin: auto;

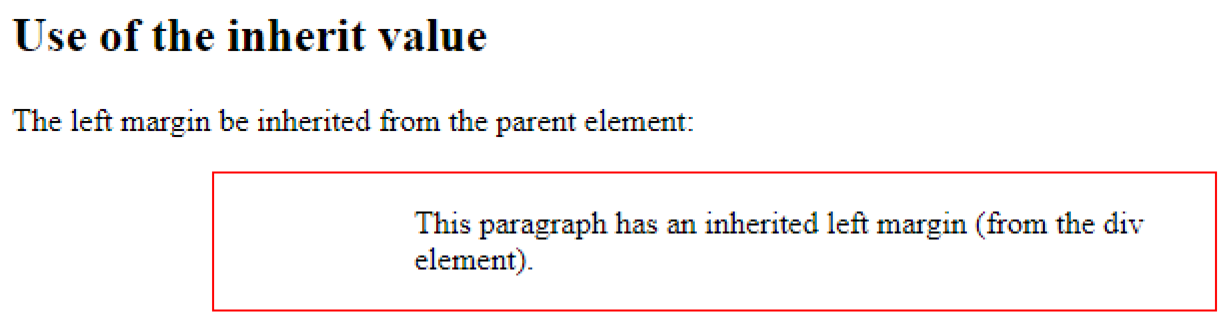
CSS Margins

The Inherit Value

- The inherit Value allows us to inherit the parent element property to other elements.
- This example lets the left margin of the <p class="ex1"> element be inherited from the parent element (<div>):



```
<!DOCTYPE html>
<html>
<head>
<style>
    div {
        border: 1px solid ■red;
        width: 500px;
        margin-left: 100px;
    }

    p.ex1 {
        margin-left: inherit;
    }
</style>
</head>
<body>
    <h2>Use of the inherit value</h2>
    <p>
        The left margin be inherited from the parent element:
    </p>
    <div>
        <p class="ex1">
            This paragraph has an inherited left margin (from the div element).
        </p>
    </div>
</body>
</html>
```



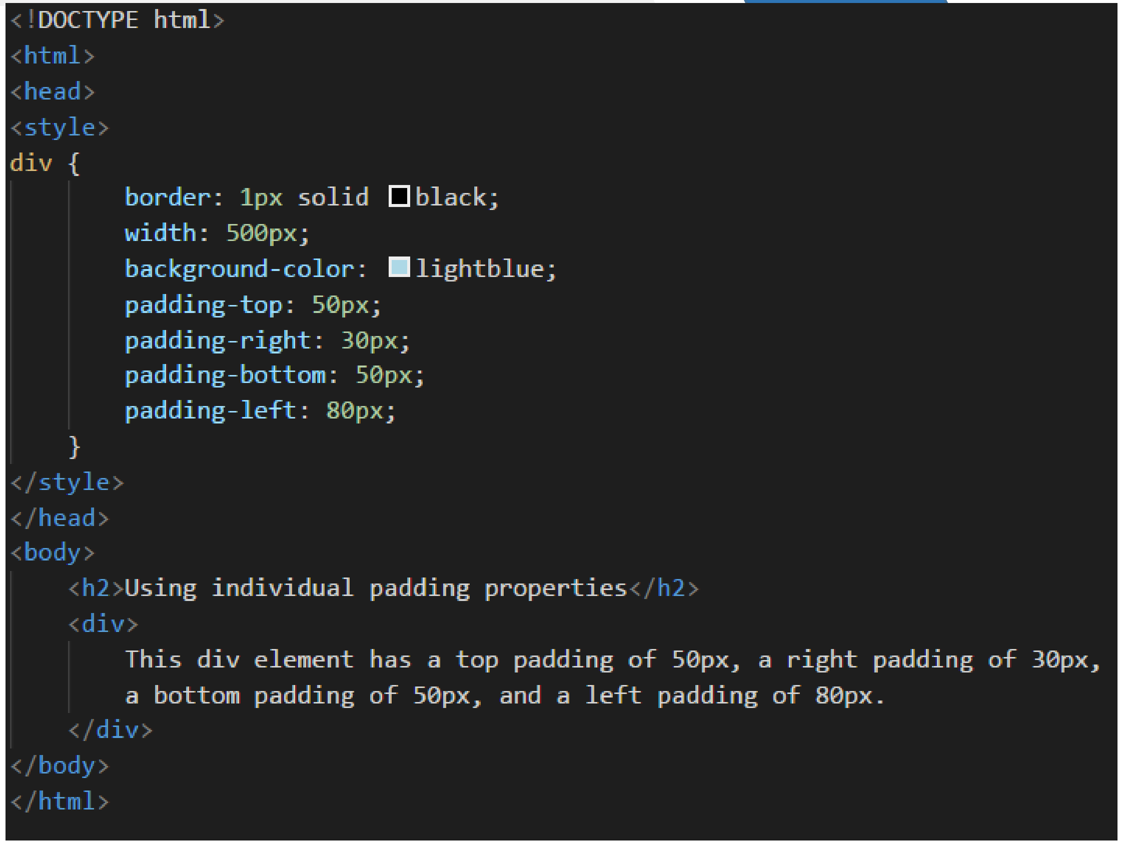
Use of the inherit value

The left margin be inherited from the parent element:

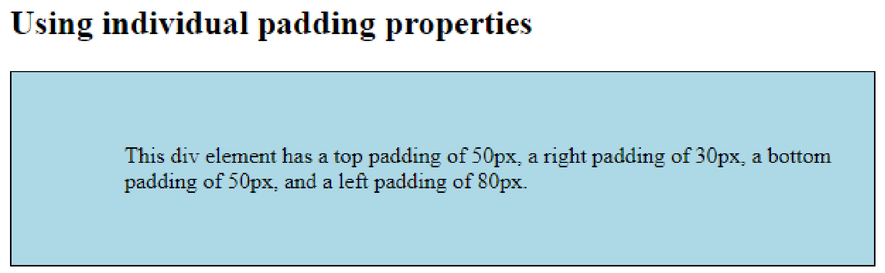
This paragraph has an inherited left margin (from the div element).

CSS Padding

- The CSS padding properties are generate space around an element's content, inside of any defined borders.
- CSS has properties for specifying the padding for each side of an element:
  - padding-top
  - padding-right
  - padding-bottom
  - padding-left
- All the padding properties can have the following values:
  - length - specifies a padding in px, pt, cm, etc.
  - % - specifies a padding in % of the width of the containing element
  - inherit - specifies that the padding should be inherited from the parent element
- The example sets different paddings for all four sides of a <div> element:



```
<!DOCTYPE html>
<html>
<head>
<style>
div {
        border: 1px solid ■black;
        width: 500px;
        background-color: ■lightblue;
        padding-top: 50px;
        padding-right: 30px;
        padding-bottom: 50px;
        padding-left: 80px;
    }
</style>
</head>
<body>
    <h2>Using individual padding properties</h2>
    <div>
        This div element has a top padding of 50px, a right padding of 30px,
        a bottom padding of 50px, and a left padding of 80px.
    </div>
</body>
</html>
```



Using individual padding properties

This div element has a top padding of 50px, a right padding of 30px, a bottom padding of 50px, and a left padding of 80px.

CSS Padding

Padding - Shorthand Property

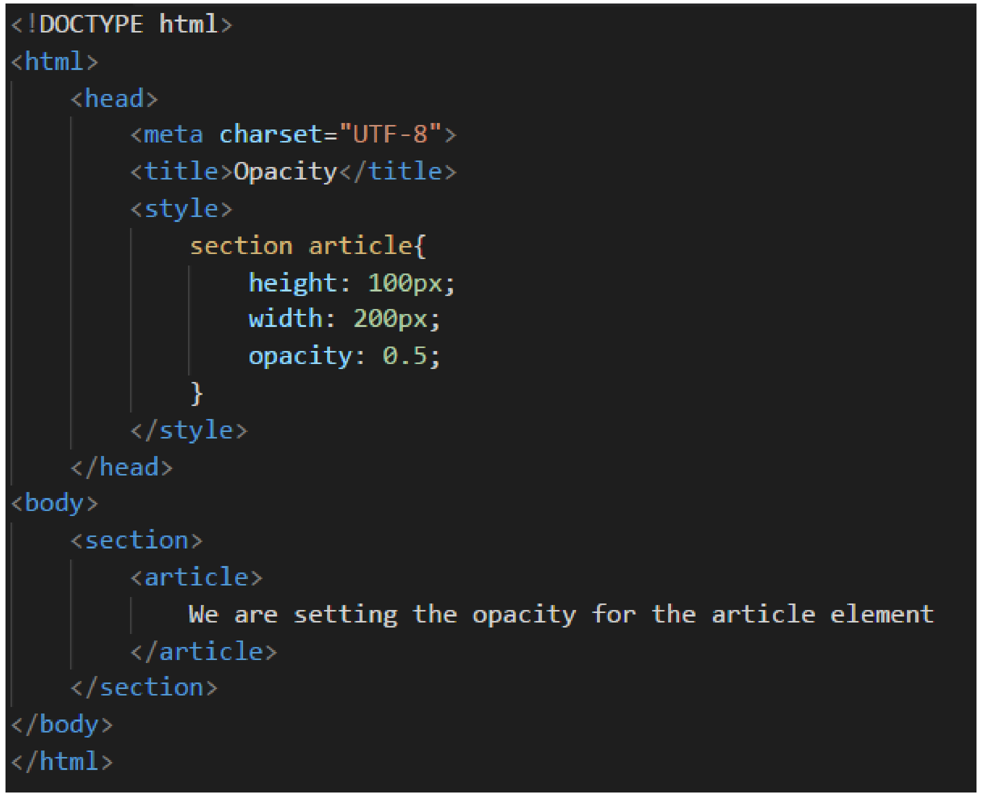
- Like Margin, padding also has shorthand property.
- If the padding property has four values:
    - padding : 25px 50px 75px 100px;
    - top padding is 25px
    - right padding is 50px
    - bottom padding is 75px
    - left padding is 100px
- Similarly, we can set 3, 2 and single values.

Main CSS3.0 Specific Properties

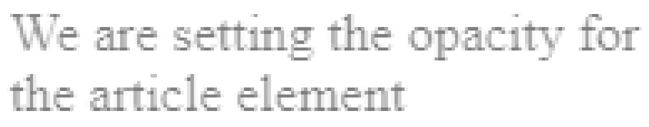
Opacity

Opacity for an element can be easily achieved using CSS3.0

- In this example, we are saying that every article within a section should have opacity of 50%.
- When we set the opacity of an element, we are also setting the opacity of the contents as well and this can have undesired effects.



```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="UTF-8">
        <title>Opacity</title>
        <style>
            section article{
                height: 100px;
                width: 200px;
                opacity: 0.5;
            }
        </style>
    </head>
<body>
    <section>
        <article>
            We are setting the opacity for the article element
        </article>
    </section>
</body>
</html>
```



```
We are setting the opacity for
the article element
```

Main CSS3.0 Specific Properties cont.

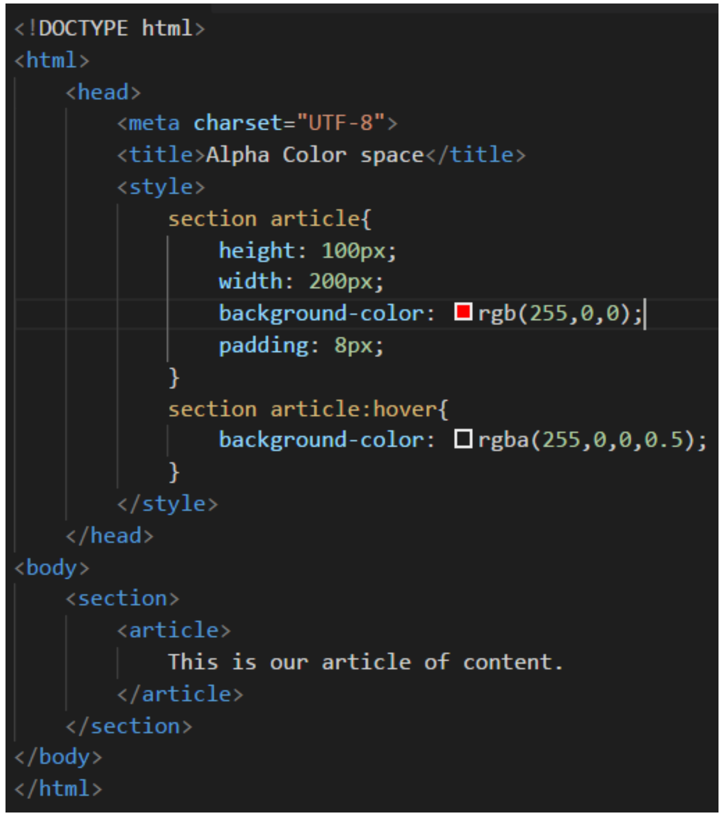
Alpha Color Space

- Instead of using the opacity property in CSS, we can set an ***Alpha Color Space*** by specifying the color or background-color of an element using the rgb color values.
- We can simply add an extra value to our rgb color values(rgba) and have four values for the colors, instead of the usual three.
- The Alpha Color Space value will take a value of between 0 and 1.
- It will specify the opacity of that element.

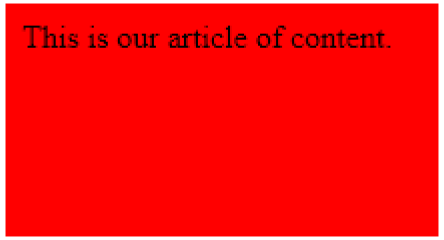
Main CSS3.0 Specific Properties cont.

Lets look in to an example

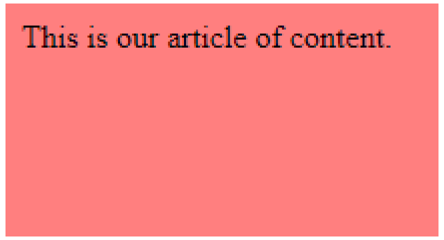
- In the example we have set all of our articles within a section to have a background-color of red.
- When these articles are hovered over, we are setting an Alpha Color Space value of 0.5 using the same color.
- So when we hover over our article elements we will have a background-color that will be slightly transparent.



```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="UTF-8">
        <title>Alpha Color space</title>
        <style>
            section article{
                height: 100px;
                width: 200px;
                background-color: rgb(255,0,0);
                padding: 8px;
            }
            section article:hover{
                background-color: rgba(255,0,0,0.5);
            }
        </style>
    </head>
<body>
    <section>
        <article>
            This is our article of content.
        </article>
    </section>
</body>
</html>
```



This is our article of content.



This is our article of content.

Main CSS3.0 Specific Properties cont.

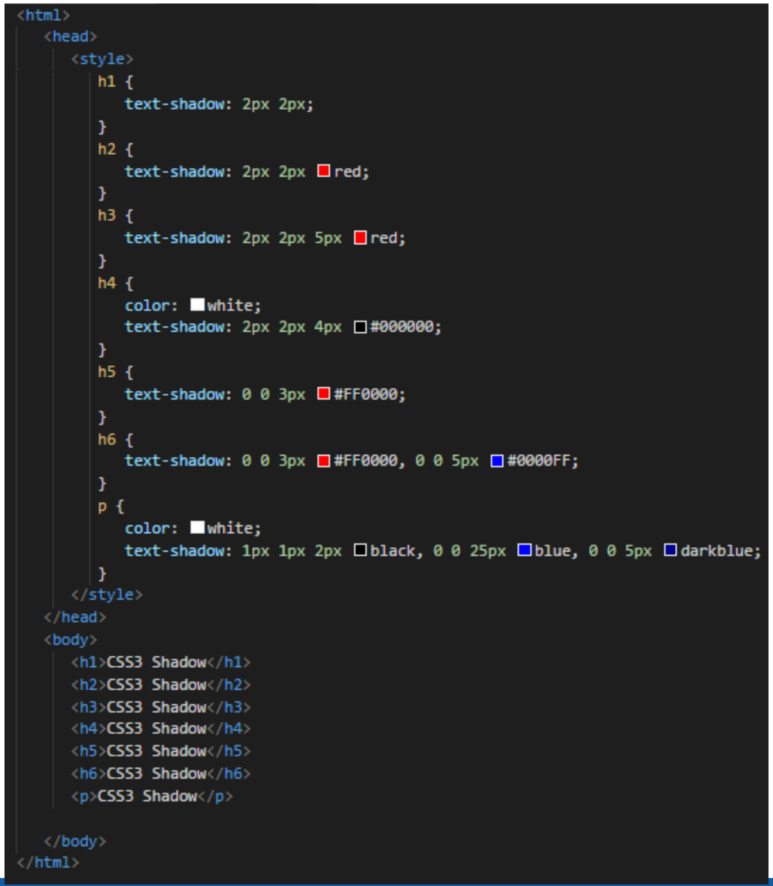
Shadow

- CSS3 supported to add shadow to text or elements.
- Shadow property has divided as follows:
    - Text shadow
    - Box Shadow

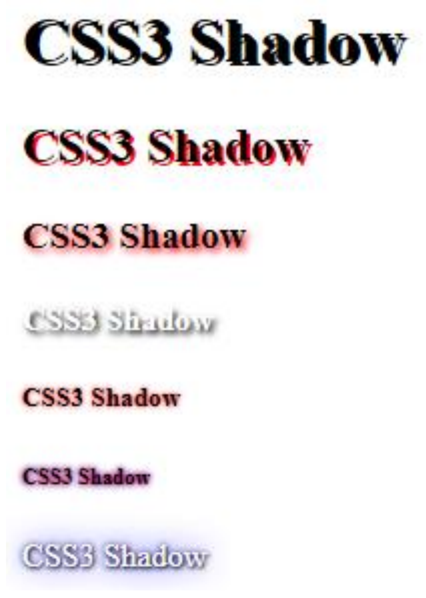
Main CSS3.0 Specific Properties cont.

Text Shadow

- ***Text shadow*** allow us to add shadow effects to text.
- The following example shows different shadow effects and it’s output.



```
<html>
   <head>
      <style>
         h1 {
            text-shadow: 2px 2px;
         }
         h2 {
            text-shadow: 2px 2px red;
         }
         h3 {
            text-shadow: 2px 2px 5px red;
         }
         h4 {
            color: white;
            text-shadow: 2px 2px 4px #000000;
         }
         h5 {
            text-shadow: 0 0 3px #FF0000;
         }
         h6 {
            text-shadow: 0 0 3px #FF0000, 0 0 5px #0000FF;
         }
         p {
            color: white;
            text-shadow: 1px 1px 2px black, 0 0 25px blue, 0 0 5px darkblue;
         }
      </style>
   </head>
   <body>
      <h1>CSS3 Shadow</h1>
      <h2>CSS3 Shadow</h2>
      <h3>CSS3 Shadow</h3>
      <h4>CSS3 Shadow</h4>
      <h5>CSS3 Shadow</h5>
      <h6>CSS3 Shadow</h6>
      <p>CSS3 Shadow</p>

   </body>
</html>
```

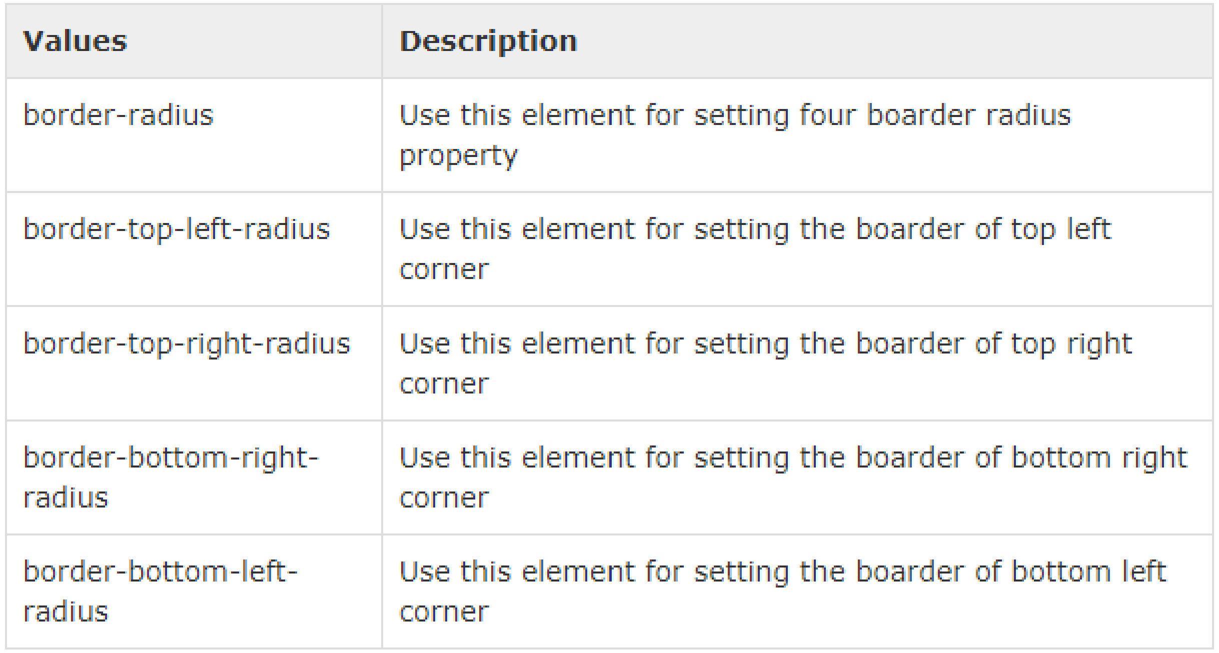


```
CSS3 Shadow
CSS3 Shadow
CSS3 Shadow
CSS3 Shadow
CSS3 Shadow
CSS3 Shadow
CSS3 Shadow
```

Main CSS3.0 Specific Properties cont.

Rounded Corners

- ***Rounded corners*** are used to add special colored corner to body or text by using the border-radius property.
- The table shows the possible values for Rounded corners.

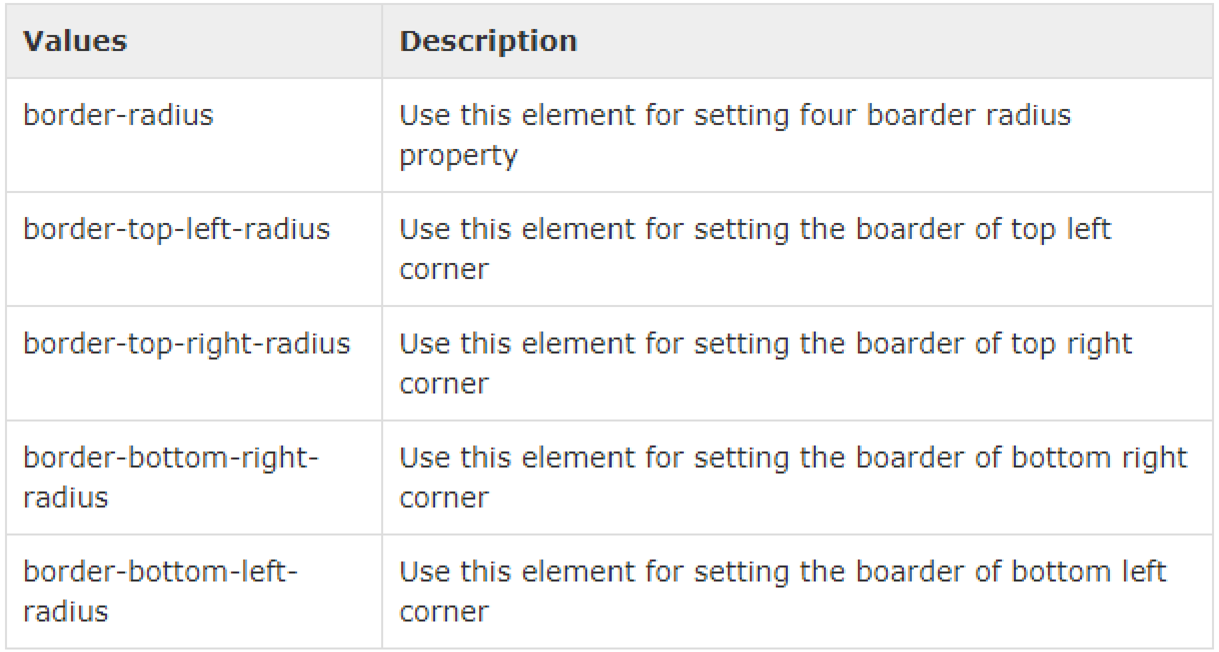


Values	Description
border-radius	Use this element for setting four boarder radius property
border-top-left-radius	Use this element for setting the boarder of top left corner
border-top-right-radius	Use this element for setting the boarder of top right corner
border-bottom-right-radius	Use this element for setting the boarder of bottom right corner
border-bottom-left-radius	Use this element for setting the boarder of bottom left corner

Main CSS3.0 Specific Properties cont.

Rounded Corners

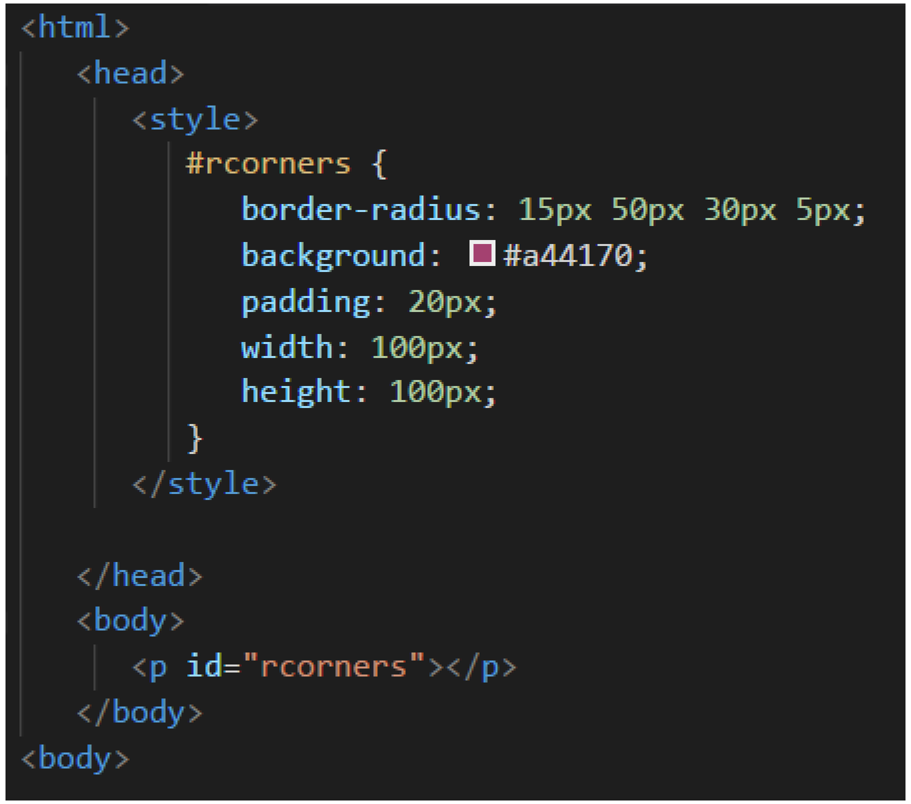
- ***Rounded corners*** are used to add special colored corner to body or text by using the border-radius property.
- The ***border-radius*** property will set a 'border radius' for each of our element’s corners resulting in rounded corners.
- The table shows the possible values for Rounded corners.



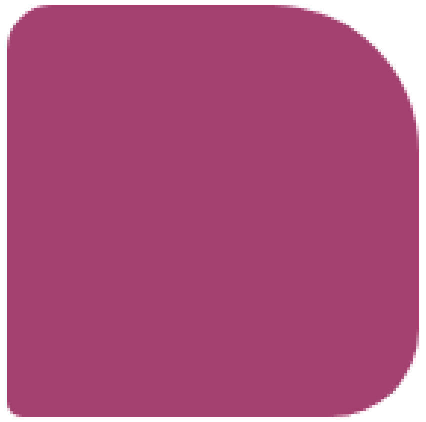
Values	Description
border-radius	Use this element for setting four boarder radius property
border-top-left-radius	Use this element for setting the boarder of top left corner
border-top-right-radius	Use this element for setting the boarder of top right corner
border-bottom-right-radius	Use this element for setting the boarder of bottom right corner
border-bottom-left-radius	Use this element for setting the boarder of bottom left corner

Main CSS3.0 Specific Properties cont.

Lets look in to an example:



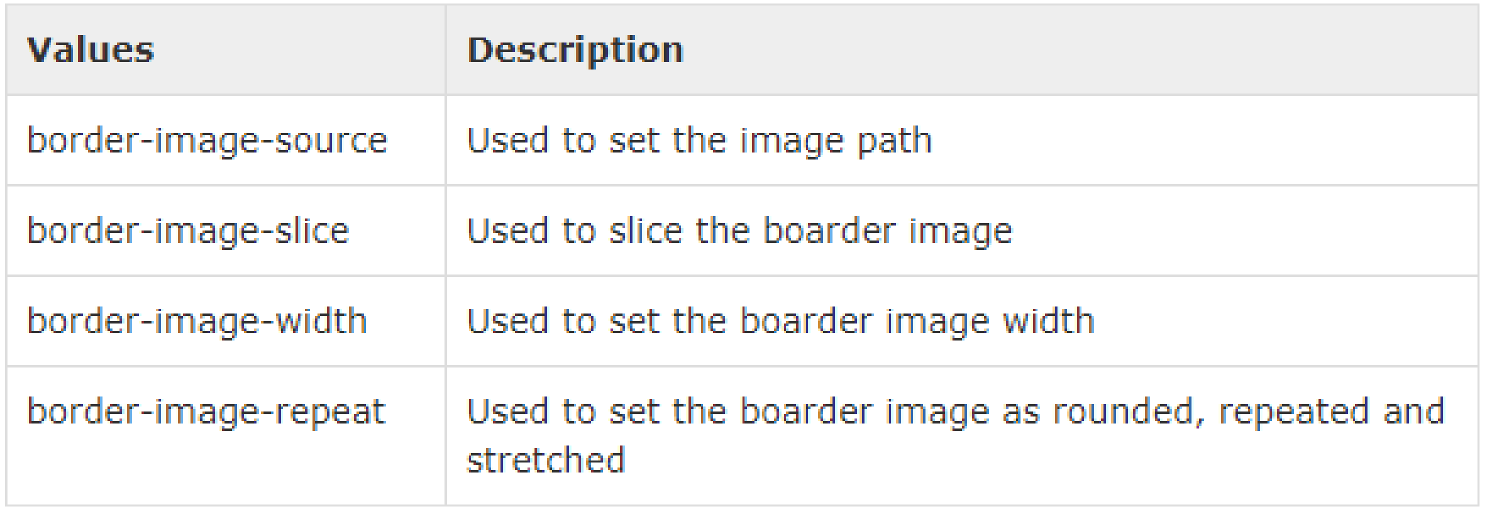
```
<html>
    <head>
        <style>
            #rcorners {
                border-radius: 15px 50px 30px 5px;
                background: #a44170;
                padding: 20px;
                width: 100px;
                height: 100px;
            }
        </style>

    </head>
    <body>
        <p id="rcorners"></p>
    </body>
<body>
```



Main CSS3.0 Specific Properties cont.

Border Images

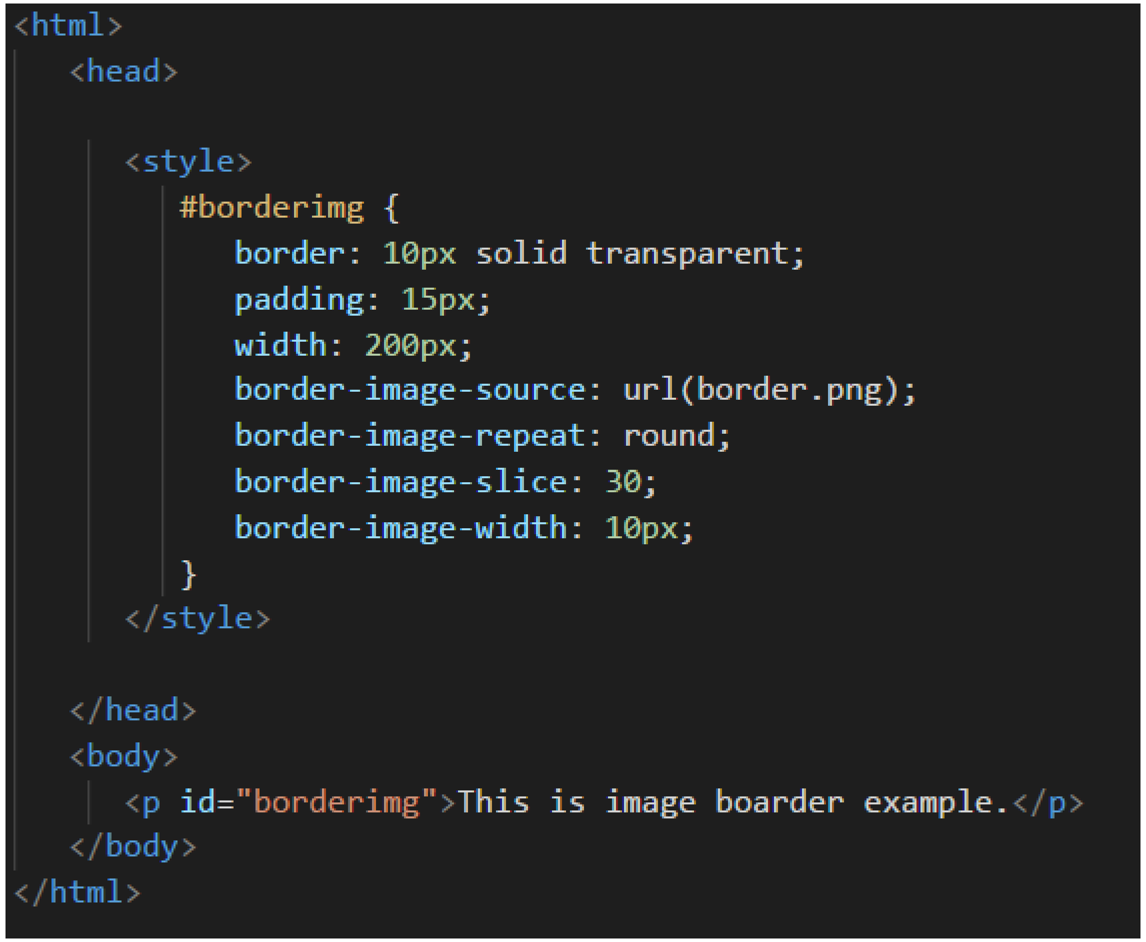
- ***Border image*** property is used to add image boarder to some elements.
- You don't need to use any HTML code to call boarder image.
- The most commonly used values are shown below:



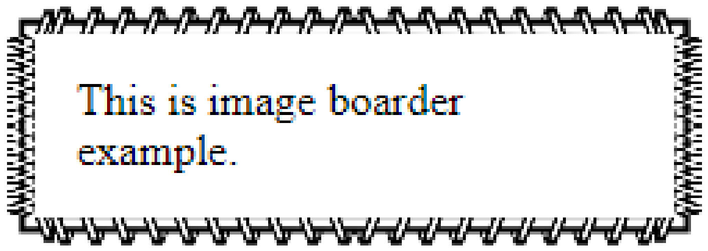
Values	Description
border-image-source	Used to set the image path
border-image-slice	Used to slice the boarder image
border-image-width	Used to set the boarder image width
border-image-repeat	Used to set the boarder image as rounded, repeated and stretched

Main CSS3.0 Specific Properties cont.

The following is an example for border image and its output.



```html
<html>
    <head>

        <style>
            #borderimg {
                border: 10px solid transparent;
                padding: 15px;
                width: 200px;
                border-image-source: url(border.png);
                border-image-repeat: round;
                border-image-slice: 30;
                border-image-width: 10px;
            }
        </style>

    </head>
    <body>
        <p id="borderimg">This is image boarder example.</p>
    </body>
</html>
```



Main CSS3.0 Specific Properties cont.

Gradient

- ***Gradients*** displays the combination of two or more colors.

There are 2 types of ***gradients*** used in CSS3

- Linear Gradients(top/bottom/left/right/diagonally)
- Radial Gradients

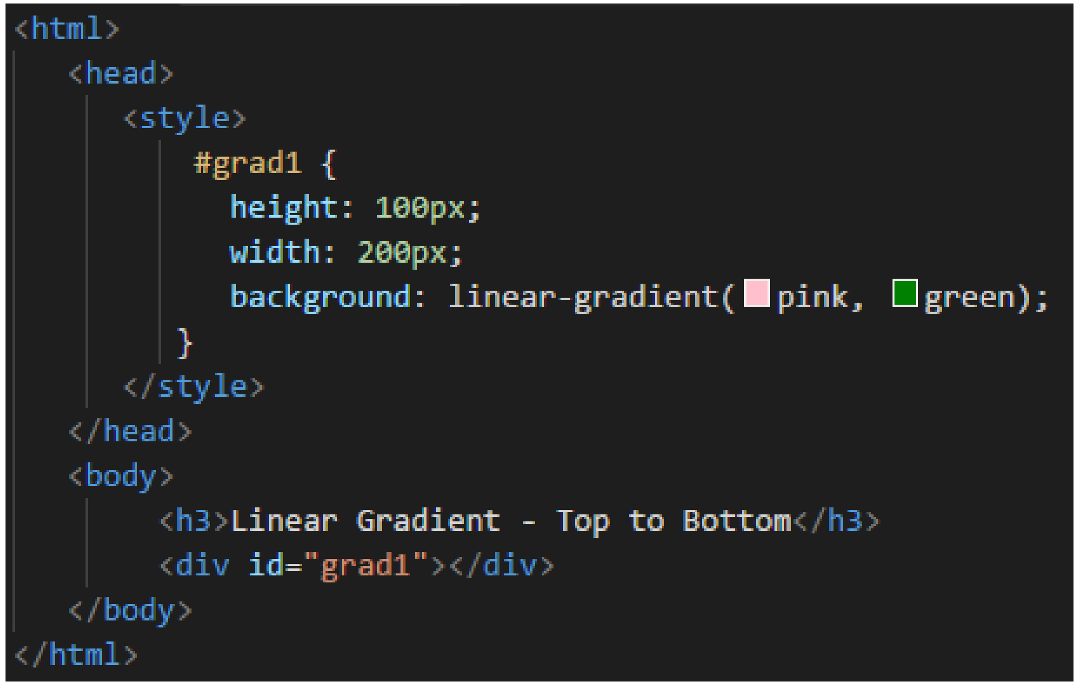
Main CSS3.0 Specific Properties cont.

Linear Gradient

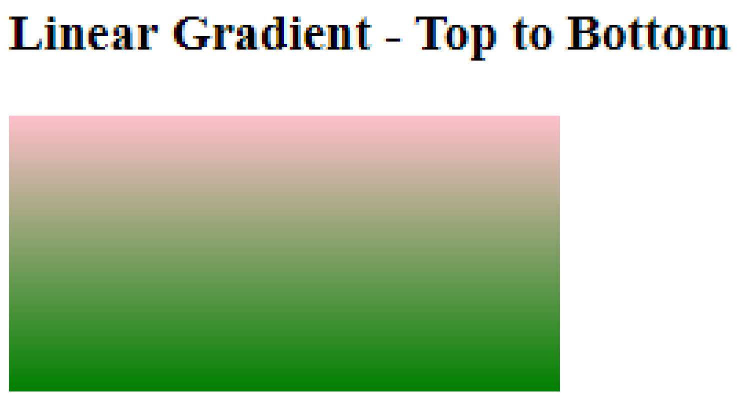
- ***Linear gradients*** are used to arrange two or more colors in linear formats like top to bottom, left to right, diagonal.
- Lets look in to example for each gradients

Main CSS3.0 Specific Properties cont.

Example for Linear Gradient (Top to Bottom)

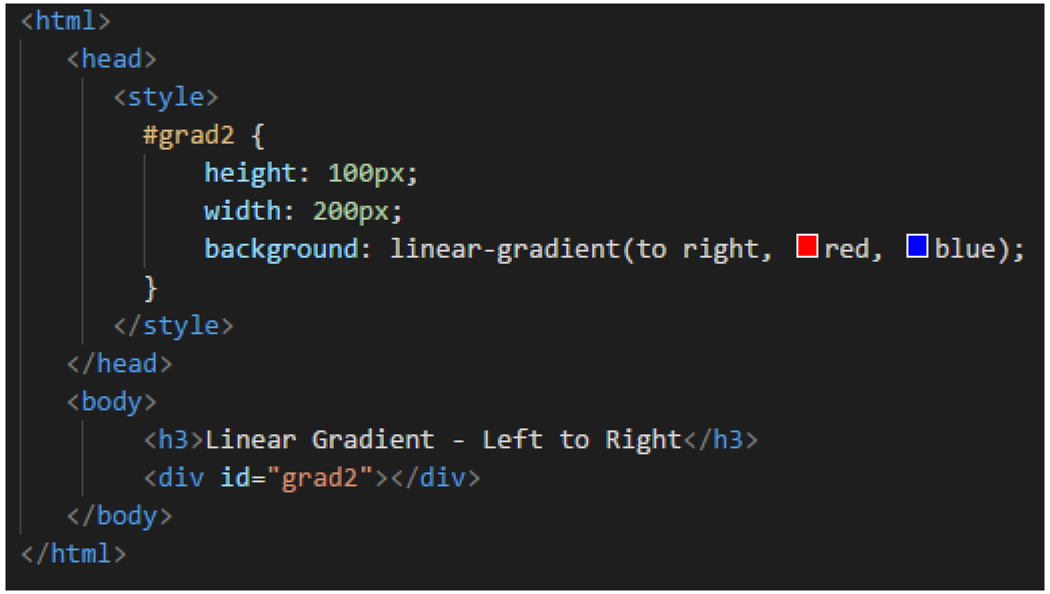


```
<html>
    <head>
        <style>
            #grad1 {
              height: 100px;
              width: 200px;
              background: linear-gradient(pink, green);
            }
        </style>
    </head>
    <body>
        <h3>Linear Gradient - Top to Bottom</h3>
        <div id="grad1"></div>
    </body>
</html>
```

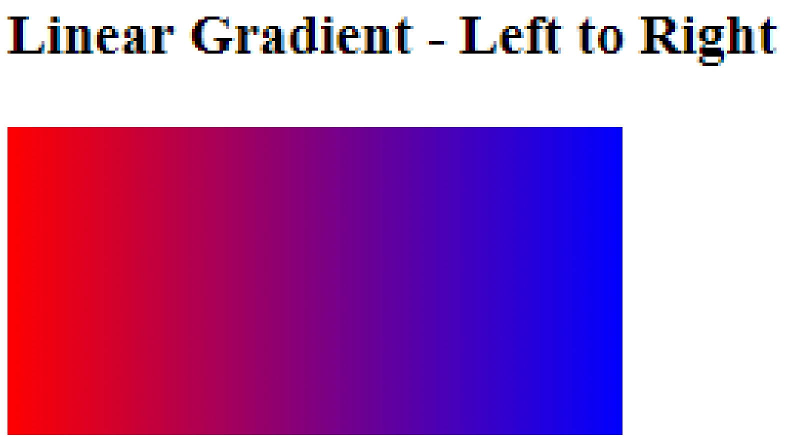


Main CSS3.0 Specific Properties cont.

Example for *Linear Gradient* (Left to right)



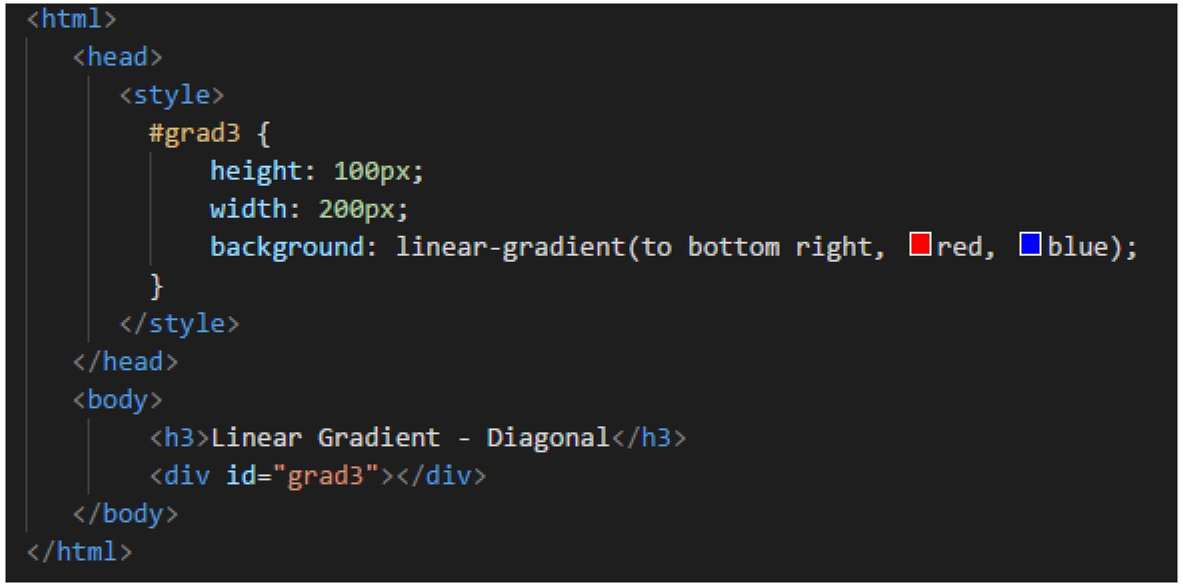
```
<html>
    <head>
        <style>
          #grad2 {
              height: 100px;
              width: 200px;
              background: linear-gradient(to right, 🟥red, 🟦blue);
          }
        </style>
    </head>
    <body>
          <h3>Linear Gradient - Left to Right</h3>
          <div id="grad2"></div>
    </body>
</html>
```



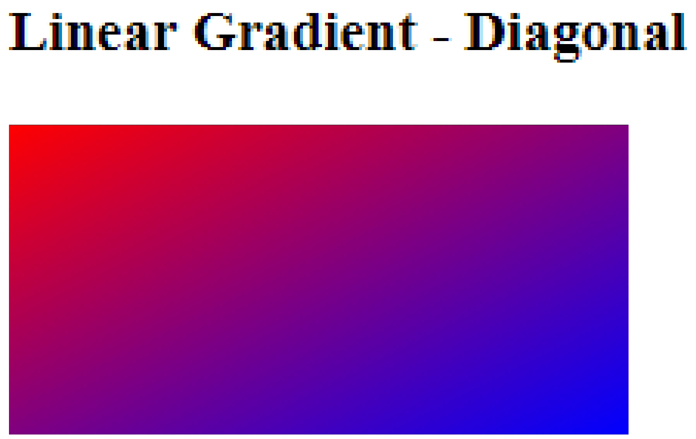
Main CSS3.0 Specific Properties cont.

Example for Linear Gradient (Diagonal)

- ***Diagonal*** starts at top left and right bottom



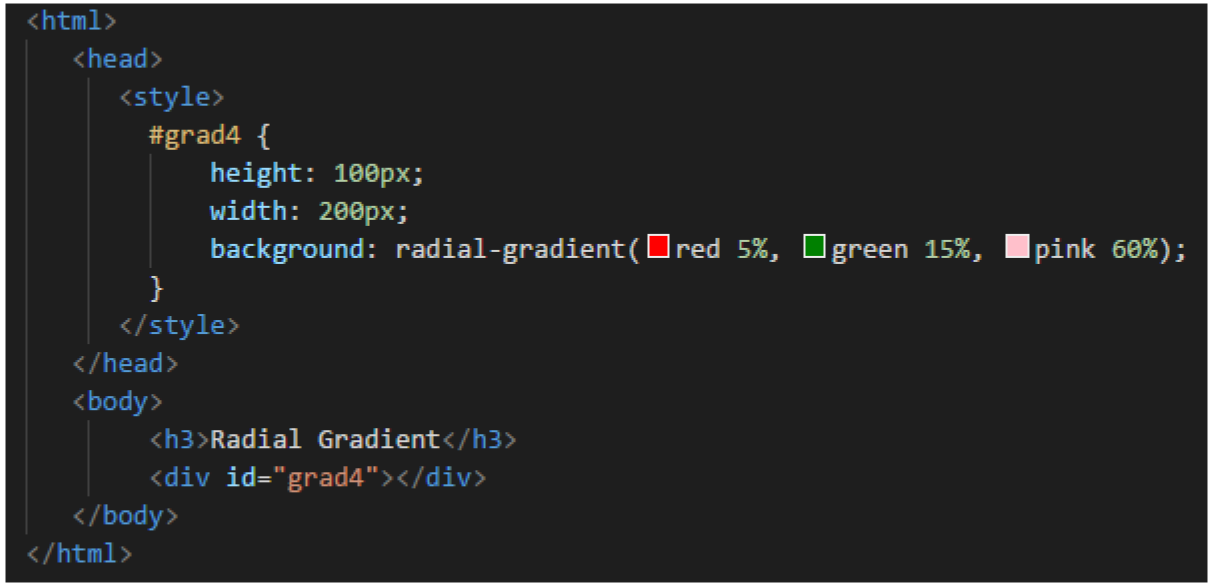
```
<html>
  <head>
    <style>
      #grad3 {
          height: 100px;
          width: 200px;
          background: linear-gradient(to bottom right, 🟥red, 🟦blue);
      }
    </style>
  </head>
  <body>
      <h3>Linear Gradient - Diagonal</h3>
      <div id="grad3"></div>
  </body>
</html>
```



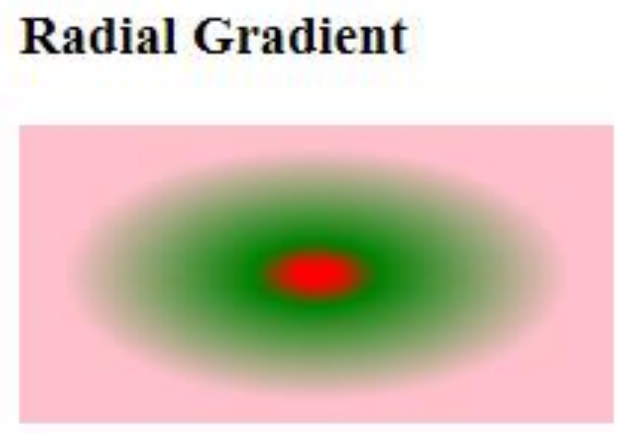
Main CSS3.0 Specific Properties cont.

Example for Radial Gradient

- ***Radial gradients*** appears at center



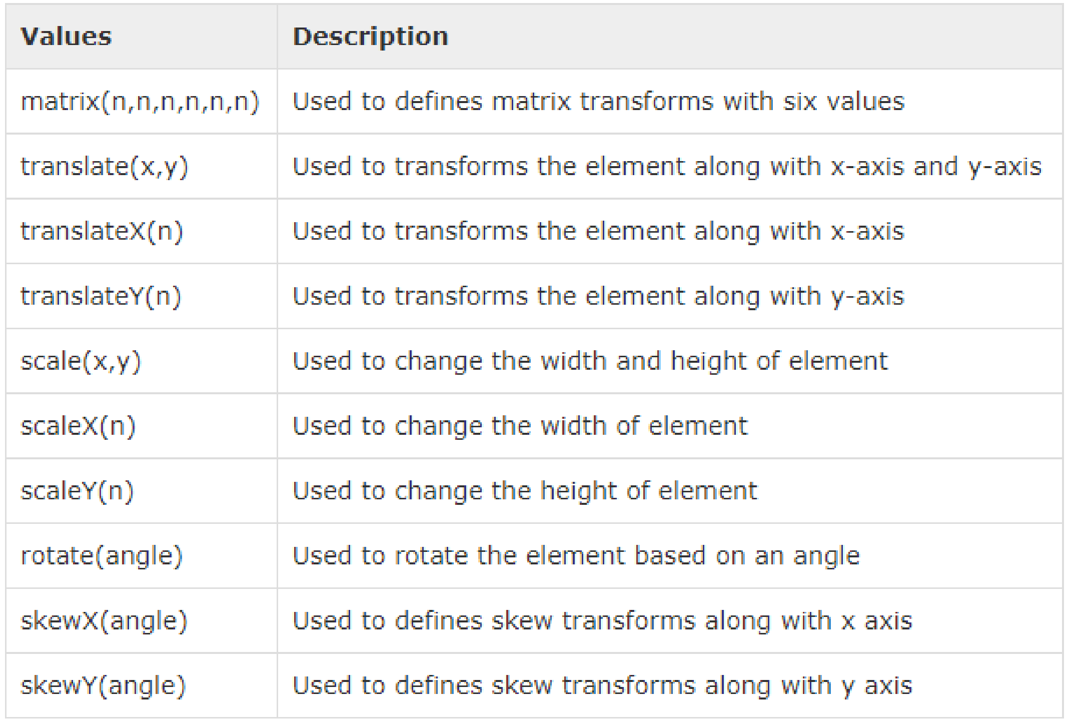
```
<html>
    <head>
        <style>
          #grad4 {
              height: 100px;
              width: 200px;
              background: radial-gradient(🟥red 5%, 🟩green 15%, ⬜pink 60%);
          }
        </style>
    </head>
    <body>
          <h3>Radial Gradient</h3>
          <div id="grad4"></div>
    </body>
</html>
```



Main CSS3.0 Specific Properties cont.

2D Transform

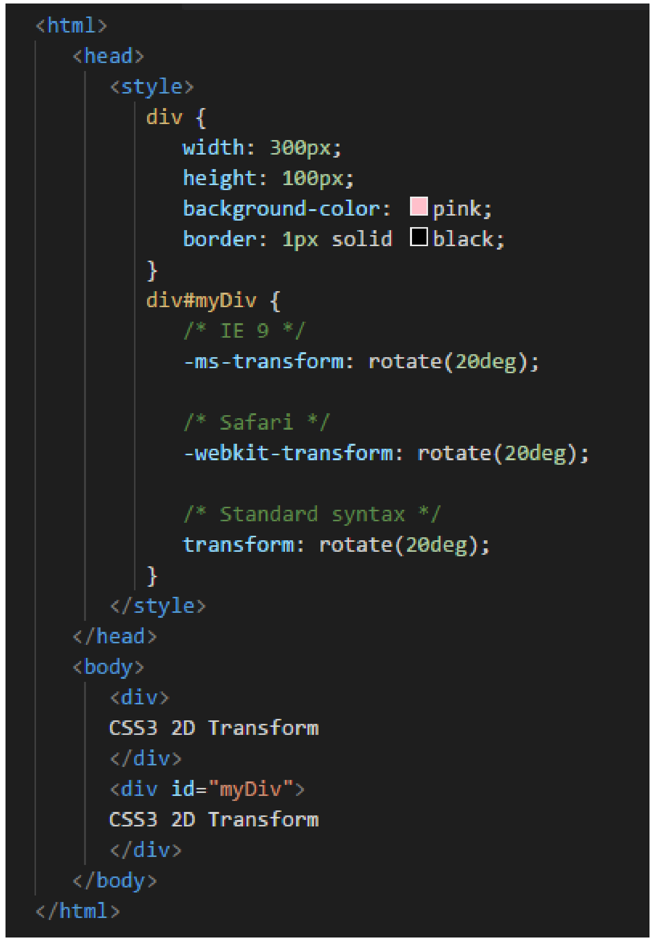
- ***2D transforms*** are used to re-change the element structure as translate, rotate, scale, and skew.
- The following table has contained common values which are used in 2D transforms.



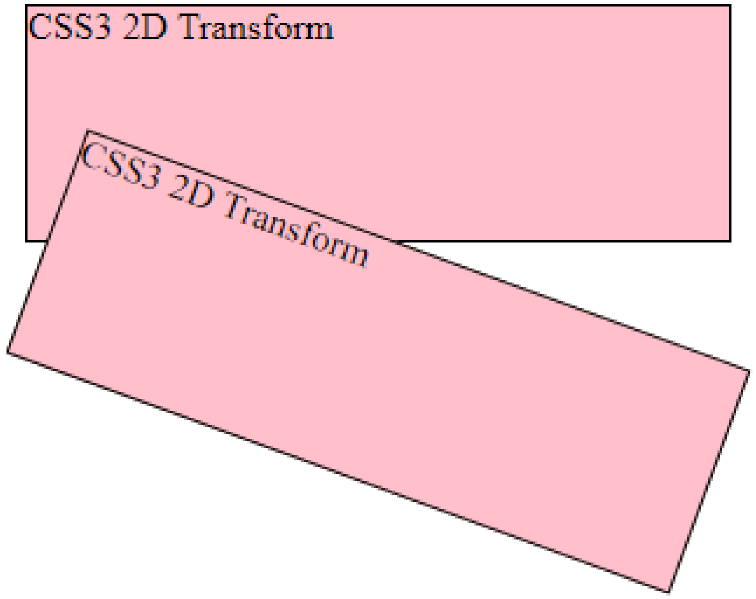
Values	Description
matrix(n,n,n,n,n,n)	Used to defines matrix transforms with six values
translate(x,y)	Used to transforms the element along with x-axis and y-axis
translateX(n)	Used to transforms the element along with x-axis
translateY(n)	Used to transforms the element along with y-axis
scale(x,y)	Used to change the width and height of element
scaleX(n)	Used to change the width of element
scaleY(n)	Used to change the height of element
rotate(angle)	Used to rotate the element based on an angle
skewX(angle)	Used to defines skew transforms along with x axis
skewY(angle)	Used to defines skew transforms along with y axis

Main CSS3.0 Specific Properties cont.

Lets look in to Box rotation with 20 degrees angle.



```
<html>
   <head>
      <style>
         div {
            width: 300px;
            height: 100px;
            background-color: pink;
            border: 1px solid black;
         }
         div#myDiv {
            /* IE 9 */
            -ms-transform: rotate(20deg);

            /* Safari */
            -webkit-transform: rotate(20deg);

            /* Standard syntax */
            transform: rotate(20deg);
         }
      </style>
   </head>
   <body>
      <div>
      CSS3 2D Transform
      </div>
      <div id="myDiv">
      CSS3 2D Transform
      </div>
   </body>
</html>
```

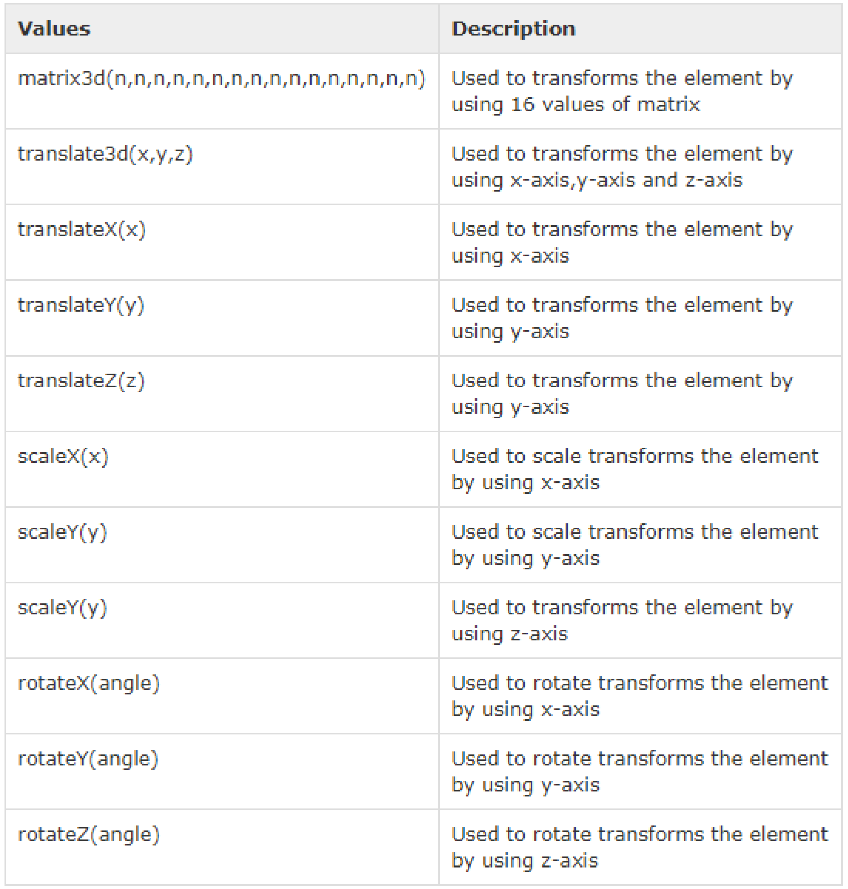


Main CSS3.0 Specific Properties cont.

3D Transform

Using 3D Transform we can move element to x-axis, y-axis and z-axis.

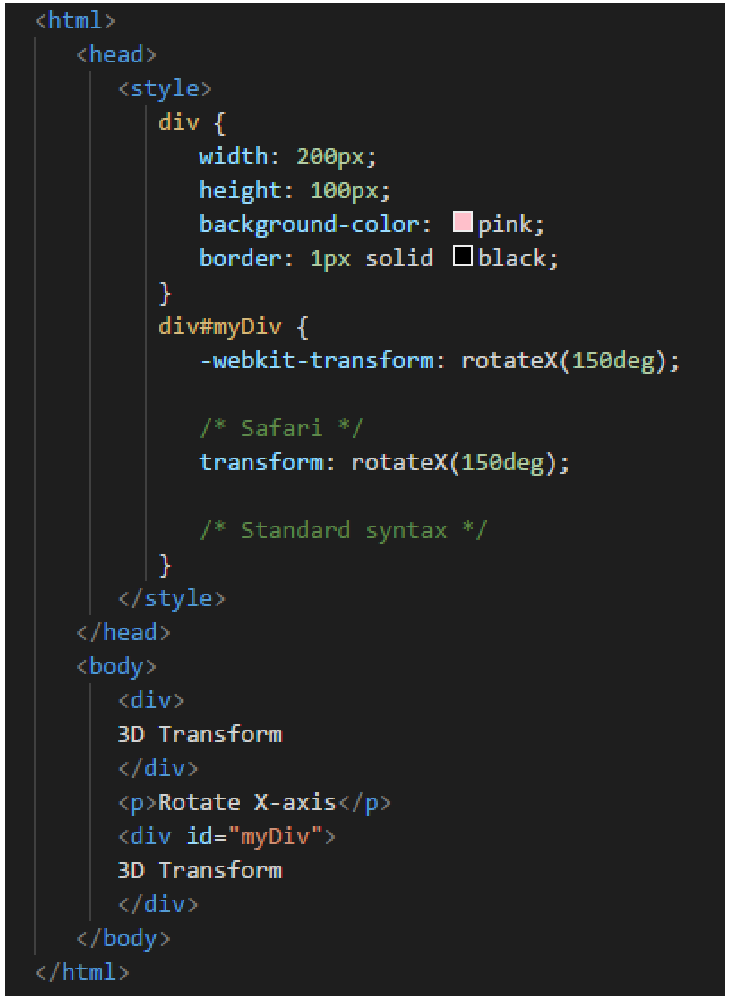
Below methods are used to call 3D transforms:



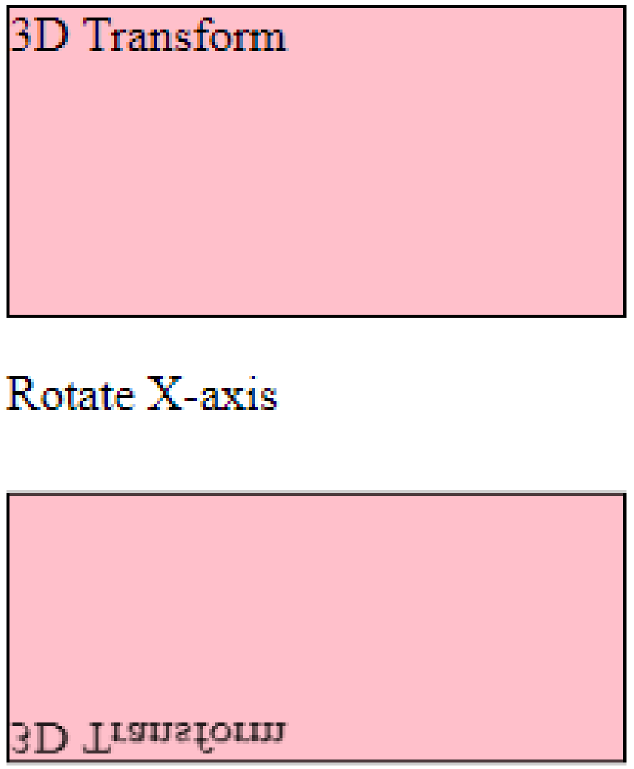
Values	Description
matrix3d(n,n,n,n,n,n,n,n,n,n,n,n,n,n,n,n)	Used to transforms the element by using 16 values of matrix
translate3d(x,y,z)	Used to transforms the element by using x-axis,y-axis and z-axis
translateX(x)	Used to transforms the element by using x-axis
translateY(y)	Used to transforms the element by using y-axis
translateZ(z)	Used to transforms the element by using y-axis
scaleX(x)	Used to scale transforms the element by using x-axis
scaleY(y)	Used to scale transforms the element by using y-axis
scaleY(y)	Used to transforms the element by using z-axis
rotateX(angle)	Used to rotate transforms the element by using x-axis
rotateY(angle)	Used to rotate transforms the element by using y-axis
rotateZ(angle)	Used to rotate transforms the element by using z-axis

Main CSS3.0 Specific Properties cont.

Lets see an example for x-axis 3D transforms.



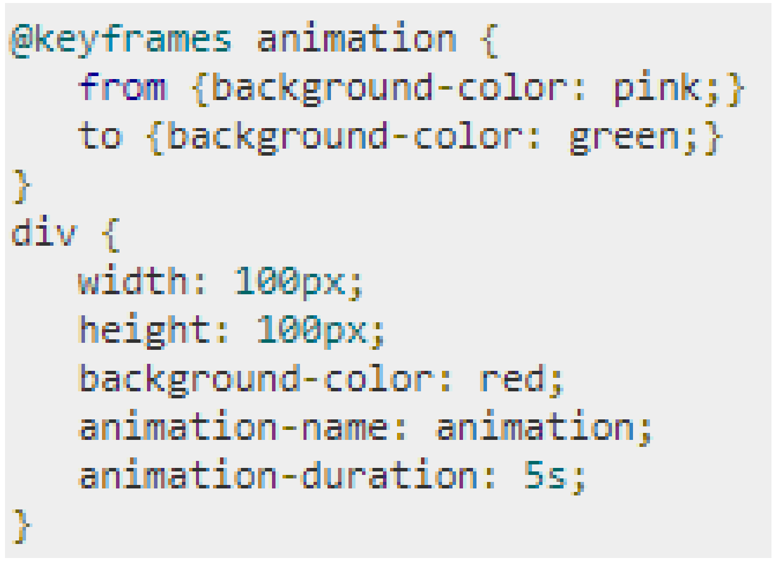
```
<html>
    <head>
        <style>
            div {
                width: 200px;
                height: 100px;
                background-color: pink;
                border: 1px solid black;
            }
            div#myDiv {
                -webkit-transform: rotateX(150deg);

                /* Safari */
                transform: rotateX(150deg);

                /* Standard syntax */
            }
        </style>
    </head>
    <body>
        <div>
        3D Transform
        </div>
        <p>Rotate X-axis</p>
        <div id="myDiv">
        3D Transform
        </div>
    </body>
</html>
```



Main CSS3.0 Specific Properties cont.

Animation

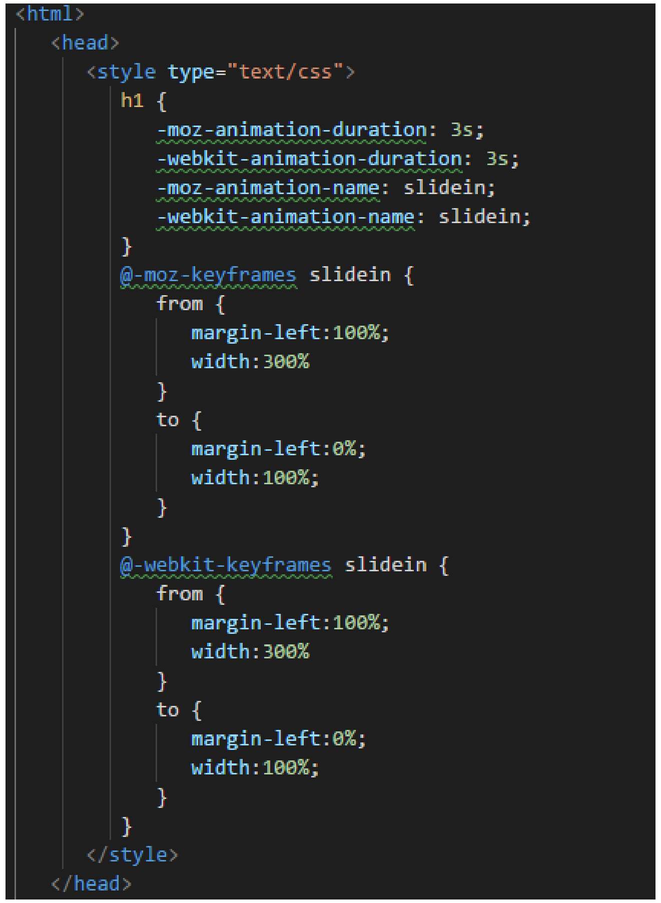
- ***Animation*** is process of making shape changes and creating motions with elements.
- ***Keyframes*** - will control the intermediate animation steps in CSS3.
- The following example shows height, width, color, name and duration of animation with keyframes syntax.



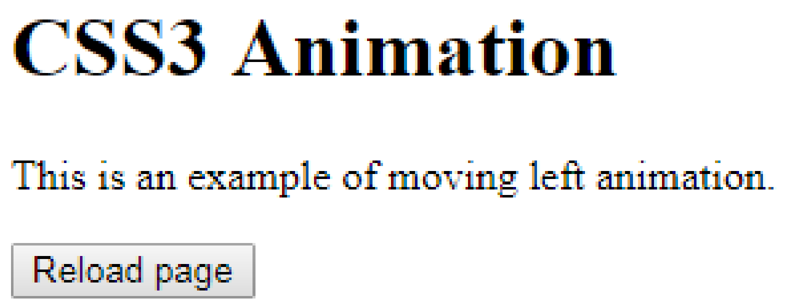
```
@keyframes animation {
   from {background-color: pink;}
   to {background-color: green;}
}
div {
   width: 100px;
   height: 100px;
   background-color: red;
   animation-name: animation;
   animation-duration: 5s;
}
```

Main CSS3.0 Specific Properties cont.

Lets look in to an example for Moving left animation with keyframes



```
<html>
    <head>
        <style type="text/css">
            h1 {
                -moz-animation-duration: 3s;
                -webkit-animation-duration: 3s;
                -moz-animation-name: slidein;
                -webkit-animation-name: slidein;
            }
            @-moz-keyframes slidein {
                from {
                    margin-left:100%;
                    width:300%
                }
                to {
                    margin-left:0%;
                    width:100%;
                }
            }
            @-webkit-keyframes slidein {
                from {
                    margin-left:100%;
                    width:300%
                }
                to {
                    margin-left:0%;
                    width:100%;
                }
            }
        </style>
    </head>
```



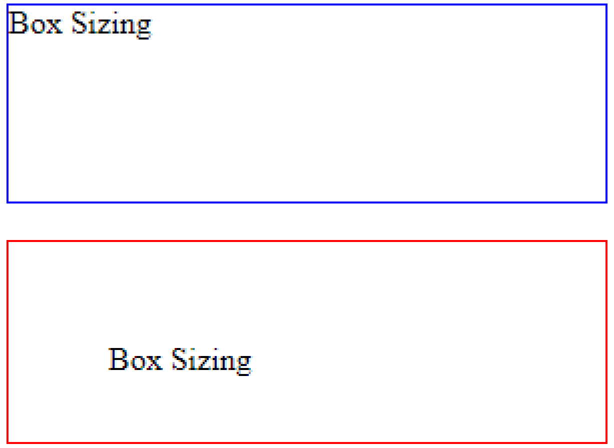
CSS3 Animation

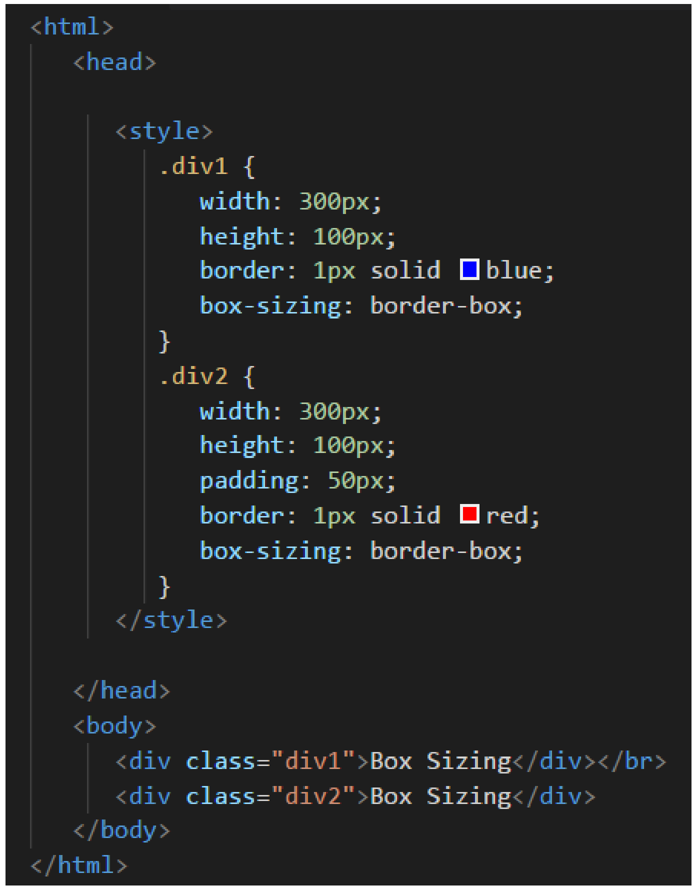
This is an example of moving left animation.

Main CSS3.0 Specific Properties cont.

CSS3 Box Sizing

- ***Box sizing*** property is used to change the height and width of element.
- Unlike in CSS2, when you set the height and width in CSS3, it appears same as the given size of the element.
- The example shows box sizing in CSS3.
- Both elements are having same height and width with box-sizing:border-box, so result is always same for both elements as shown in the example.





```
<html>
   <head>

      <style>
         .div1 {
            width: 300px;
            height: 100px;
            border: 1px solid 🟦blue;
            box-sizing: border-box;
         }
         .div2 {
            width: 300px;
            height: 100px;
            padding: 50px;
            border: 1px solid 🟥red;
            box-sizing: border-box;
         }
      </style>

   </head>
   <body>
      <div class="div1">Box Sizing</div></br>
      <div class="div2">Box Sizing</div>
   </body>
</html>
```

CSS Flexbox

- Flexbox or flexible box is a layout mode of CSS3.
- Using this mode, you can easily create layouts for complex applications and web pages.
- Flexbox layout gives complete control over the direction, alignment, order, size of the boxes.
- Let’s learn how to use the various features available in Flexbox.

CSS Flexbox cont.

Features of Flexbox

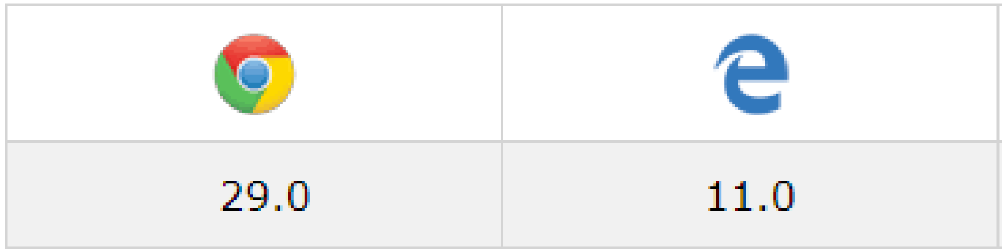
- Following are the notable features of Flexbox layout −
  - ***Direction*** − You can arrange the items on a web page in any direction such as left to right, right to left, top to bottom, and bottom to top.
  - ***Order*** − Using Flexbox, you can rearrange the order of the contents of a web page.
  - ***Wrap*** − In case of inconsistent space for the contents of a web page (in single line), you can wrap them to multiple lines (both horizontally) and vertically.
  - ***Alignment*** − Using Flexbox, you can align the contents of the webpage with respect to their container.
  - ***Resize*** − Using Flexbox, you can increase or decrease the size of the items in the page to fit in available space.

CSS Flexbox cont.

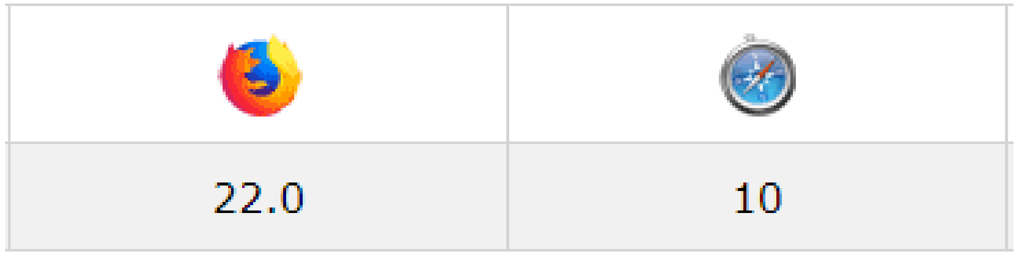
Supporting browsers

Following are the browsers that support Flexbox.

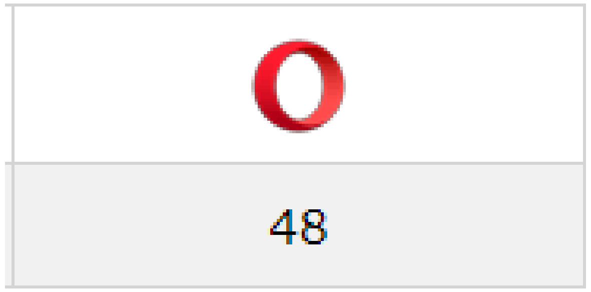
- Chrome 29+
- Firefox 22+
- Internet Explorer 11+
- Opera 12.1+
- Safari 10+
- Android 4.4+
- iOS 7.1+



29.0	11.0



22.0	10



48

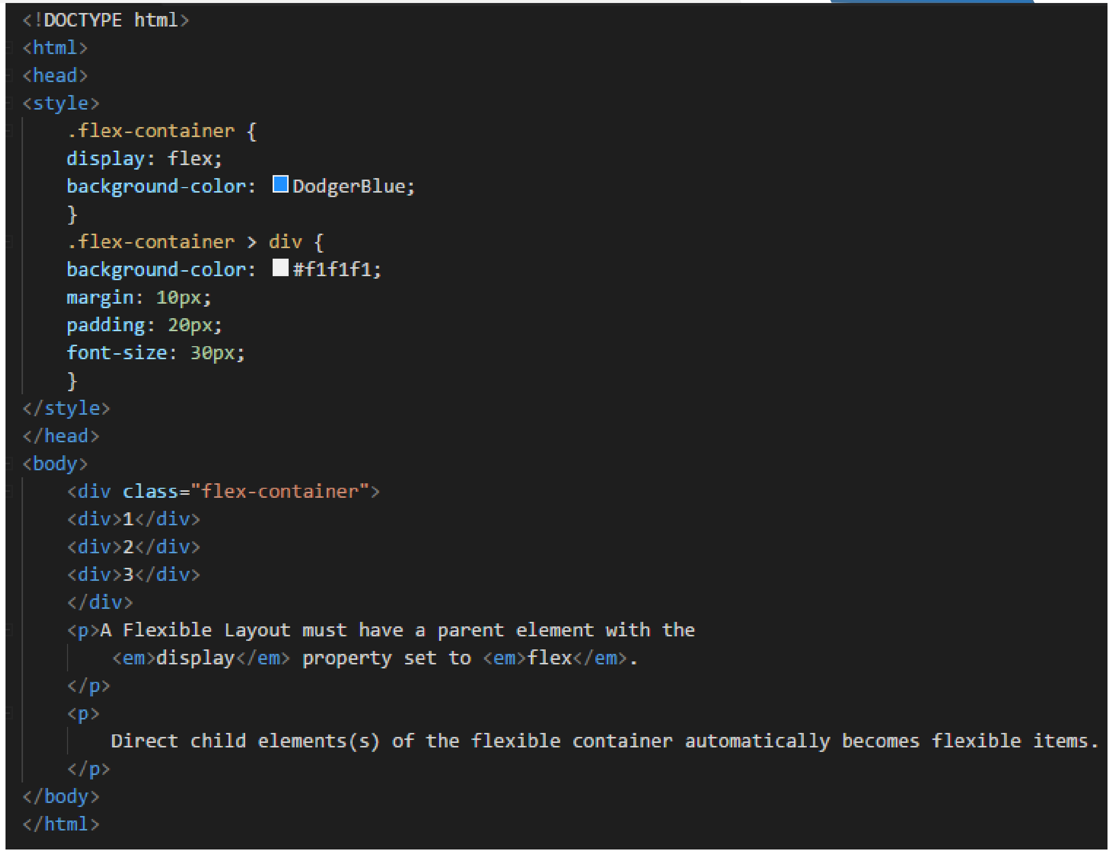
CSS Flexbox cont.

Flex Container (Parent Element)

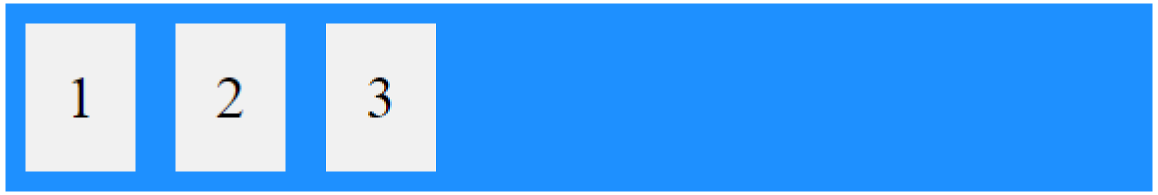
- To use Flexbox in your application, you need to create/define a ***flex container*** using the display property.
- Syntax is:

  display: flex

- The ***flex container*** becomes flexible by setting the display property to ***flex***:
- Direct child elements(s) of the flexible container automatically becomes ***flexible items***.
- The output has a flex container (the blue area) with three flex items.



```html
<!DOCTYPE html>
<html>
<head>
<style>
    .flex-container {
    display: flex;
    background-color: DodgerBlue;
    }
    .flex-container > div {
    background-color: #f1f1f1;
    margin: 10px;
    padding: 20px;
    font-size: 30px;
    }
</style>
</head>
<body>
    <div class="flex-container">
    <div>1</div>
    <div>2</div>
    <div>3</div>
    </div>
    <p>A Flexible Layout must have a parent element with the
        <em>display</em> property set to <em>flex</em>.
    </p>
    <p>
        Direct child elements(s) of the flexible container automatically becomes flexible items.
    </p>
</body>
</html>
```



CSS Flexbox cont.

- The flex container properties are:
  - flex-direction
  - flex-wrap
  - flex-flow
  - justify-content
  - align-items
  - align-content

CSS Flexbox cont.

Flex-direction

- The flex-direction property is used to specify the direction in which the elements of flex container (flex-items) are needed to be placed.
- Syntax is:

  flex-direction: row | row-reverse | column | column-reverse

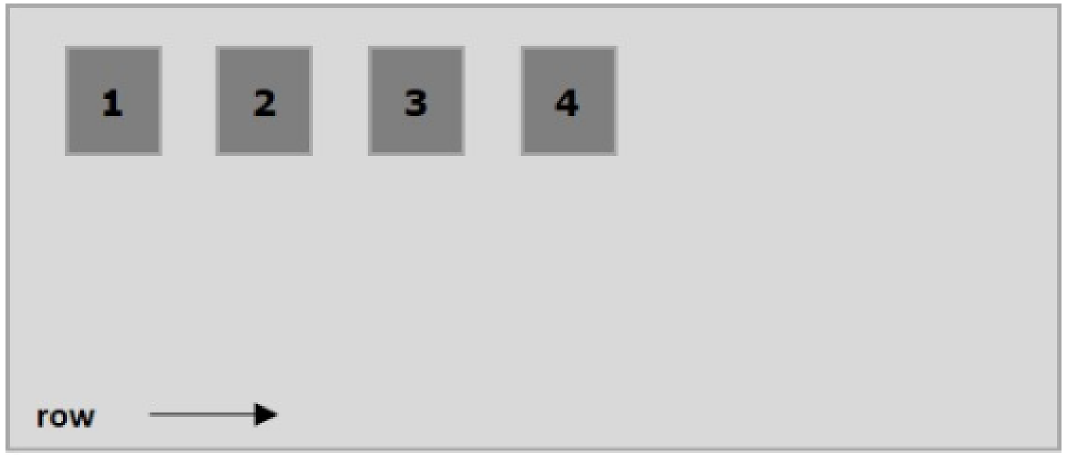
- This property accepts four values:
  - ***row*** − Arranges the elements of the container horizontally from left to right.
  - ***row-reverse*** − Arranges the elements of the container horizontally from right to left.
  - ***column*** − Arranges the elements of the container vertically from left to right.
  - ***column-reverse*** − Arranges the elements of the container vertically from right to left.

CSS Flexbox Flex-direction

- Now let's look in to few examples to demonstrate the use of the direction property.

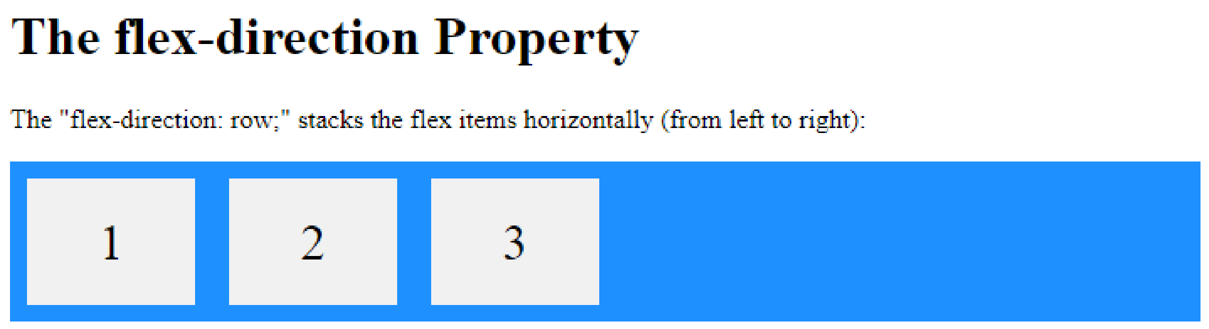
Row

- On passing ***row*** value to the direction property, the elements of the container are arranged ***horizontally*** from ***left to right*** as shown below.



CSS Flexbox Flex-direction

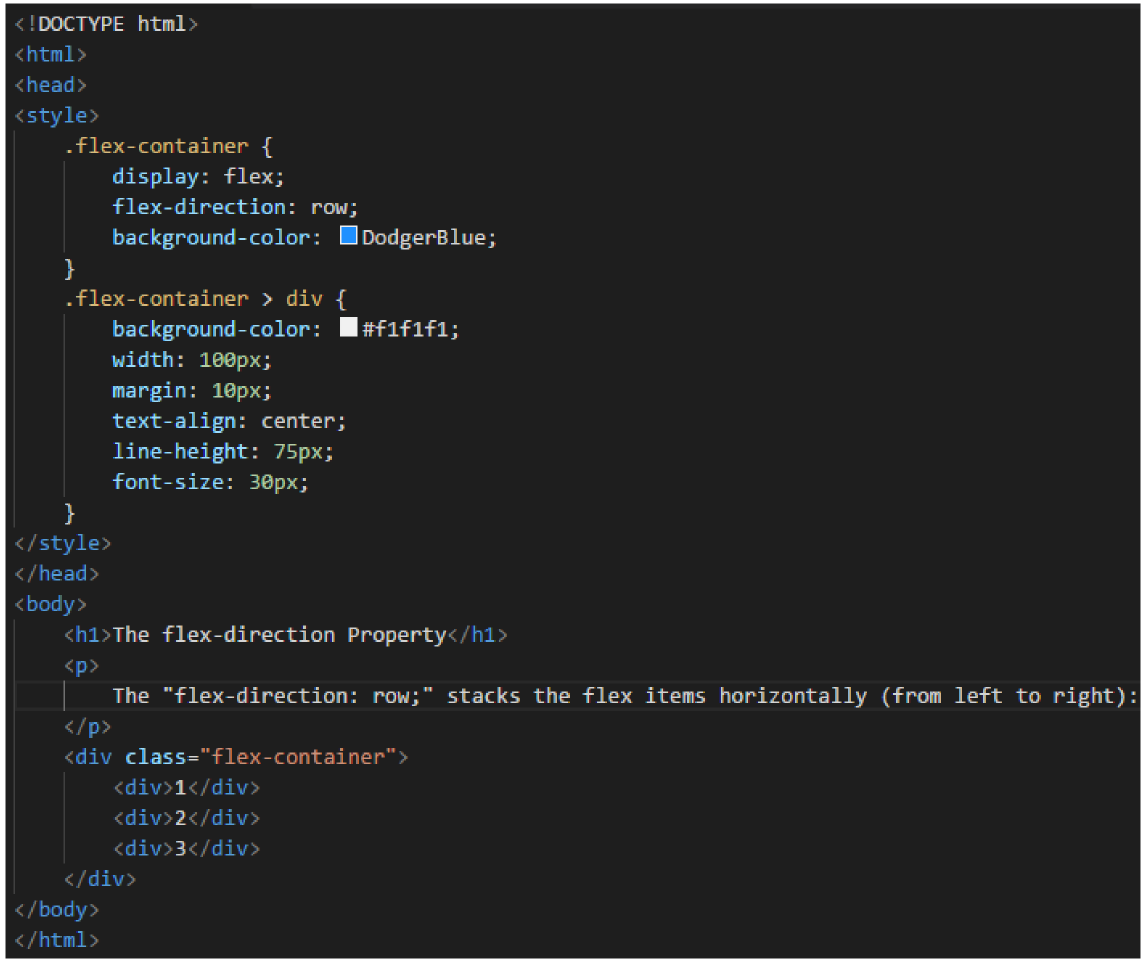
- The example demonstrates the result of passing the value row to the flex-direction property.
- Here, we are creating 3 boxes with the flex-direction value ***row***.



The flex-direction Property

The "flex-direction: row;" stacks the flex items horizontally (from left to right):

1 2 3

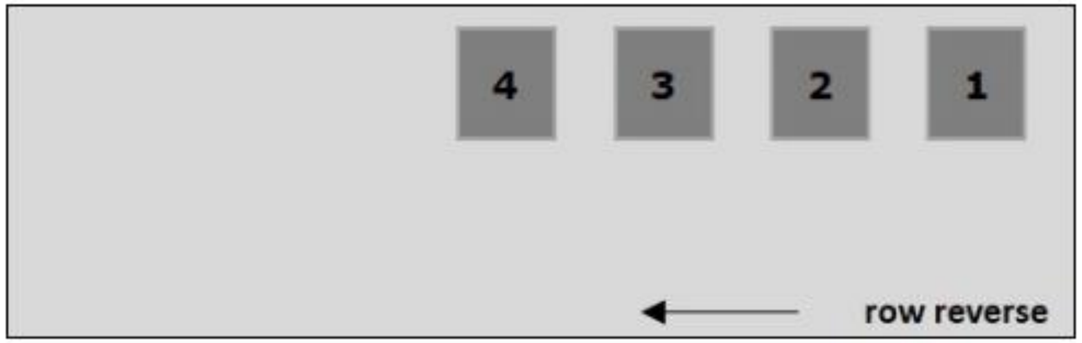


```html
<!DOCTYPE html>
<html>
<head>
<style>
    .flex-container {
        display: flex;
        flex-direction: row;
        background-color: DodgerBlue;
    }
    .flex-container > div {
        background-color: #f1f1f1;
        width: 100px;
        margin: 10px;
        text-align: center;
        line-height: 75px;
        font-size: 30px;
    }
</style>
</head>
<body>
    <h1>The flex-direction Property</h1>
    <p>
        The "flex-direction: row;" stacks the flex items horizontally (from left to right):
    </p>
    <div class="flex-container">
        <div>1</div>
        <div>2</div>
        <div>3</div>
    </div>
</body>
</html>
```

CSS Flexbox Flex-direction

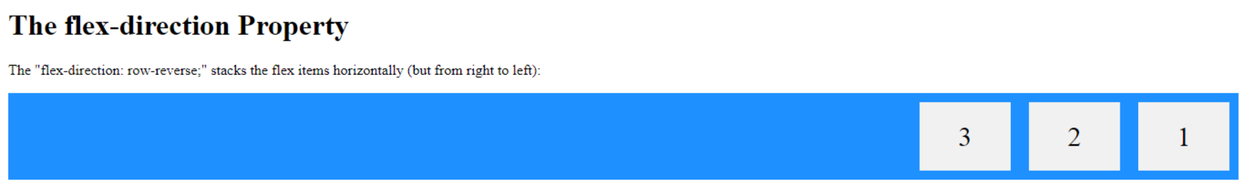
Row-reverse

- On passing this ***row-reverse*** to the direction property, the elements of the container are arranged ***horizontally*** from ***right to left*** as shown below.



CSS Flexbox Flex-direction

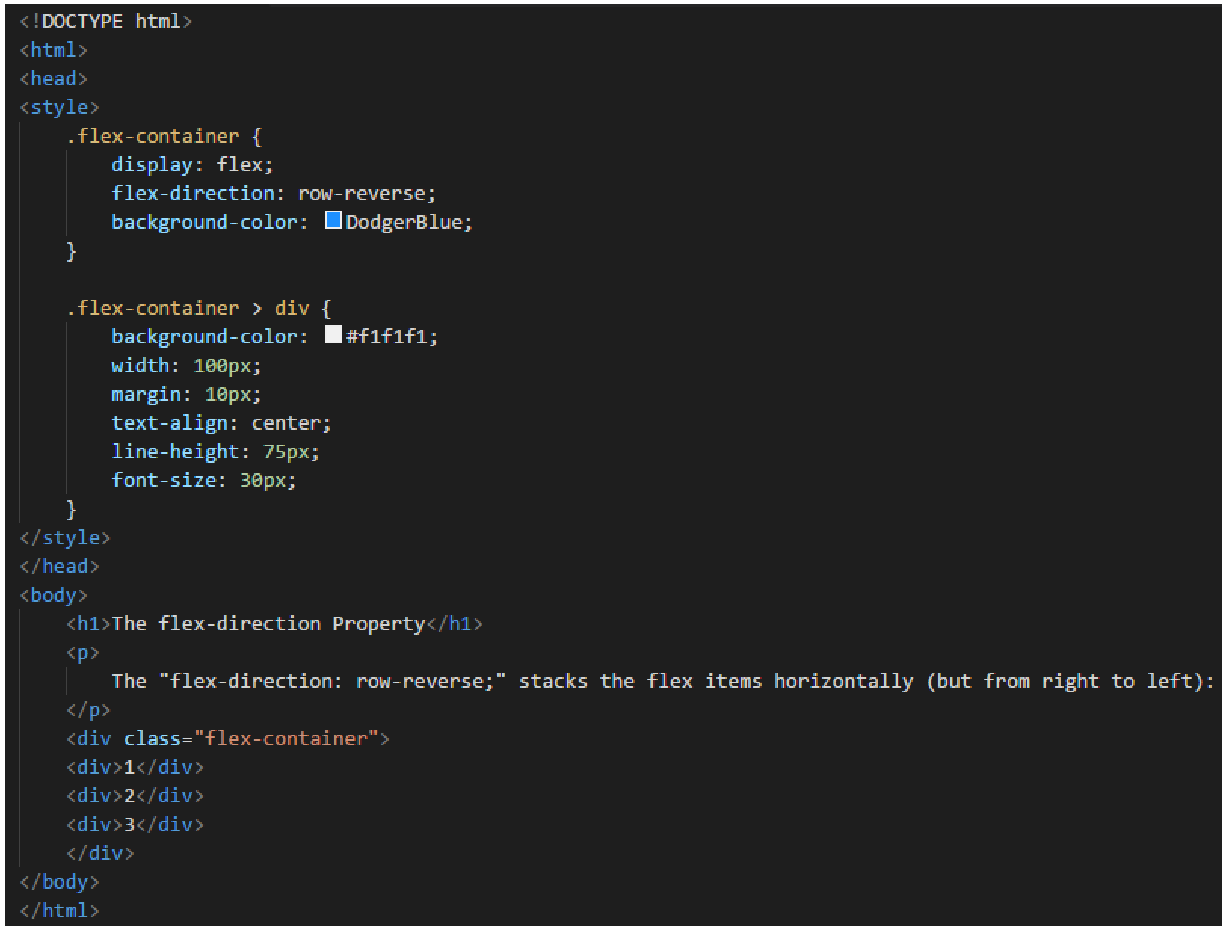
- The example demonstrates the result of passing the value ***row-reverse*** to the ***flex-direction*** property.
- Here, we are creating 3 boxes with the flex-direction value ***row-reverse***.



The flex-direction Property

The "flex-direction: row-reverse;" stacks the flex items horizontally (but from right to left):

3 2 1

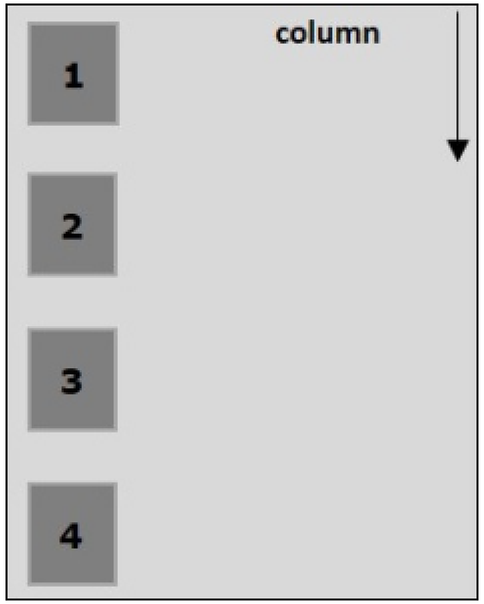


```
<!DOCTYPE html>
<html>
<head>
<style>
    .flex-container {
        display: flex;
        flex-direction: row-reverse;
        background-color: DodgerBlue;
    }

    .flex-container > div {
        background-color: #f1f1f1;
        width: 100px;
        margin: 10px;
        text-align: center;
        line-height: 75px;
        font-size: 30px;
    }
</style>
</head>
<body>
    <h1>The flex-direction Property</h1>
    <p>
        The "flex-direction: row-reverse;" stacks the flex items horizontally (but from right to left):
    </p>
    <div class="flex-container">
    <div>1</div>
    <div>2</div>
    <div>3</div>
    </div>
</body>
</html>
```

CSS Flexbox Flex-direction

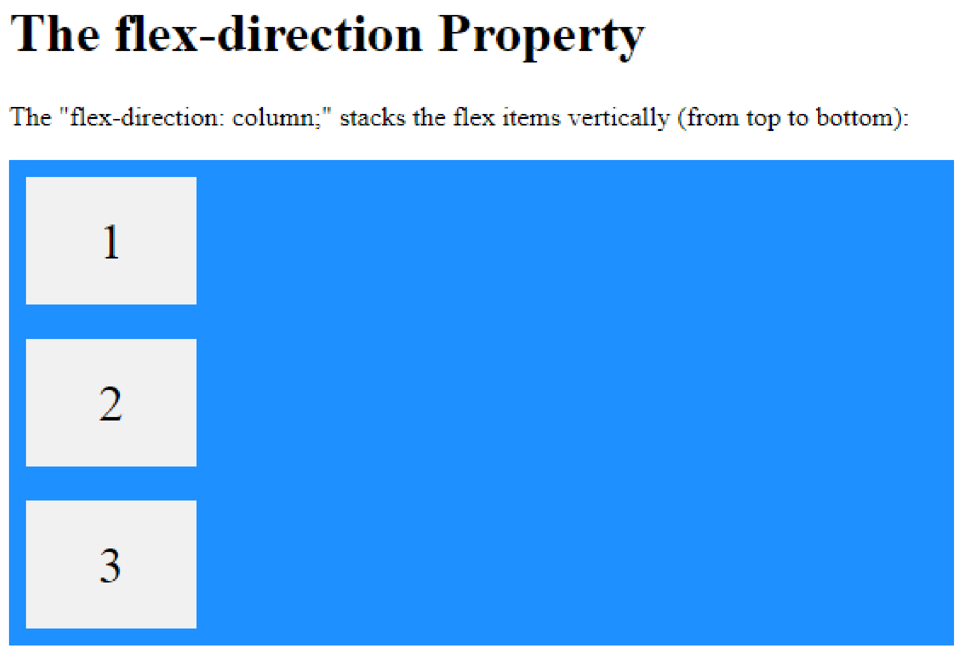
Column

- On passing ***column*** value to the direction property, the elements of the container are arranged ***vertically*** from ***top to bottom*** as shown below.



CSS Flexbox Flex-direction

- The example demonstrates the result of passing the value ***column*** to the flex-direction property.
- Here, we are creating 3 boxes with the ***flex-direction*** value ***column***.



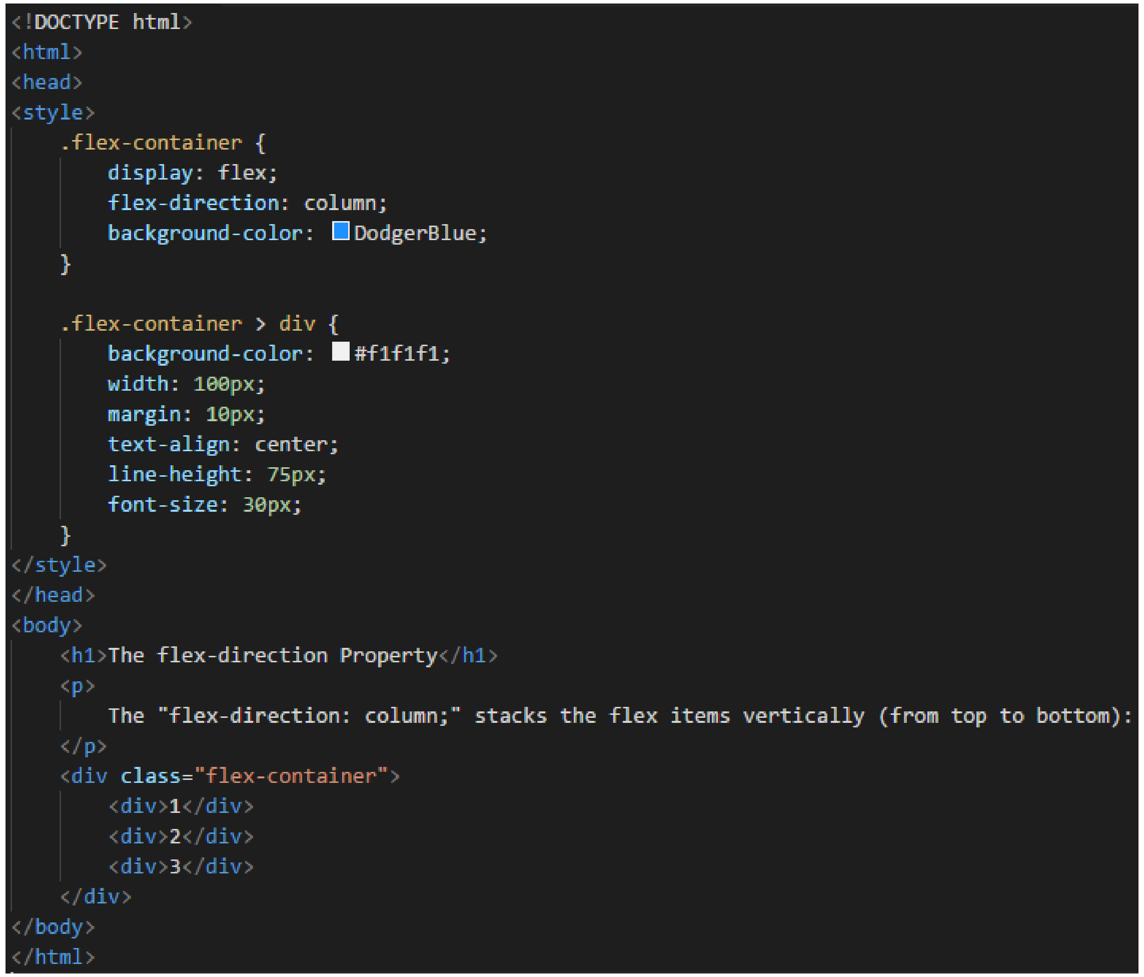
The flex-direction Property

The "flex-direction: column;" stacks the flex items vertically (from top to bottom):

1

2

3

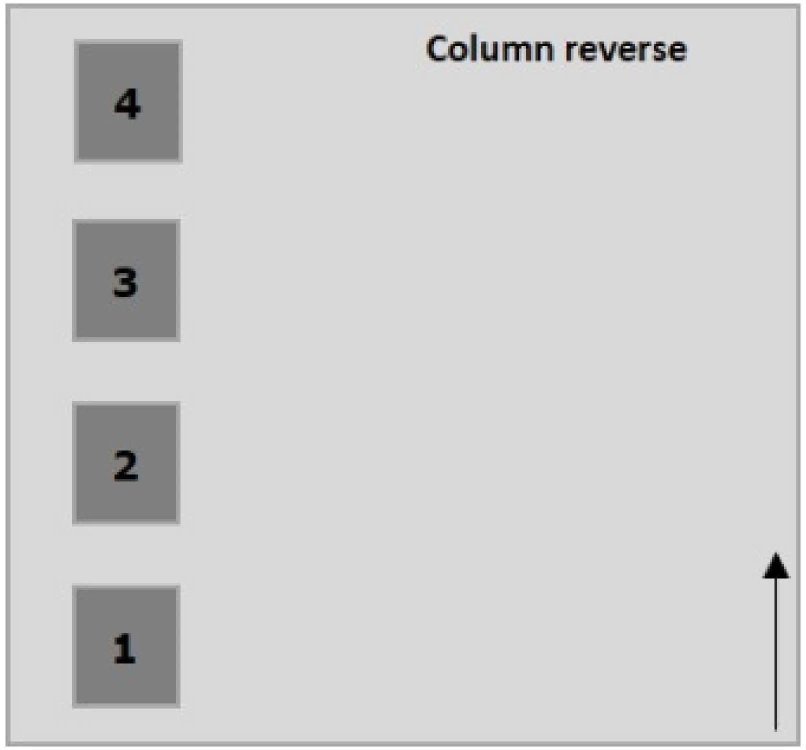


```
<!DOCTYPE html>
<html>
<head>
<style>
    .flex-container {
        display: flex;
        flex-direction: column;
        background-color: DodgerBlue;
    }

    .flex-container > div {
        background-color: #f1f1f1;
        width: 100px;
        margin: 10px;
        text-align: center;
        line-height: 75px;
        font-size: 30px;
    }
</style>
</head>
<body>
    <h1>The flex-direction Property</h1>
    <p>
        The "flex-direction: column;" stacks the flex items vertically (from top to bottom):
    </p>
    <div class="flex-container">
        <div>1</div>
        <div>2</div>
        <div>3</div>
    </div>
</body>
</html>
```

CSS Flexbox Flex-direction

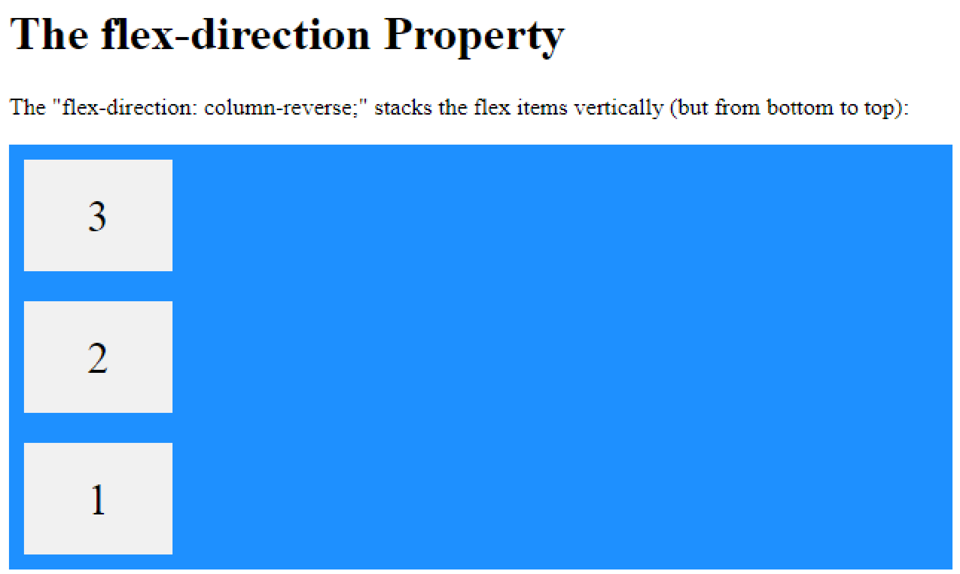
Column-reverse

- On passing ***column-reverse*** value to the direction property, the elements of the container are arranged ***vertically*** from ***bottom to top*** as shown below.



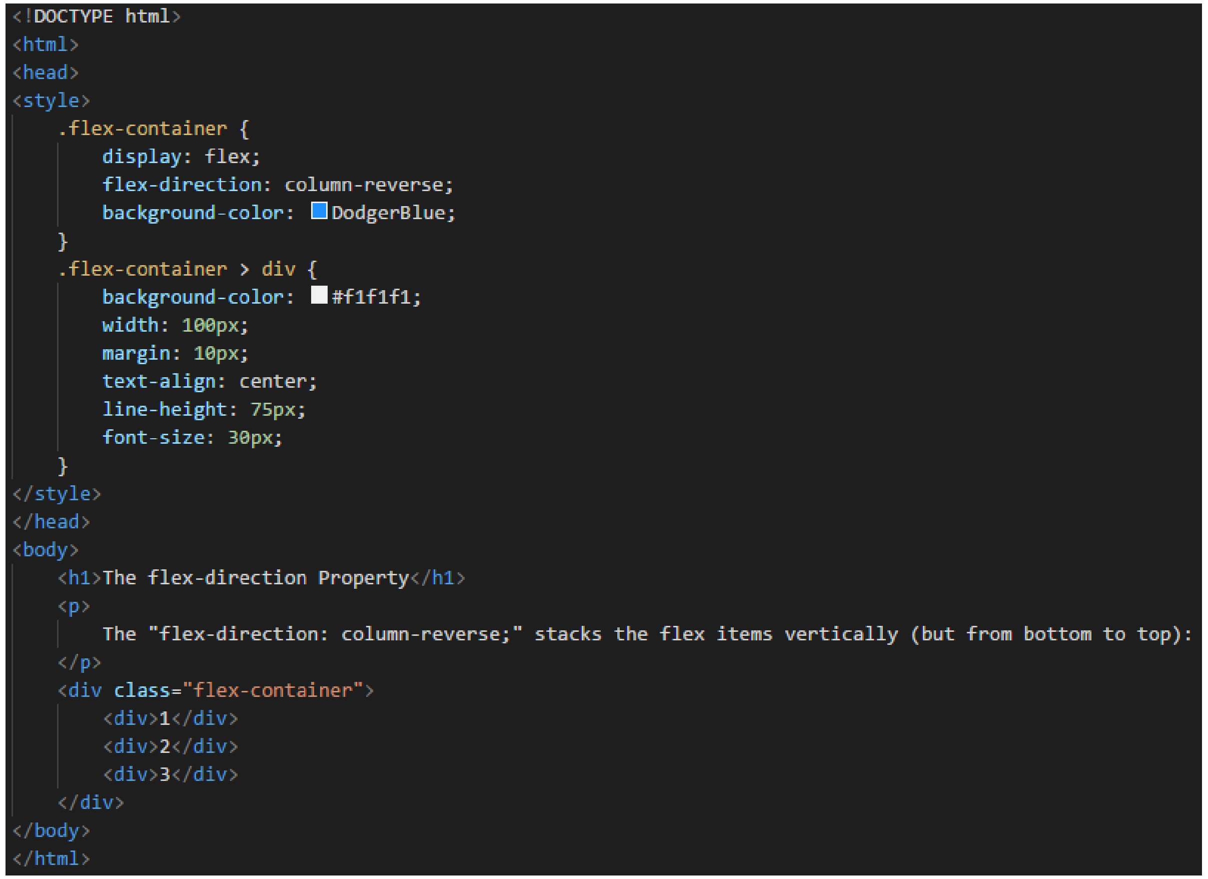
CSS Flexbox Flex-direction

- The example demonstrates the result of passing the value ***column-reverse*** to the flex-direction property.
- Here, we are creating 3 boxes with the ***flex-direction*** value ***column-reverse***.



The flex-direction Property

The "flex-direction: column-reverse;" stacks the flex items vertically (but from bottom to top):



```
<!DOCTYPE html>
<html>
<head>
<style>
    .flex-container {
        display: flex;
        flex-direction: column-reverse;
        background-color: DodgerBlue;
    }
    .flex-container > div {
        background-color: #f1f1f1;
        width: 100px;
        margin: 10px;
        text-align: center;
        line-height: 75px;
        font-size: 30px;
    }
</style>
</head>
<body>
    <h1>The flex-direction Property</h1>
    <p>
        The "flex-direction: column-reverse;" stacks the flex items vertically (but from bottom to top):
    </p>
    <div class="flex-container">
        <div>1</div>
        <div>2</div>
        <div>3</div>
    </div>
</body>
</html>
```

CSS Flexbox Flex-wrap

- The ***flex-wrap*** property specifies whether the flex items should ***wrap or not***.
- Syntax:

  flex-wrap: nowrap | wrap | wrap-reverse

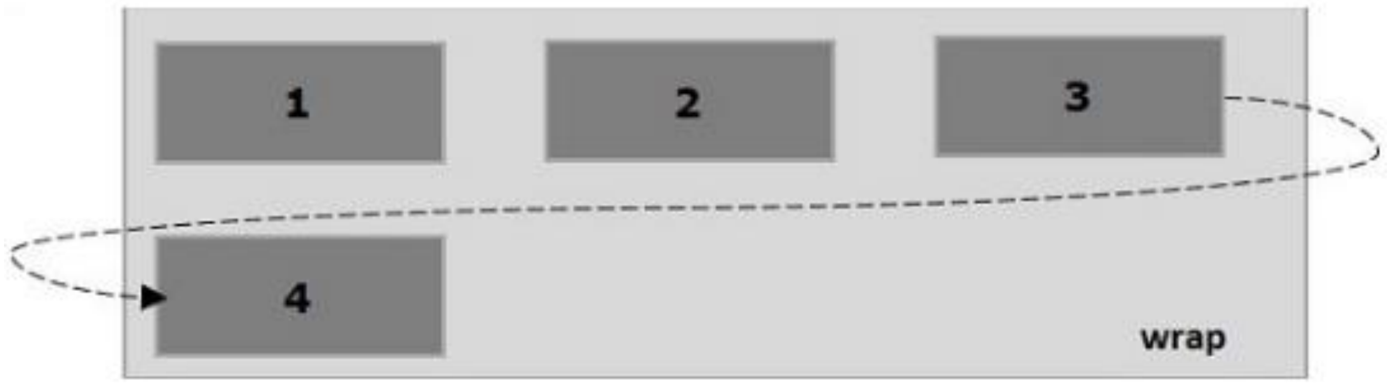
- This property accepts the following values:
  - nowrap - value specifies that the flex items will not wrap (this is default).
  - wrap – In case of insufficient space for them, the elements of the container (flex-items) will wrap into additional flex lines from top to bottom.
  - wrap-reverse – In case of insufficient space for them, the elements of the container (flex-items) will wrap into additional flex lines from bottom to top.

CSS Flexbox Flex-wrap cont.

- Now, Let's see how to use the ***wrap*** property, with examples.

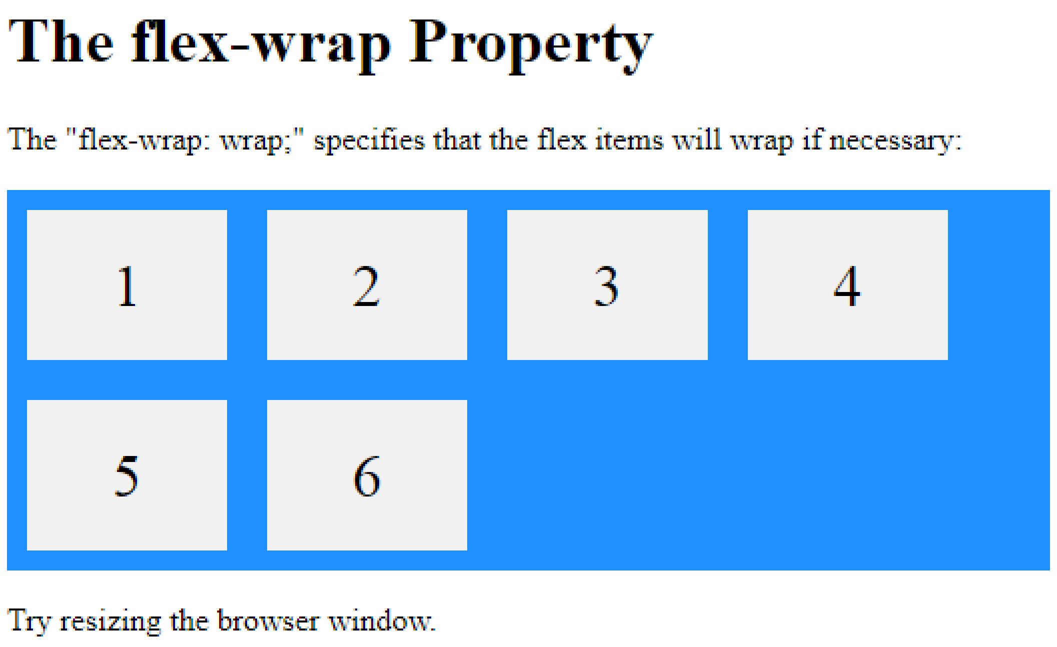
Wrap

- On passing the value **wrap** to the property ***flex-wrap***, the elements of the container are arranged ***horizontally*** from ***left to right*** as shown below.



CSS Flexbox Flex-wrap cont.

- The example demonstrates the result of passing the value ***wrap*** to the ***flex-wrap*** property.
- Here, we are creating 6 boxes with the ***flex-direction*** value ***row***.

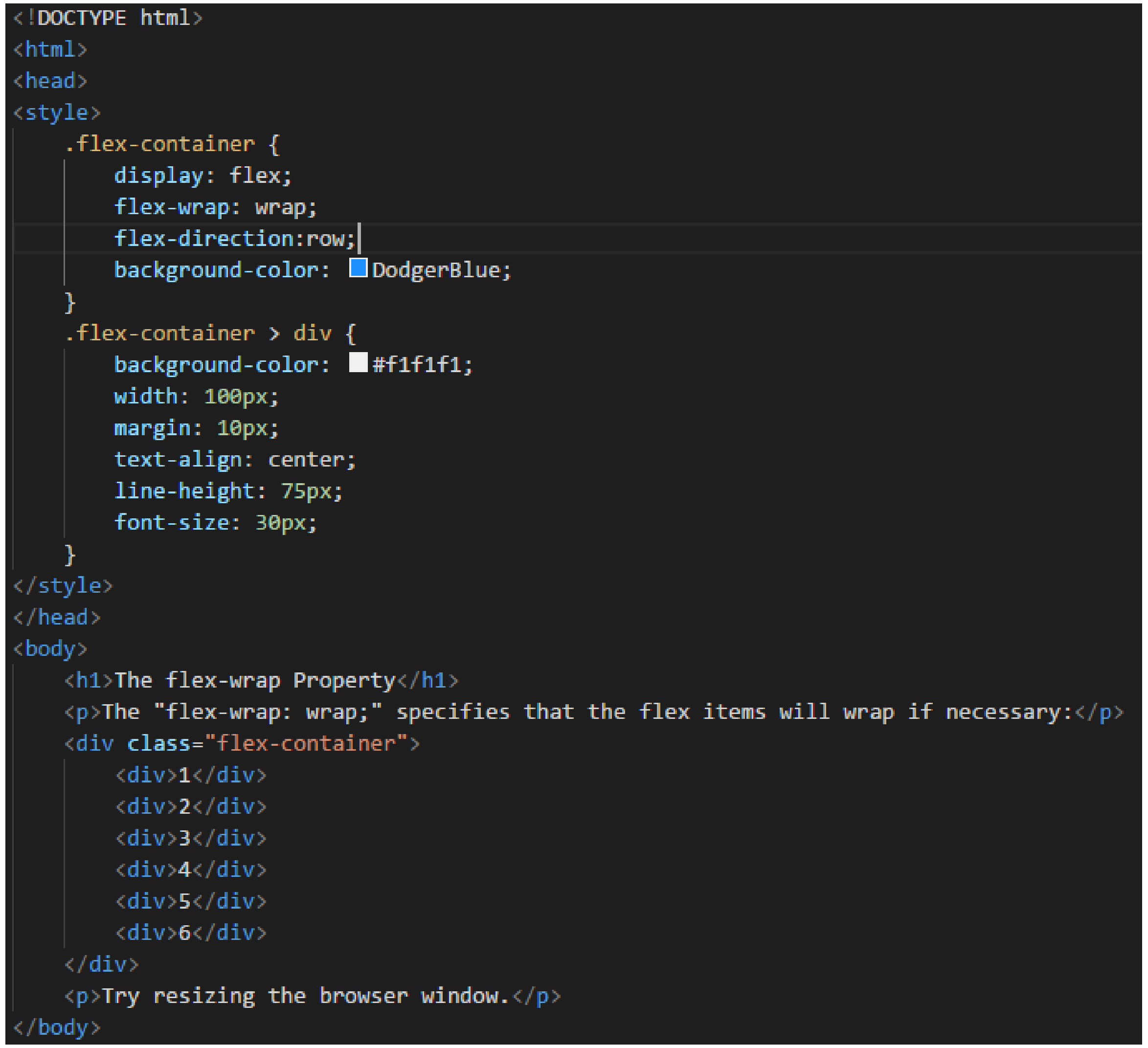


The flex-wrap Property

The "flex-wrap: wrap;" specifies that the flex items will wrap if necessary:

1 2 3 4
5 6

Try resizing the browser window.

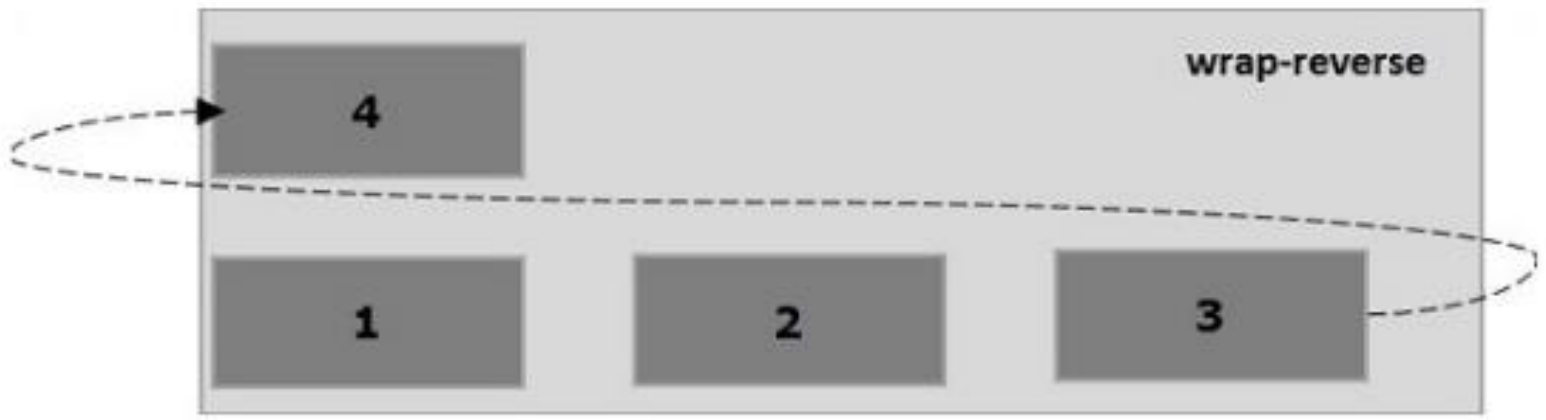


```
<!DOCTYPE html>
<html>
<head>
<style>
    .flex-container {
        display: flex;
        flex-wrap: wrap;
        flex-direction:row;
        background-color: 🟦DodgerBlue;
    }
    .flex-container > div {
        background-color: ⬜#f1f1f1;
        width: 100px;
        margin: 10px;
        text-align: center;
        line-height: 75px;
        font-size: 30px;
    }
</style>
</head>
<body>
    <h1>The flex-wrap Property</h1>
    <p>The "flex-wrap: wrap;" specifies that the flex items will wrap if necessary:</p>
    <div class="flex-container">
        <div>1</div>
        <div>2</div>
        <div>3</div>
        <div>4</div>
        <div>5</div>
        <div>6</div>
    </div>
    <p>Try resizing the browser window.</p>
</body>
```

CSS Flexbox Flex-wrap cont.

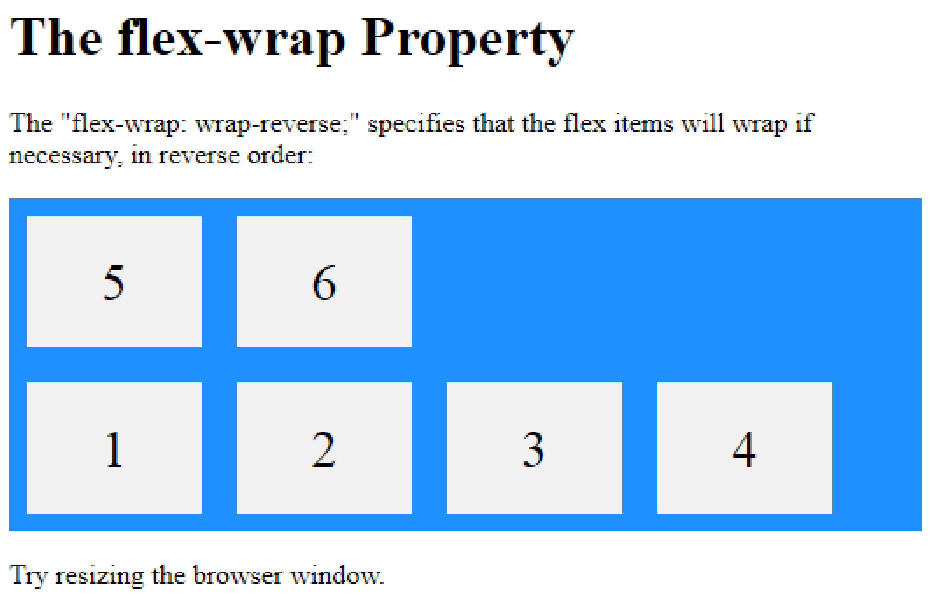
Wrap-reverse

- On passing the value ***wrap-reverse*** to the property ***flex-wrap***, the elements of the container are arranged ***horizontally*** from ***left to right*** as shown below.



CSS Flexbox Flex-wrap cont.

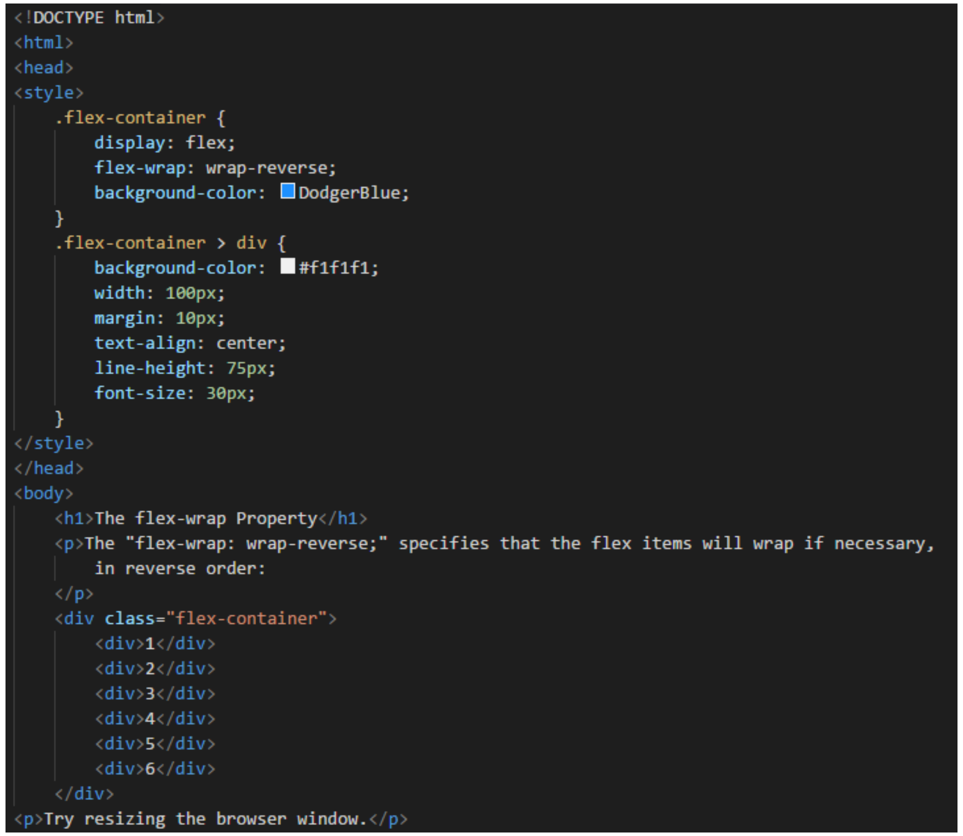
- The example demonstrates the result of passing the value **wrap-reverse** to the ***flex-wrap*** property.
- Here, we are creating 6 boxes with the ***flex-direction*** value ***row***.



The flex-wrap Property

The "flex-wrap: wrap-reverse;" specifies that the flex items will wrap if necessary, in reverse order:

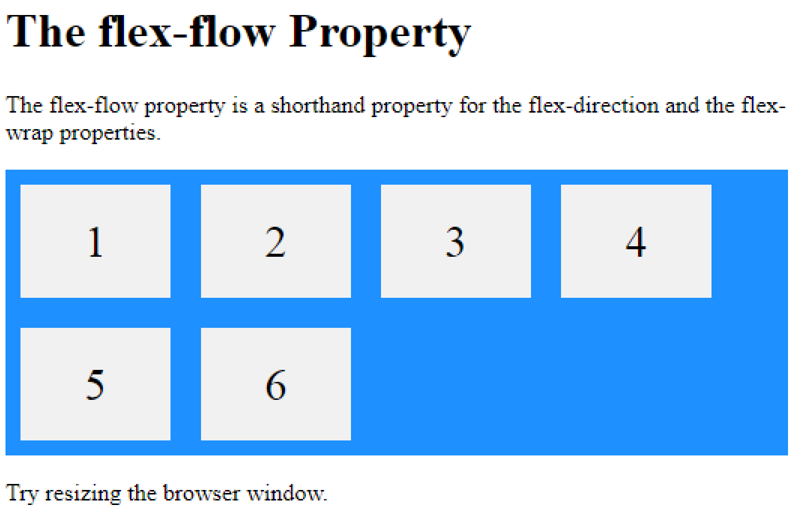
Try resizing the browser window.



```
<!DOCTYPE html>
<html>
<head>
<style>
    .flex-container {
        display: flex;
        flex-wrap: wrap-reverse;
        background-color: DodgerBlue;
    }
    .flex-container > div {
        background-color: #f1f1f1;
        width: 100px;
        margin: 10px;
        text-align: center;
        line-height: 75px;
        font-size: 30px;
    }
</style>
</head>
<body>
    <h1>The flex-wrap Property</h1>
    <p>The "flex-wrap: wrap-reverse;" specifies that the flex items will wrap if necessary,
        in reverse order:
    </p>
    <div class="flex-container">
        <div>1</div>
        <div>2</div>
        <div>3</div>
        <div>4</div>
        <div>5</div>
        <div>6</div>
    </div>
<p>Try resizing the browser window.</p>
```

CSS Flexbox Flex-flow

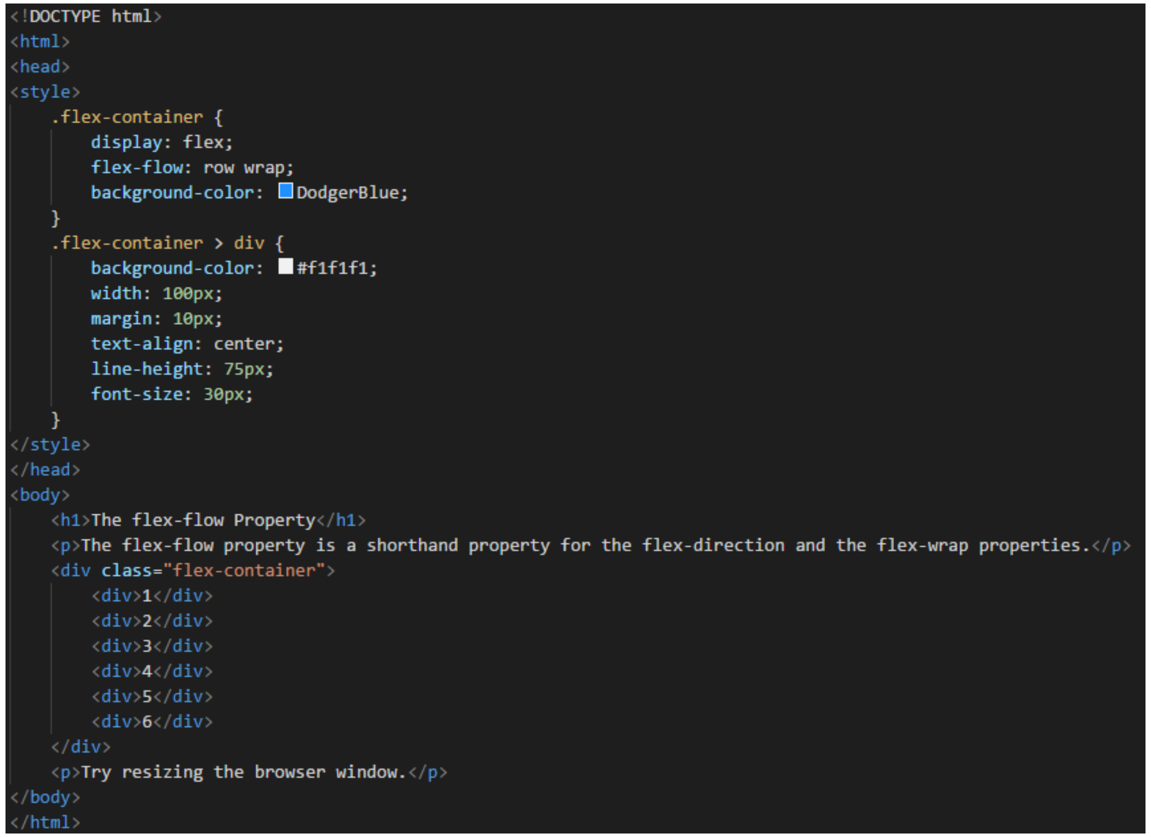
- The ***flex-flow*** property is a shorthand property for setting both the ***flex-direction*** and ***flex-wrap*** properties.
- Let's look in to an example, which demonstrate setting both the ***flex-direction*** and ***flex-wrap*** properties using ***flex-flow*** property.



The flex-flow Property

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

Try resizing the browser window.



```
<!DOCTYPE html>
<html>
<head>
<style>
    .flex-container {
        display: flex;
        flex-flow: row wrap;
        background-color: DodgerBlue;
    }
    .flex-container > div {
        background-color: #f1f1f1;
        width: 100px;
        margin: 10px;
        text-align: center;
        line-height: 75px;
        font-size: 30px;
    }
</style>
</head>
<body>
    <h1>The flex-flow Property</h1>
    <p>The flex-flow property is a shorthand property for the flex-direction and the flex-wrap properties.</p>
    <div class="flex-container">
        <div>1</div>
        <div>2</div>
        <div>3</div>
        <div>4</div>
        <div>5</div>
        <div>6</div>
    </div>
    <p>Try resizing the browser window.</p>
</body>
</html>
```

CSS Flexbox Justifying Contents

- Using the property ***justify-content***, you can align the contents along the main axis by distributing the extra space as intended.
- Syntax is:

justify-content: flex-start | flex-end | center | space-between | space-around;

This property accepts the following values:

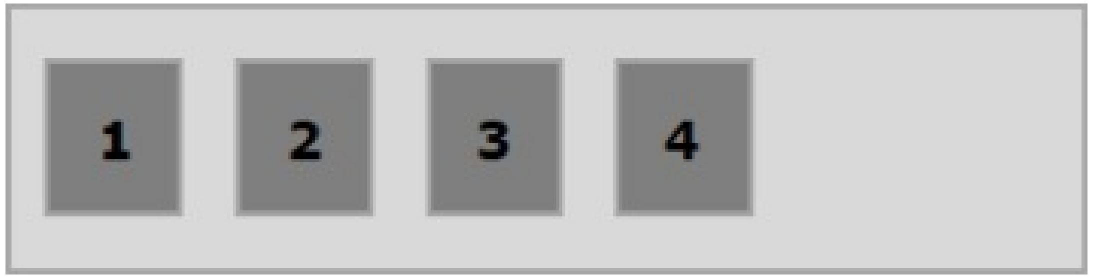
- ***flex-start*** − The flex-items are placed at the start of the container.
- ***flex-end*** − The flex-items are placed at the end of the container.
- ***center*** − The flex-items are placed at the center of the container, where the extra space is equally distributed at the start and at the end of the flex-items.
- ***space-between*** − The extra space is equally distributed between the flex-items.
- ***space-around*** − The extra space is equally distributed between the flex items such that the space between the edges of the container and its contents is half as the space between the flex-items.

CSS Flexbox Justifying Contents

- Now, we will see how to use the justify-content property, with examples.

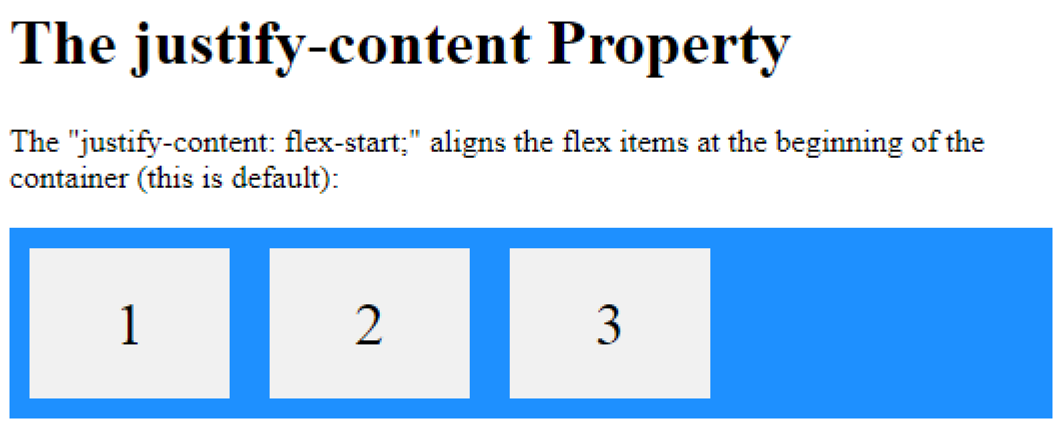
Flex-start

- On passing ***flex-start*** value to the property ***justify-content***, the flex-items are placed at the ***start of the container*** like shown below.



CSS Flexbox Justifying Contents

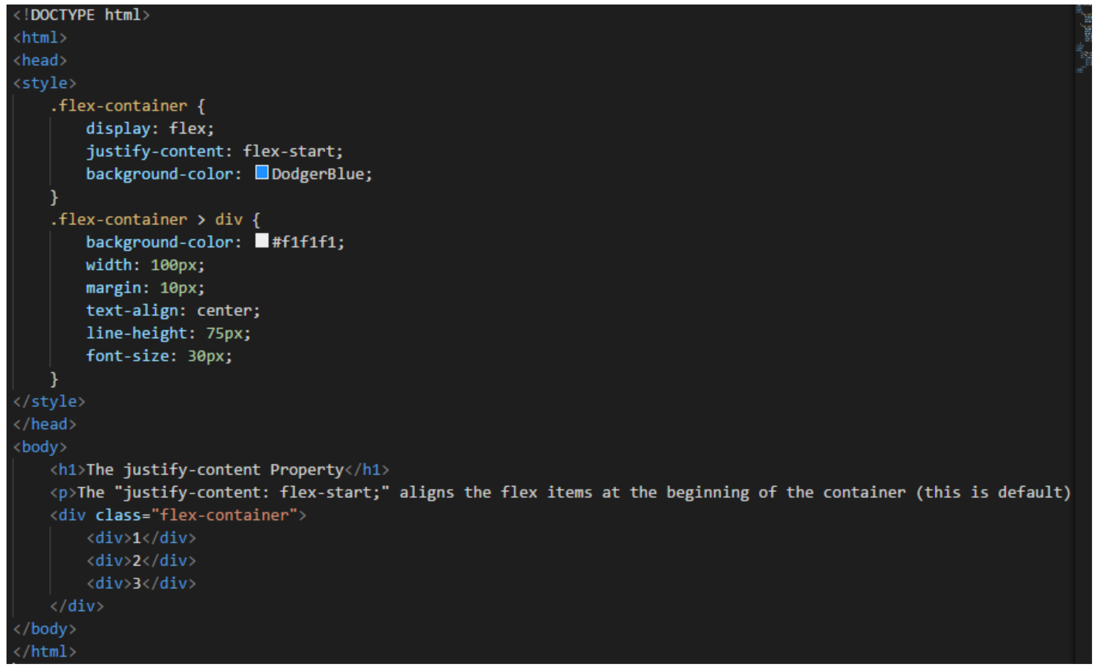
- The example demonstrates the result of passing the value ***flex-start*** to the ***justify-content*** property.



The justify-content Property

The "justify-content: flex-start;" aligns the flex items at the beginning of the container (this is default):

1 2 3

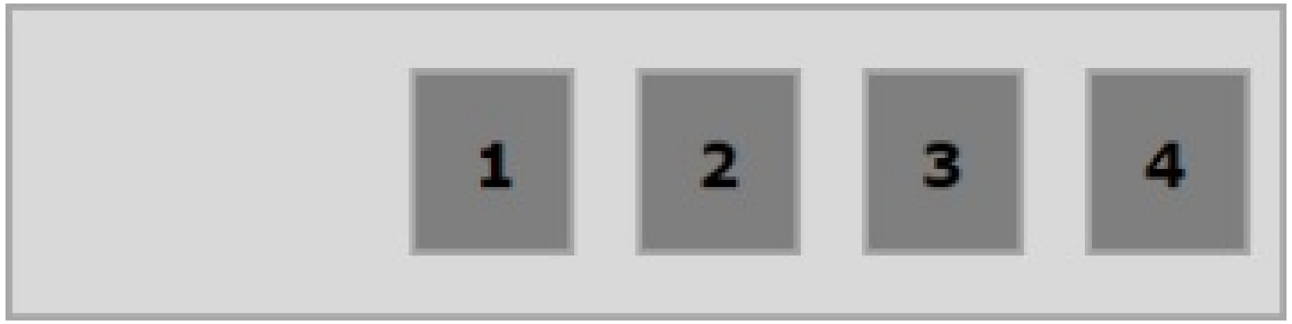


```
<!DOCTYPE html>
<html>
<head>
<style>
    .flex-container {
        display: flex;
        justify-content: flex-start;
        background-color: DodgerBlue;
    }
    .flex-container > div {
        background-color: #f1f1f1;
        width: 100px;
        margin: 10px;
        text-align: center;
        line-height: 75px;
        font-size: 30px;
    }
</style>
</head>
<body>
    <h1>The justify-content Property</h1>
    <p>The "justify-content: flex-start;" aligns the flex items at the beginning of the container (this is default)
    <div class="flex-container">
        <div>1</div>
        <div>2</div>
        <div>3</div>
    </div>
</body>
</html>
```

CSS Flexbox Justifying Contents

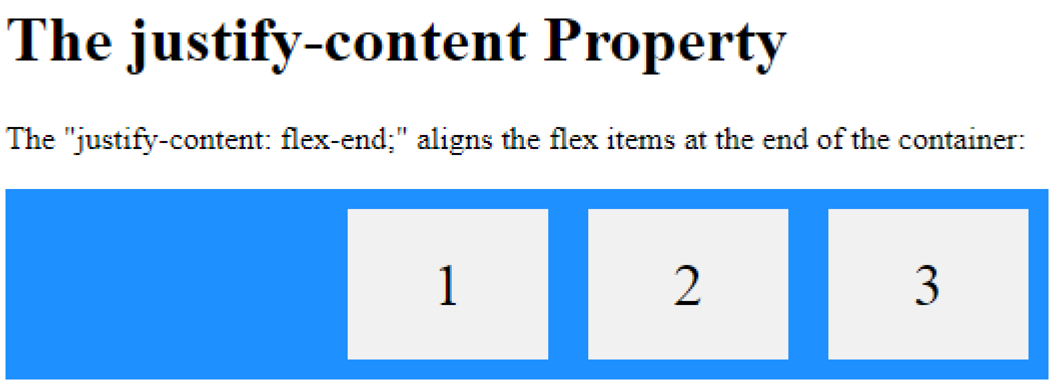
Flex-end

- On passing ***flex-end*** value to the property ***justify-content***, the ***flex-items*** are placed at the ***end of the container*** like shown below.



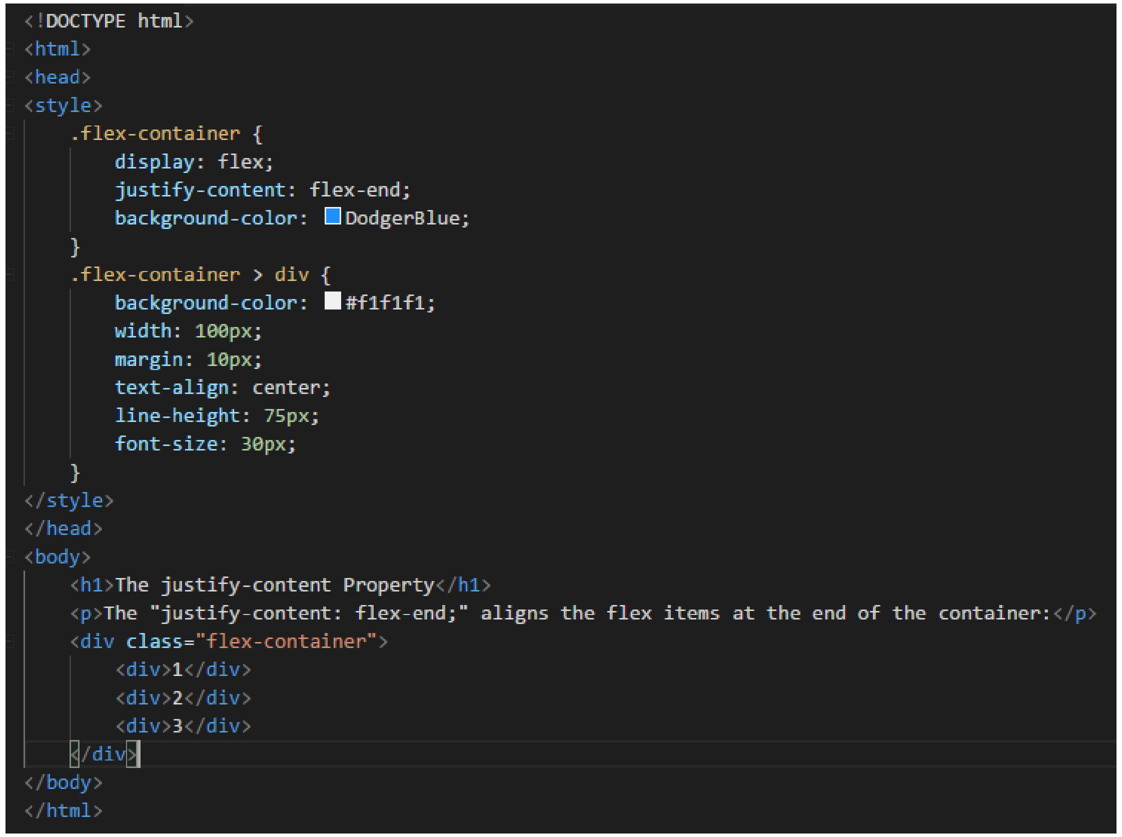
CSS Flexbox Justifying Contents

- The example demonstrates the result of passing the value ***flex-end*** to the ***justify-content*** property.



The justify-content Property

The "justify-content: flex-end;" aligns the flex items at the end of the container:

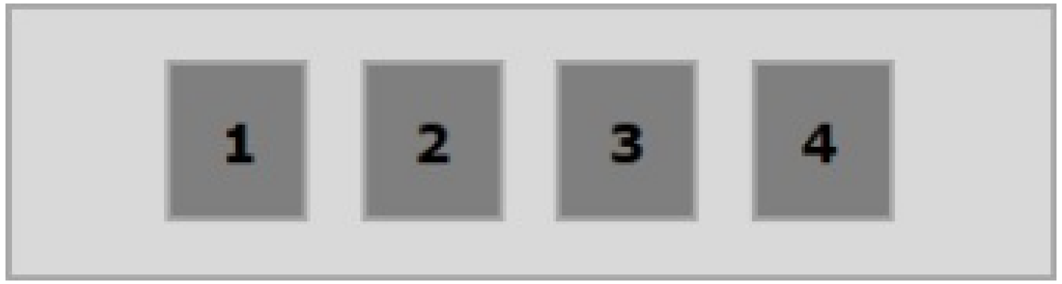


```
<!DOCTYPE html>
<html>
<head>
<style>
    .flex-container {
        display: flex;
        justify-content: flex-end;
        background-color: DodgerBlue;
    }
    .flex-container > div {
        background-color: #f1f1f1;
        width: 100px;
        margin: 10px;
        text-align: center;
        line-height: 75px;
        font-size: 30px;
    }
</style>
</head>
<body>
    <h1>The justify-content Property</h1>
    <p>The "justify-content: flex-end;" aligns the flex items at the end of the container:</p>
    <div class="flex-container">
        <div>1</div>
        <div>2</div>
        <div>3</div>
    </div>
</body>
</html>
```

CSS Flexbox Justifying Contents

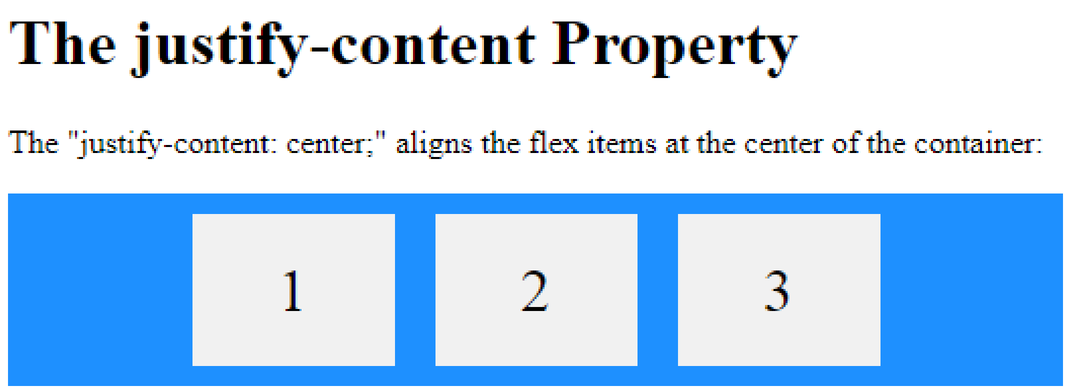
Center

- On passing ***center*** value to the property ***justify-content***, the flex-items are placed at the ***center of the container***, where the extra space is equally distributed at the start and at the end of the flex-items.



CSS Flexbox Justifying Contents

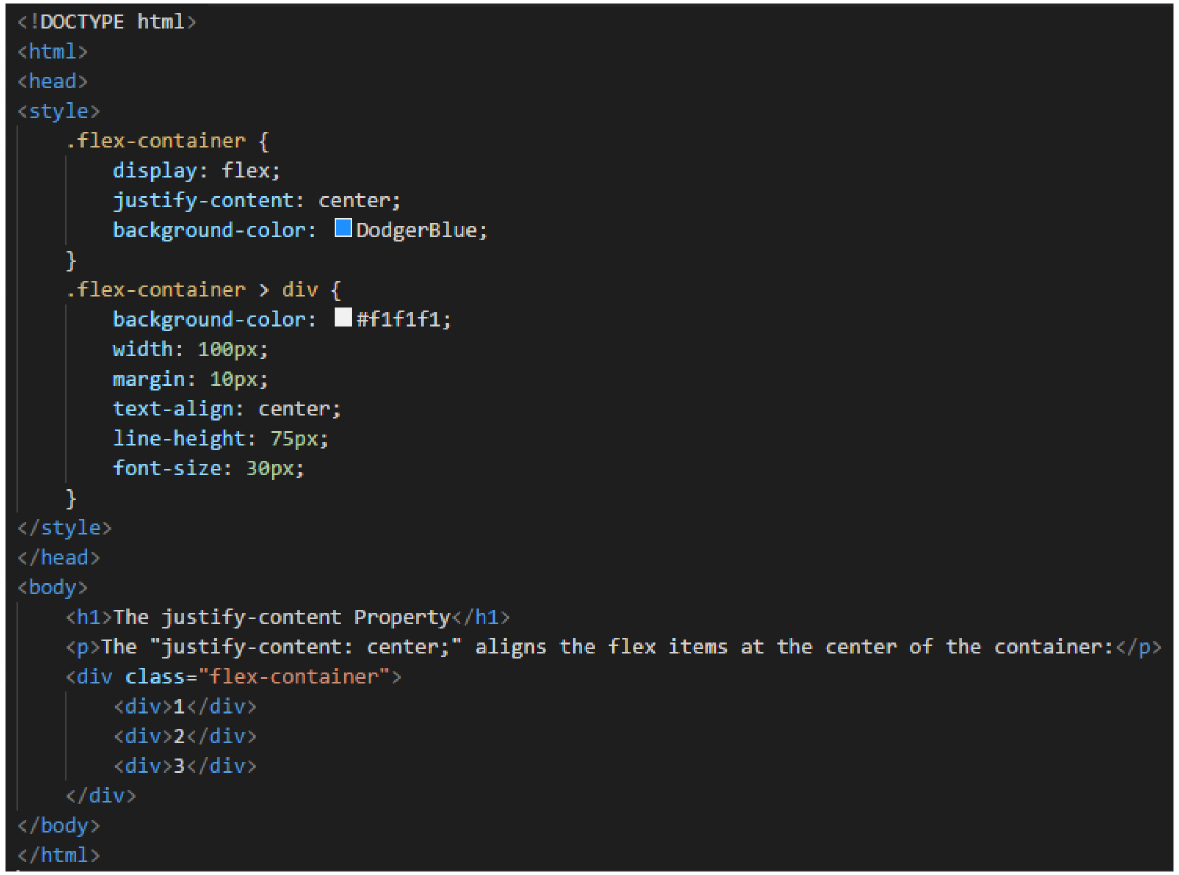
- The example demonstrates the result of passing the value ***center*** to the ***justify-content*** property.



The justify-content Property

The "justify-content: center;" aligns the flex items at the center of the container:

```
1  2  3
```

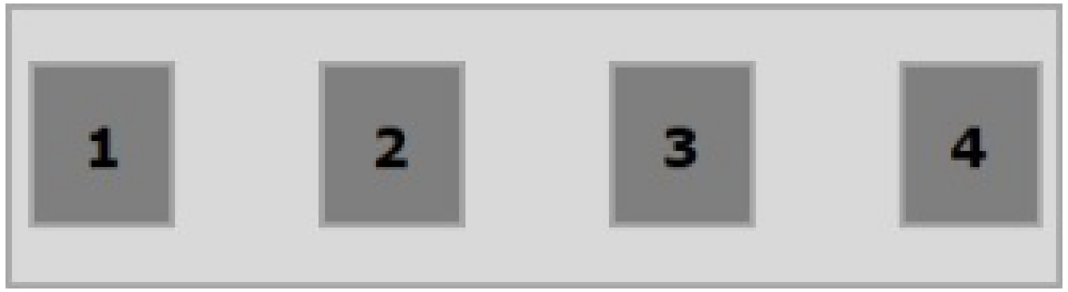


```
<!DOCTYPE html>
<html>
<head>
<style>
    .flex-container {
        display: flex;
        justify-content: center;
        background-color: DodgerBlue;
    }
    .flex-container > div {
        background-color: #f1f1f1;
        width: 100px;
        margin: 10px;
        text-align: center;
        line-height: 75px;
        font-size: 30px;
    }
</style>
</head>
<body>
    <h1>The justify-content Property</h1>
    <p>The "justify-content: center;" aligns the flex items at the center of the container:</p>
    <div class="flex-container">
        <div>1</div>
        <div>2</div>
        <div>3</div>
    </div>
</body>
</html>
```

CSS Flexbox Justifying Contents

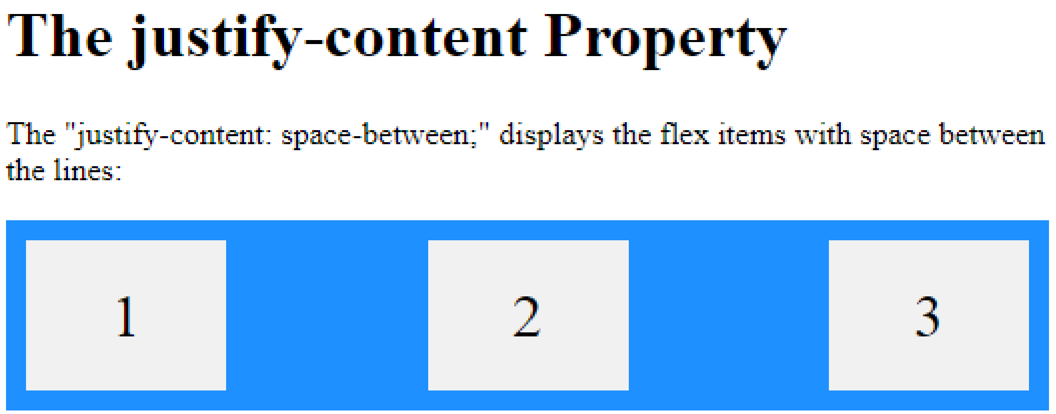
Space-between

- On passing ***space-between*** value to the property ***justify-content***, the extra space is equally distributed between the flex items
- Also the space between any two flex-items is the same and the start and end of the flex-items touch the edges of the container.



CSS Flexbox Justifying Contents

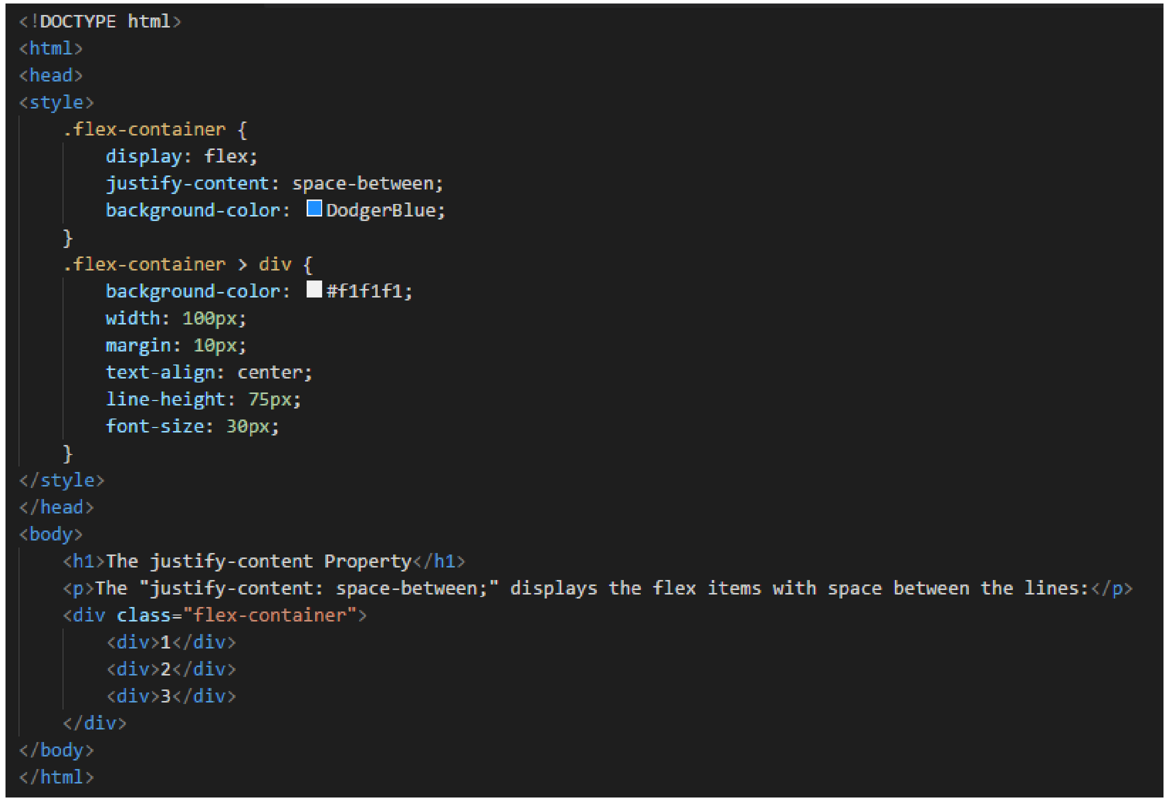
- The example demonstrates the result of passing the value ***space-between*** to the ***justify-content*** property.



The justify-content Property

The "justify-content: space-between;" displays the flex items with space between the lines:

1 2 3

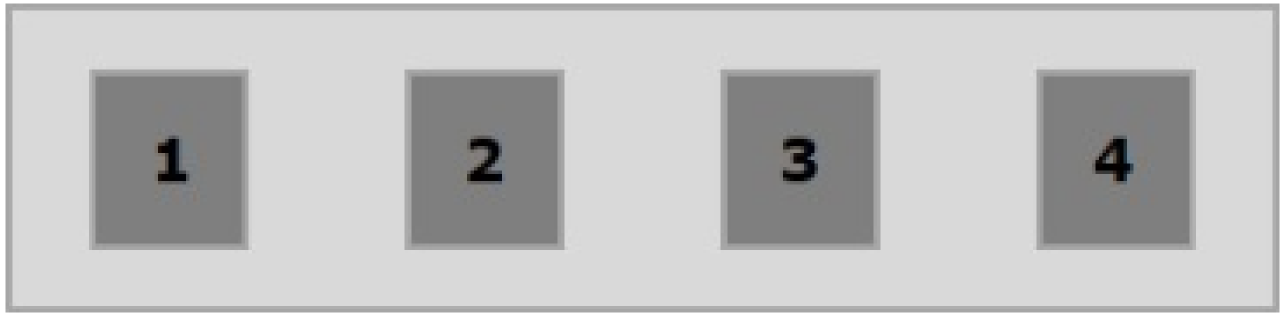


```
<!DOCTYPE html>
<html>
<head>
<style>
    .flex-container {
        display: flex;
        justify-content: space-between;
        background-color: DodgerBlue;
    }
    .flex-container > div {
        background-color: #f1f1f1;
        width: 100px;
        margin: 10px;
        text-align: center;
        line-height: 75px;
        font-size: 30px;
    }
</style>
</head>
<body>
    <h1>The justify-content Property</h1>
    <p>The "justify-content: space-between;" displays the flex items with space between the lines:</p>
    <div class="flex-container">
        <div>1</div>
        <div>2</div>
        <div>3</div>
    </div>
</body>
</html>
```

CSS Flexbox Justifying Contents

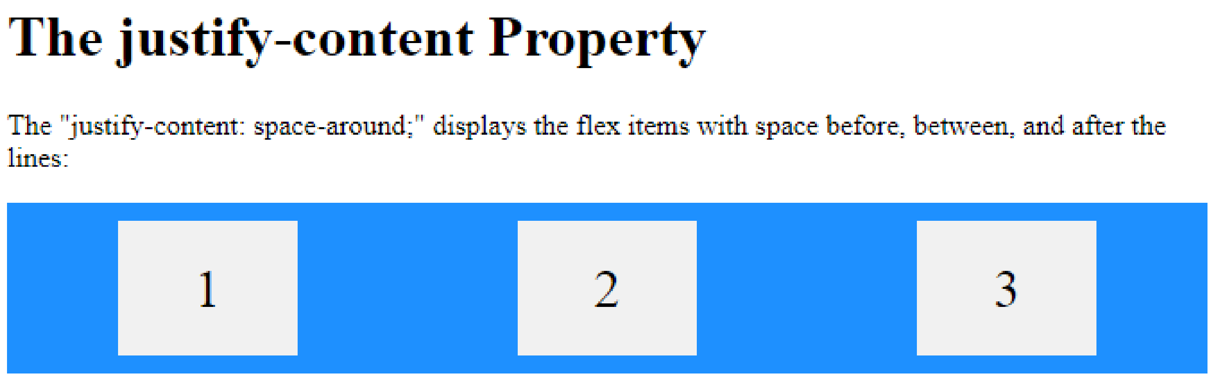
Space-around

- On passing **space-around** value to the property ***justify-content***, the extra space is equally distributed between the flex-items such that the space between any two flex-items is the same.
- However, the space between the edges of the container and its contents (the start and end of the flex-items) is half as the space between the flex items.



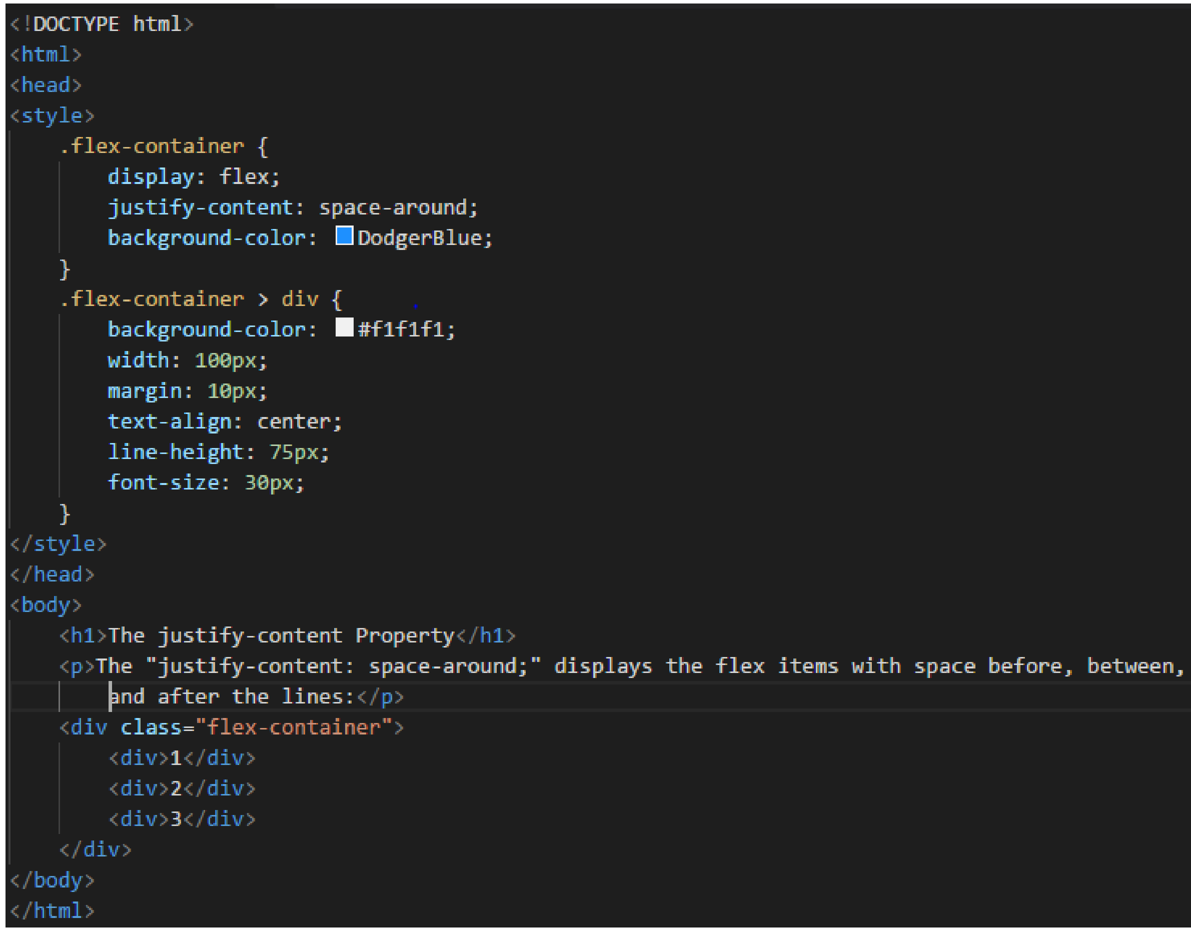
CSS Flexbox Justifying Contents

- The following example demonstrates the result of passing the value ***space-around*** to the ***justify-content property***.



The justify-content Property

The "justify-content: space-around;" displays the flex items with space before, between, and after the lines:



```
<!DOCTYPE html>
<html>
<head>
<style>
    .flex-container {
        display: flex;
        justify-content: space-around;
        background-color: DodgerBlue;
    }
    .flex-container > div {
        background-color: #f1f1f1;
        width: 100px;
        margin: 10px;
        text-align: center;
        line-height: 75px;
        font-size: 30px;
    }
</style>
</head>
<body>
    <h1>The justify-content Property</h1>
    <p>The "justify-content: space-around;" displays the flex items with space before, between,
        and after the lines:</p>
    <div class="flex-container">
        <div>1</div>
        <div>2</div>
        <div>3</div>
    </div>
</body>
</html>
```

CSS Flexbox Align Items

- The **align-items** property is same as **justify content**. But here, the items were aligned across the cross access (vertically).
- Syntax is:

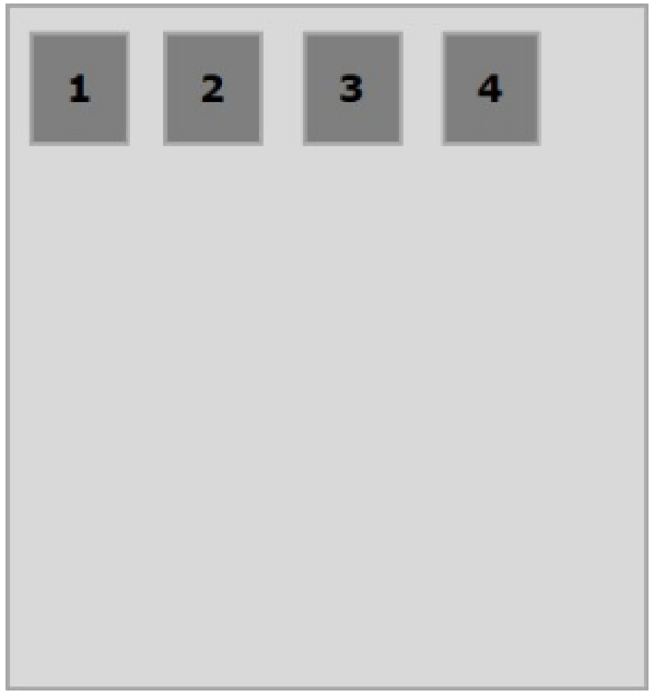
  align-items: flex-start | flex-end | center | baseline | stretch;

- This property accepts the following values:
  - ***flex-start*** − The flex items were aligned vertically at the top of the container.
  - ***flex-end*** − The flex items were aligned vertically at the bottom of the container.
  - ***flex-center*** − The flex items were aligned vertically at the center of the container.
  - ***stretch*** − The flex items were aligned vertically such that they fill up the whole vertical space of the container.
  - ***baseline*** − The flex items were aligned such that the baseline of their text align along a horizontal line.

CSS Flexbox Align Items

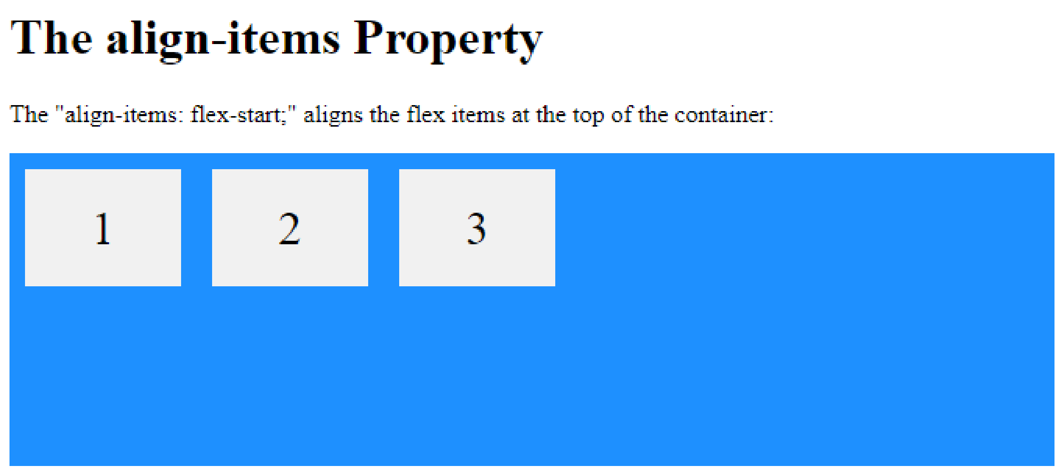
flex-start

- On passing ***flex-start*** value to the property ***align-items***, the flex items were aligned ***vertically*** at the ***top of the container***.



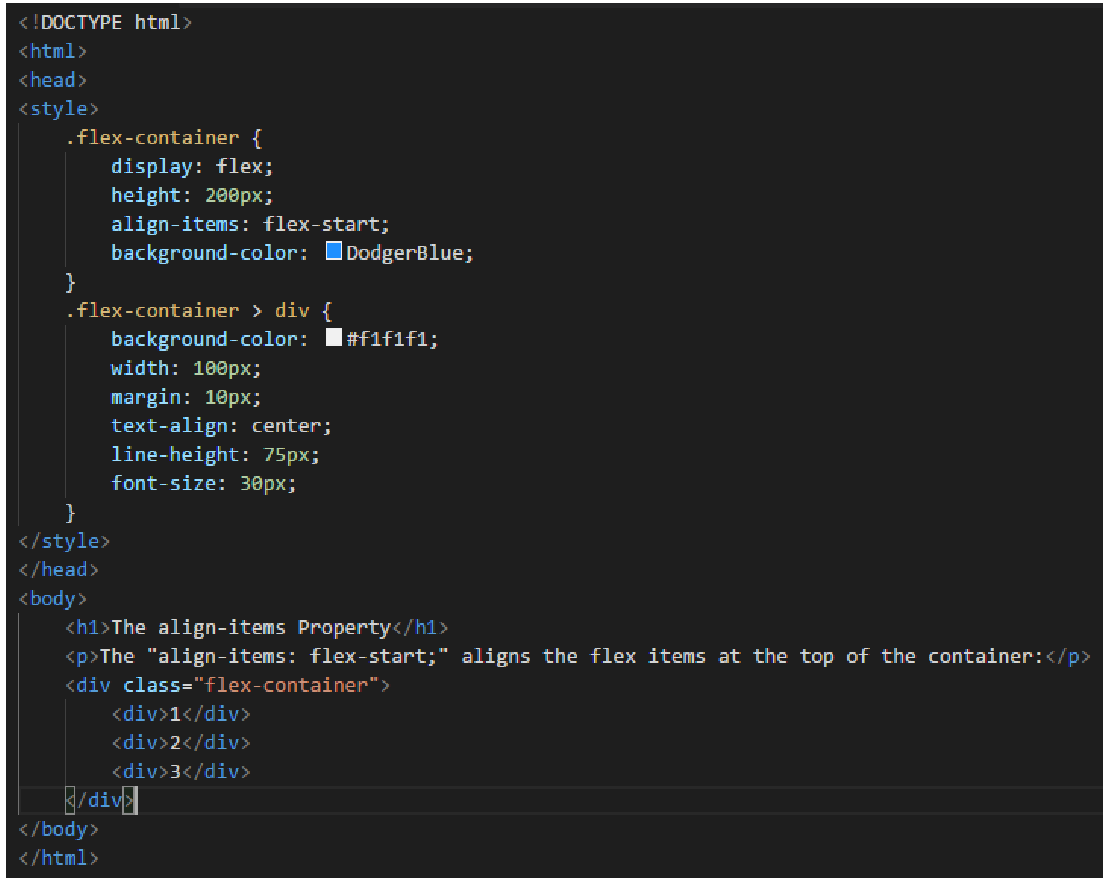
CSS Flexbox Align Items

The example demonstrates the result of passing the value ***flex-start*** to the ***align-items*** property.



The align-items Property

The "align-items: flex-start;" aligns the flex items at the top of the container:

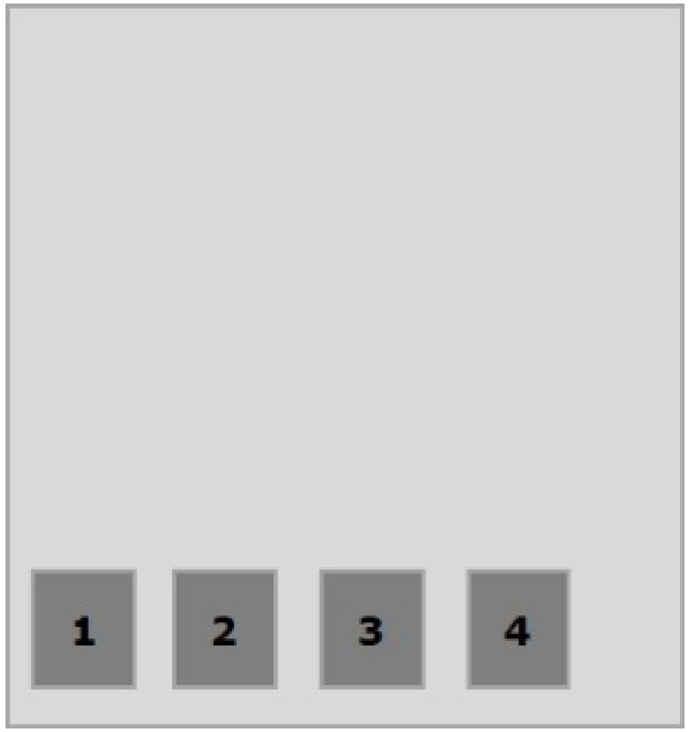


```
<!DOCTYPE html>
<html>
<head>
<style>
    .flex-container {
        display: flex;
        height: 200px;
        align-items: flex-start;
        background-color: DodgerBlue;
    }
    .flex-container > div {
        background-color: #f1f1f1;
        width: 100px;
        margin: 10px;
        text-align: center;
        line-height: 75px;
        font-size: 30px;
    }
</style>
</head>
<body>
    <h1>The align-items Property</h1>
    <p>The "align-items: flex-start;" aligns the flex items at the top of the container:</p>
    <div class="flex-container">
        <div>1</div>
        <div>2</div>
        <div>3</div>
    </div>
</body>
</html>
```

CSS Flexbox Align Items

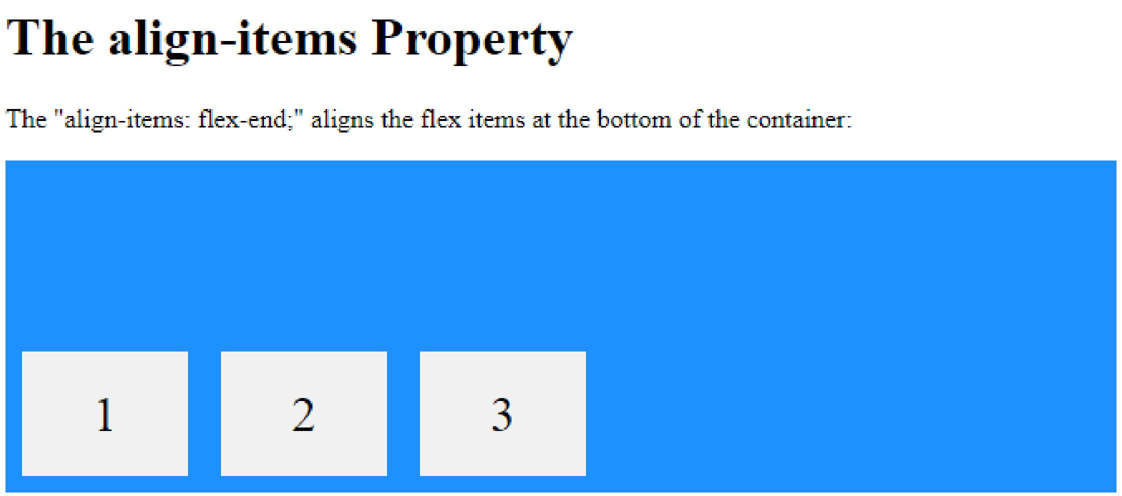
flex-end

- On passing ***flex-end*** value to the property ***align-items***, the flex-items are aligned ***vertically*** at the ***bottom of the container***.



CSS Flexbox Align Items

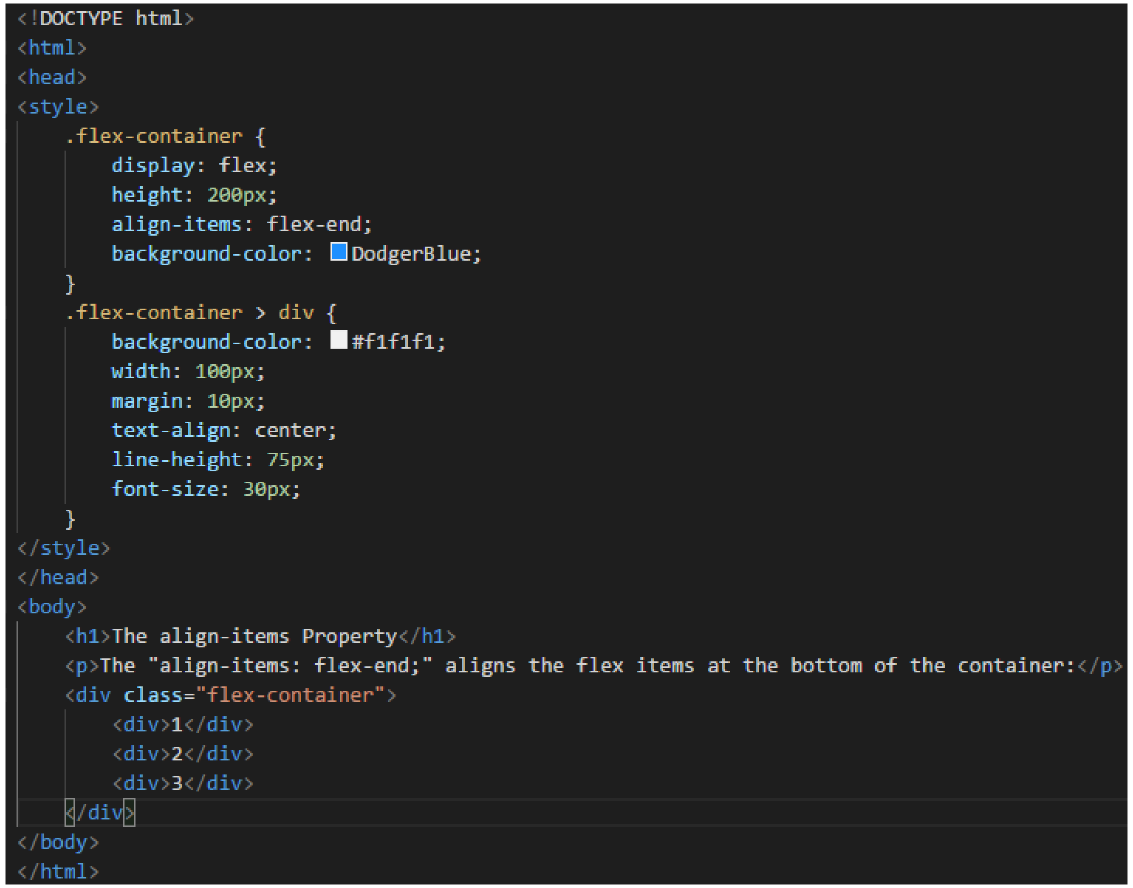
The example demonstrates the result of passing the value ***flex-end*** to the ***align-items*** property.



The align-items Property

The "align-items: flex-end;" aligns the flex items at the bottom of the container:

1 2 3

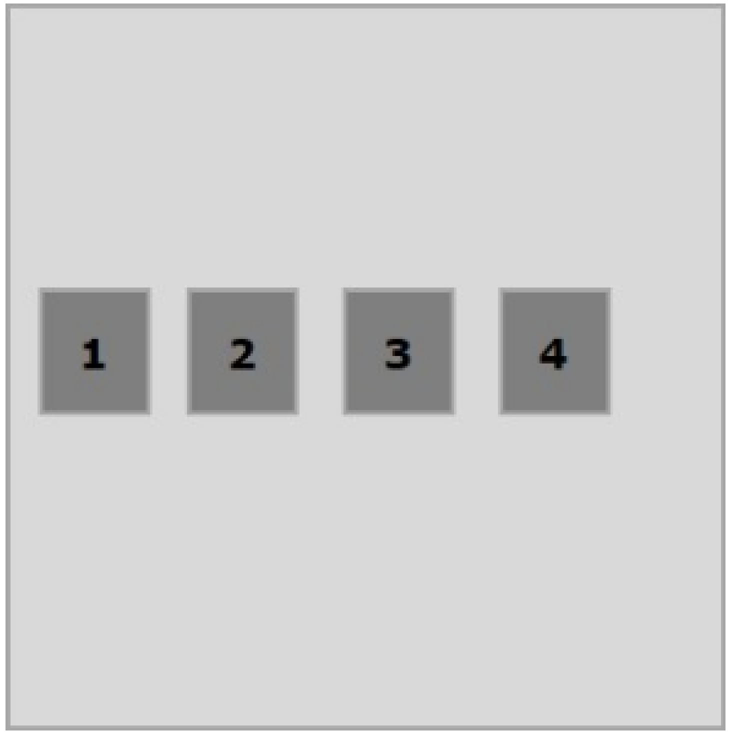


```
<!DOCTYPE html>
<html>
<head>
<style>
    .flex-container {
        display: flex;
        height: 200px;
        align-items: flex-end;
        background-color: DodgerBlue;
    }
    .flex-container > div {
        background-color: #f1f1f1;
        width: 100px;
        margin: 10px;
        text-align: center;
        line-height: 75px;
        font-size: 30px;
    }
</style>
</head>
<body>
    <h1>The align-items Property</h1>
    <p>The "align-items: flex-end;" aligns the flex items at the bottom of the container:</p>
    <div class="flex-container">
        <div>1</div>
        <div>2</div>
        <div>3</div>
    </div>
</body>
</html>
```

CSS Flexbox Align Items

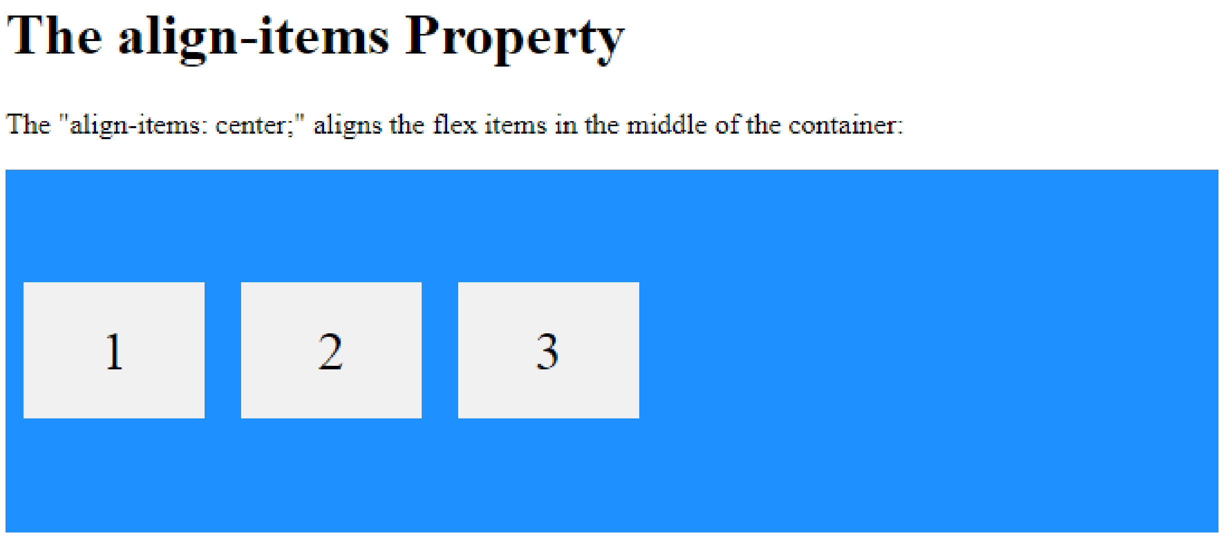
Center

- On passing ***center*** value to the property ***align-items***, the flex-items are aligned ***vertically*** at the ***center of the container***.



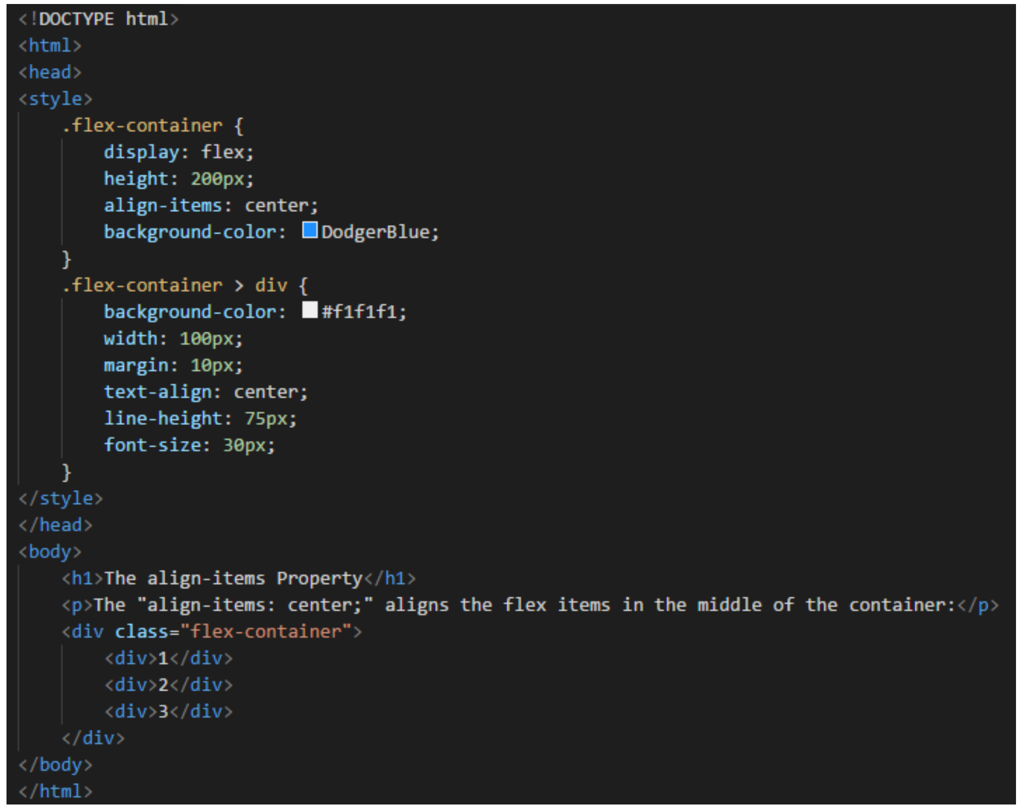
CSS Flexbox Align Items

The example demonstrates the result of passing the value ***flex-center*** to the ***align-items*** property.



The align-items Property

The "align-items: center;" aligns the flex items in the middle of the container:

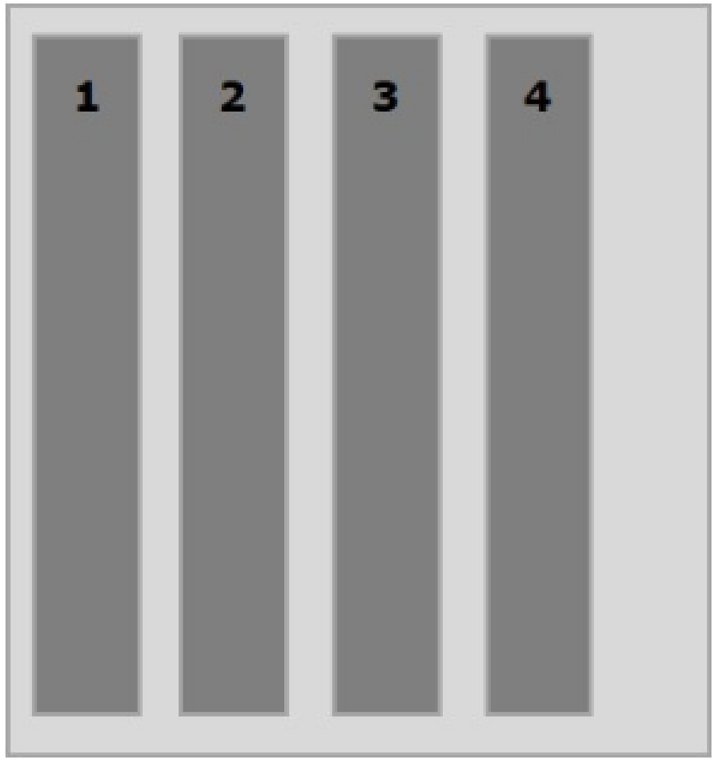


```
<!DOCTYPE html>
<html>
<head>
<style>
    .flex-container {
        display: flex;
        height: 200px;
        align-items: center;
        background-color: DodgerBlue;
    }
    .flex-container > div {
        background-color: #f1f1f1;
        width: 100px;
        margin: 10px;
        text-align: center;
        line-height: 75px;
        font-size: 30px;
    }
</style>
</head>
<body>
    <h1>The align-items Property</h1>
    <p>The "align-items: center;" aligns the flex items in the middle of the container:</p>
    <div class="flex-container">
        <div>1</div>
        <div>2</div>
        <div>3</div>
    </div>
</body>
</html>
```

CSS Flexbox Align Items

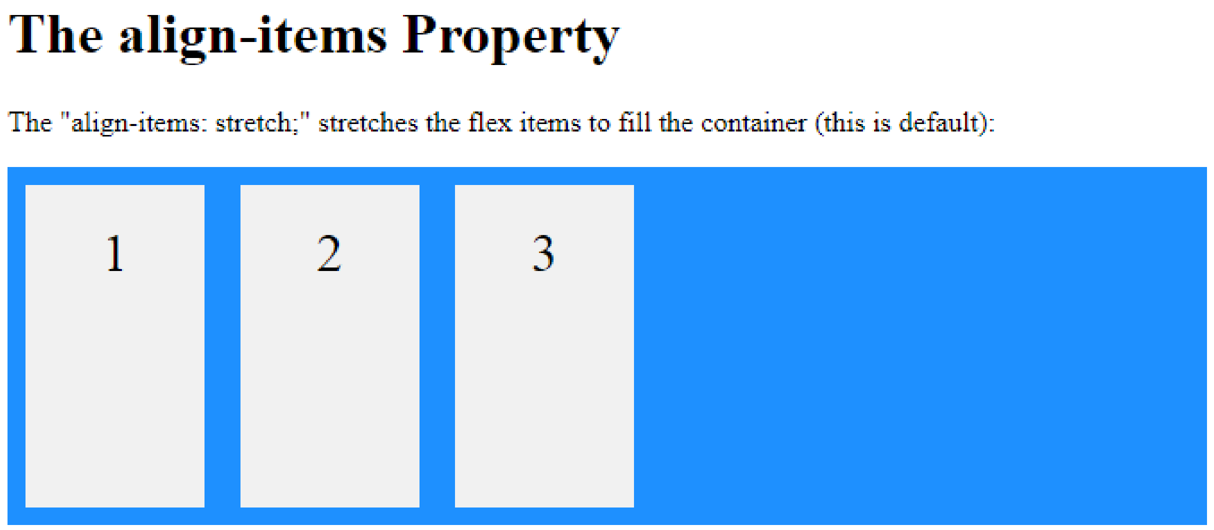
Stretch

- On passing ***stretch*** value to the property ***align-items***, the flex-items are aligned ***vertically*** such that they fill up the ***whole vertical space of the container***.



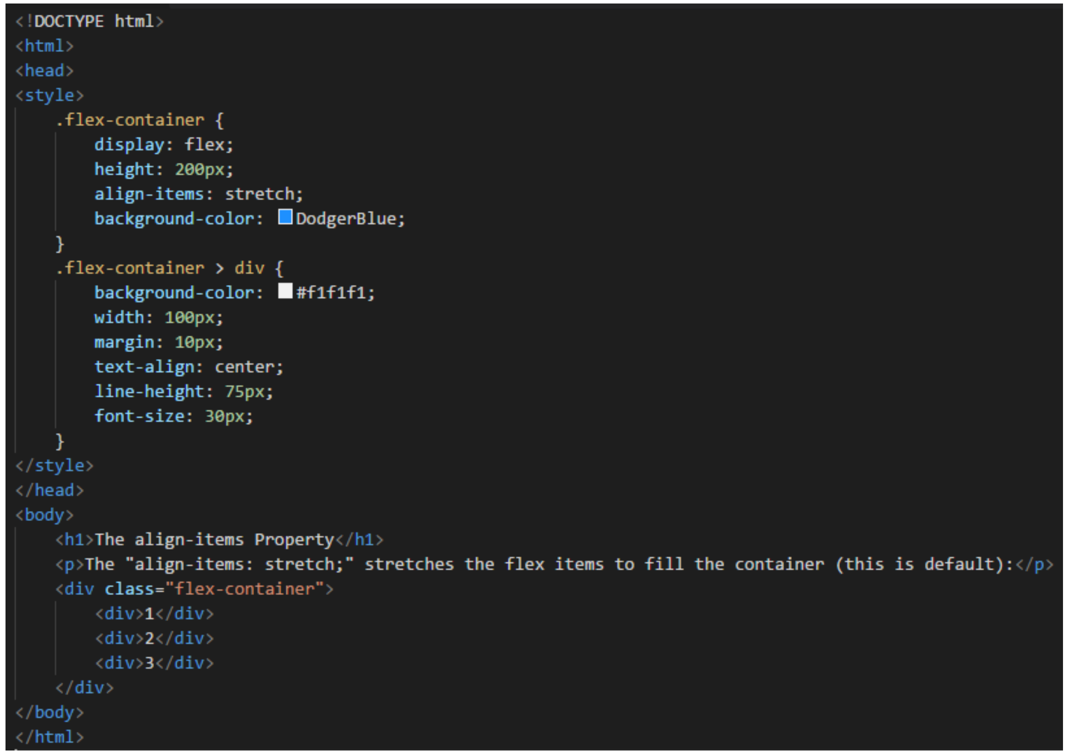
CSS Flexbox Align Items

The example demonstrates the result of passing the value ***stretch*** to the ***align-items*** property.



The align-items Property

The "align-items: stretch;" stretches the flex items to fill the container (this is default):

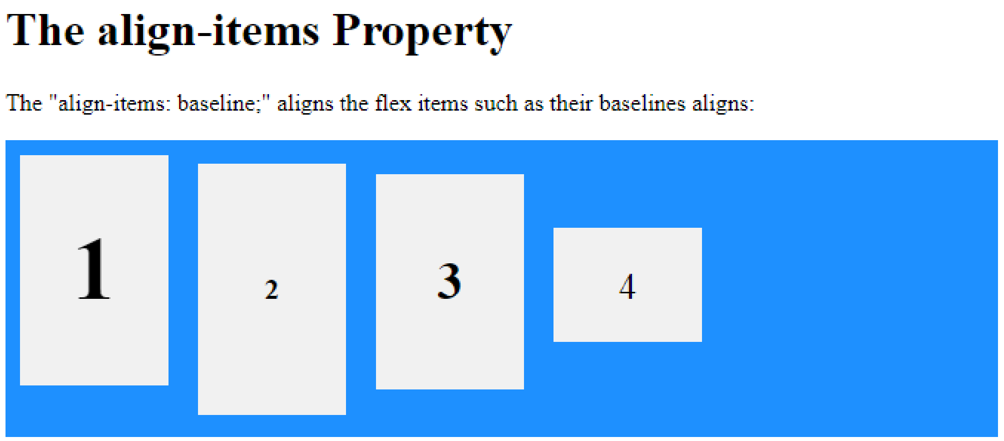


```
<!DOCTYPE html>
<html>
<head>
<style>
    .flex-container {
        display: flex;
        height: 200px;
        align-items: stretch;
        background-color: DodgerBlue;
    }
    .flex-container > div {
        background-color: #f1f1f1;
        width: 100px;
        margin: 10px;
        text-align: center;
        line-height: 75px;
        font-size: 30px;
    }
</style>
</head>
<body>
    <h1>The align-items Property</h1>
    <p>The "align-items: stretch;" stretches the flex items to fill the container (this is default):</p>
    <div class="flex-container">
        <div>1</div>
        <div>2</div>
        <div>3</div>
    </div>
</body>
</html>
```

CSS Flexbox Align Items

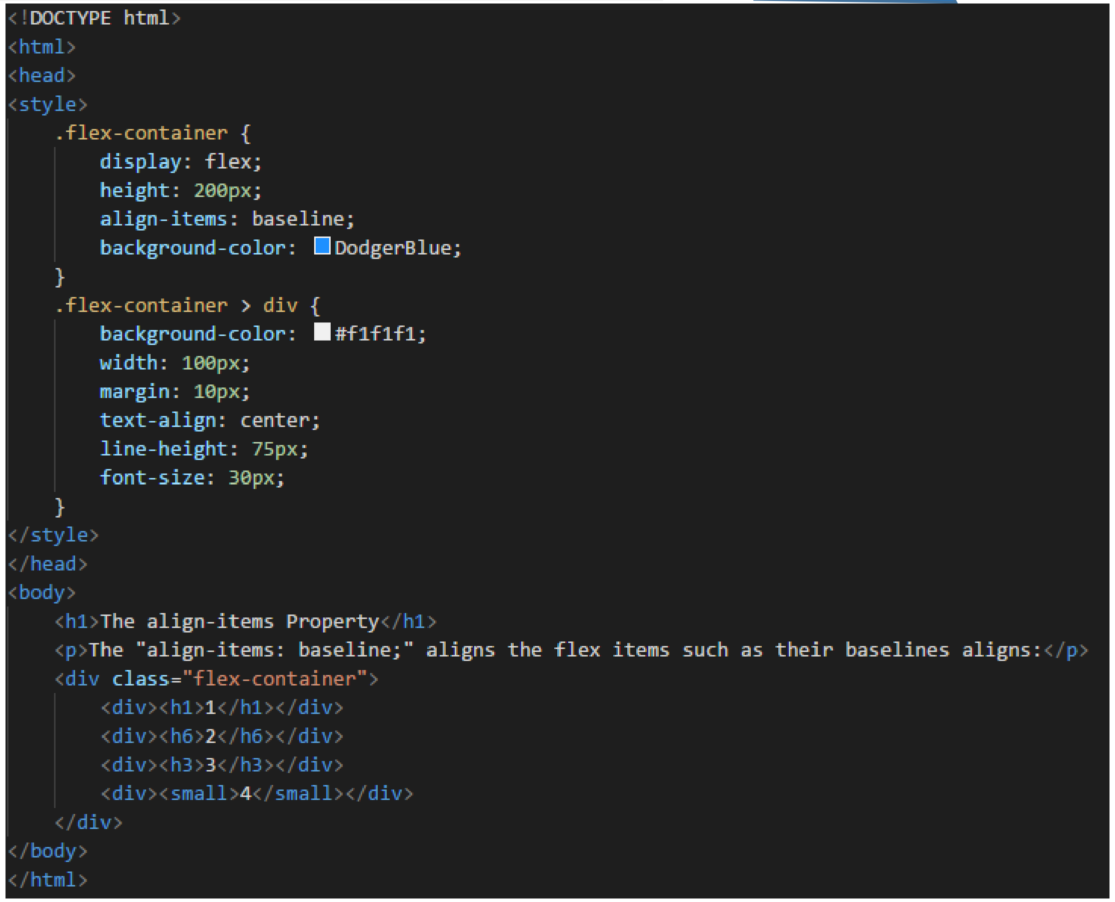
Baseline

- On passing ***baseline*** value to the property ***align-items***, the flex-items are aligned such that the ***baseline*** of their text align along a ***horizontal line***.
- The example demonstrates the result of passing the value ***baseline*** to the ***align-items*** property.
- The example uses different font-size to demonstrate that the items gets aligned by the text baseline.



The align-items Property

The "align-items: baseline;" aligns the flex items such as their baselines aligns:



```
<!DOCTYPE html>
<html>
<head>
<style>
    .flex-container {
        display: flex;
        height: 200px;
        align-items: baseline;
        background-color: DodgerBlue;
    }
    .flex-container > div {
        background-color: #f1f1f1;
        width: 100px;
        margin: 10px;
        text-align: center;
        line-height: 75px;
        font-size: 30px;
    }
</style>
</head>
<body>
    <h1>The align-items Property</h1>
    <p>The "align-items: baseline;" aligns the flex items such as their baselines aligns:</p>
    <div class="flex-container">
        <div><h1>1</h1></div>
        <div><h6>2</h6></div>
        <div><h3>3</h3></div>
        <div><small>4</small></div>
    </div>
</body>
</html>
```

CSS Flexbox Align Content

- In case the flex-container has multiple lines (when, flex-wrap: wrap), the align-content property defines the alignment of each line within the container.
- Syntax is:

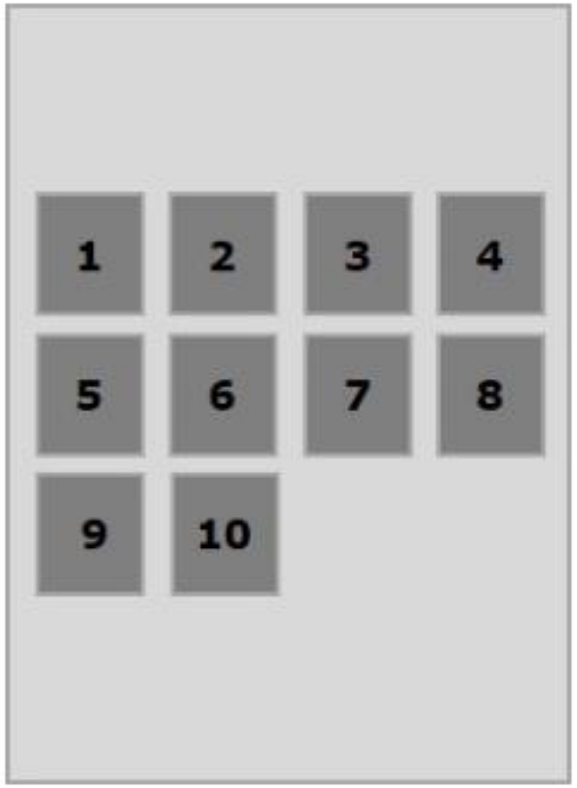
align-content: flex-start | flex-end | center | space-between | space-around | stretch;

- This property accepts the following values:
  - ***stretch*** − The lines in the content will stretch to fill up the remaining space.
  - ***flex-start*** − All the lines in the content are packed at the start of the container.
  - ***flex-end*** − All the lines in the content are packed at the end of the container.
  - ***center*** − All the lines in the content are packed at the center of the container.
  - ***space-between*** − The extra space is distributed between the lines evenly.
  - ***space-around*** − The extra space is distributed between the lines evenly with equal space around each line (including the first and last lines)

CSS Flexbox Align Content cont.

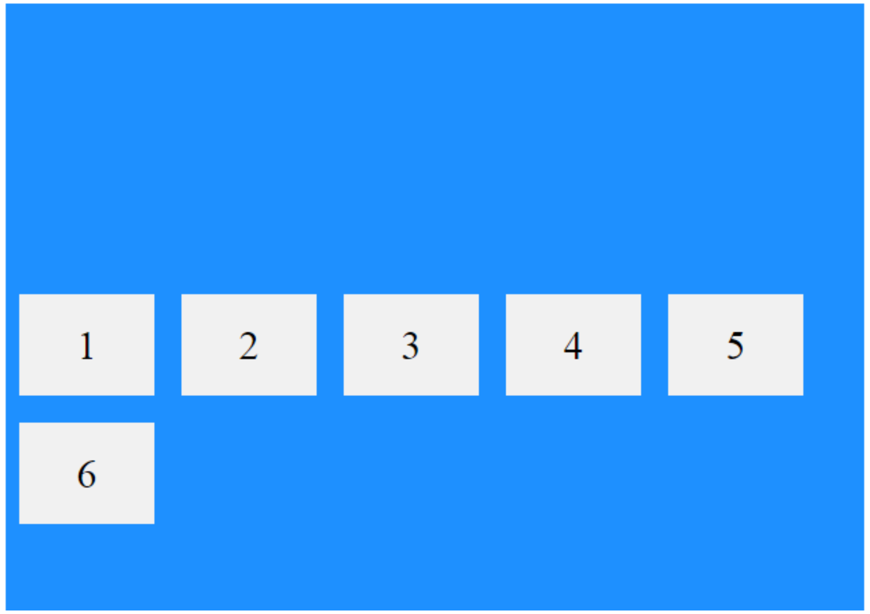
Center

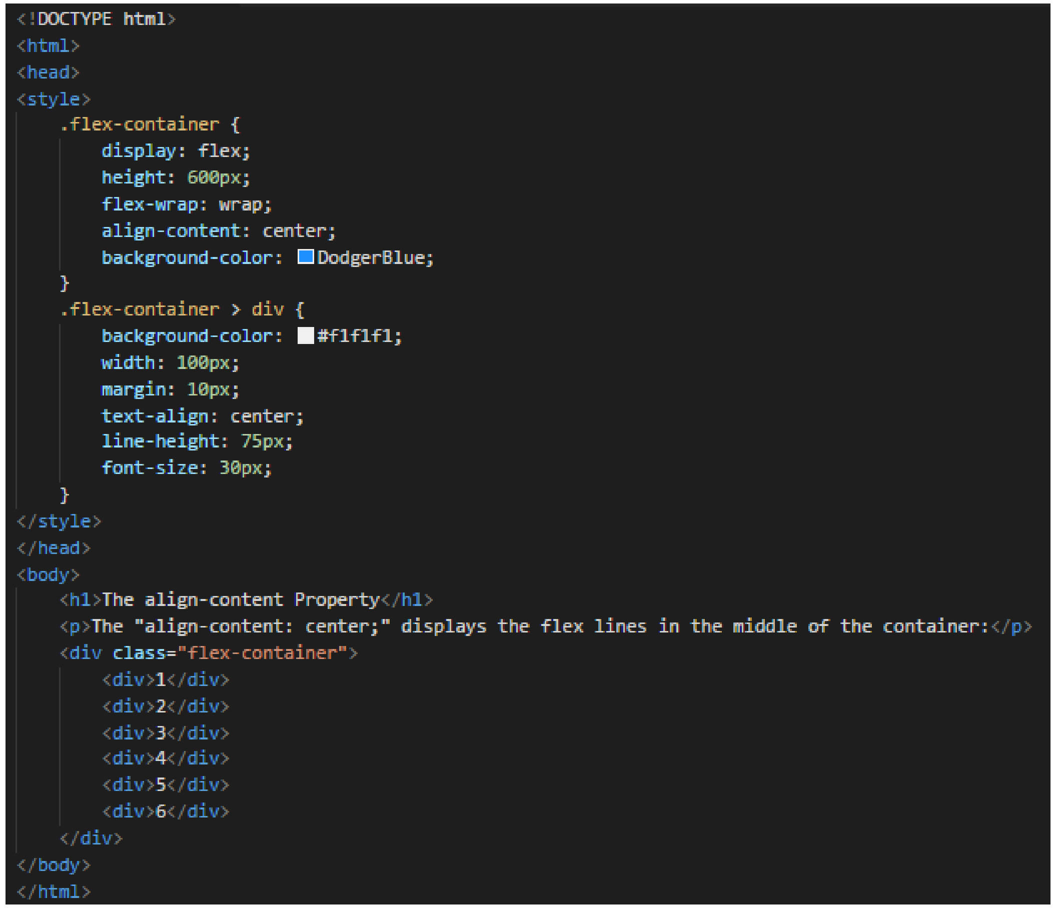
- On passing center value to the property ***align-content***, all the lines are packed at the ***center of the container***.



CSS Flexbox Align Content cont.

- The example demonstrates the result of passing the value ***center*** to the ***align-content*** property.



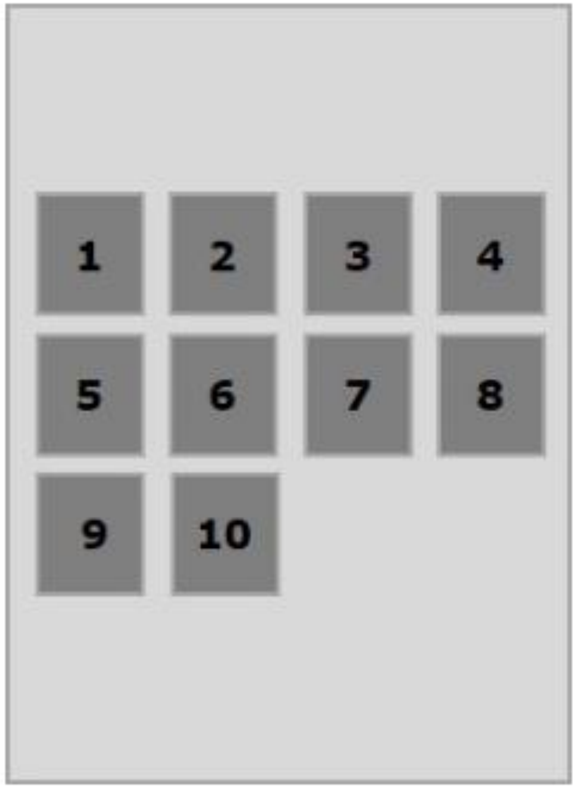


```
<!DOCTYPE html>
<html>
<head>
<style>
    .flex-container {
        display: flex;
        height: 600px;
        flex-wrap: wrap;
        align-content: center;
        background-color: DodgerBlue;
    }
    .flex-container > div {
        background-color: #f1f1f1;
        width: 100px;
        margin: 10px;
        text-align: center;
        line-height: 75px;
        font-size: 30px;
    }
</style>
</head>
<body>
    <h1>The align-content Property</h1>
    <p>The "align-content: center;" displays the flex lines in the middle of the container:</p>
    <div class="flex-container">
        <div>1</div>
        <div>2</div>
        <div>3</div>
        <div>4</div>
        <div>5</div>
        <div>6</div>
    </div>
</body>
</html>
```

CSS Flexbox Align Content cont.

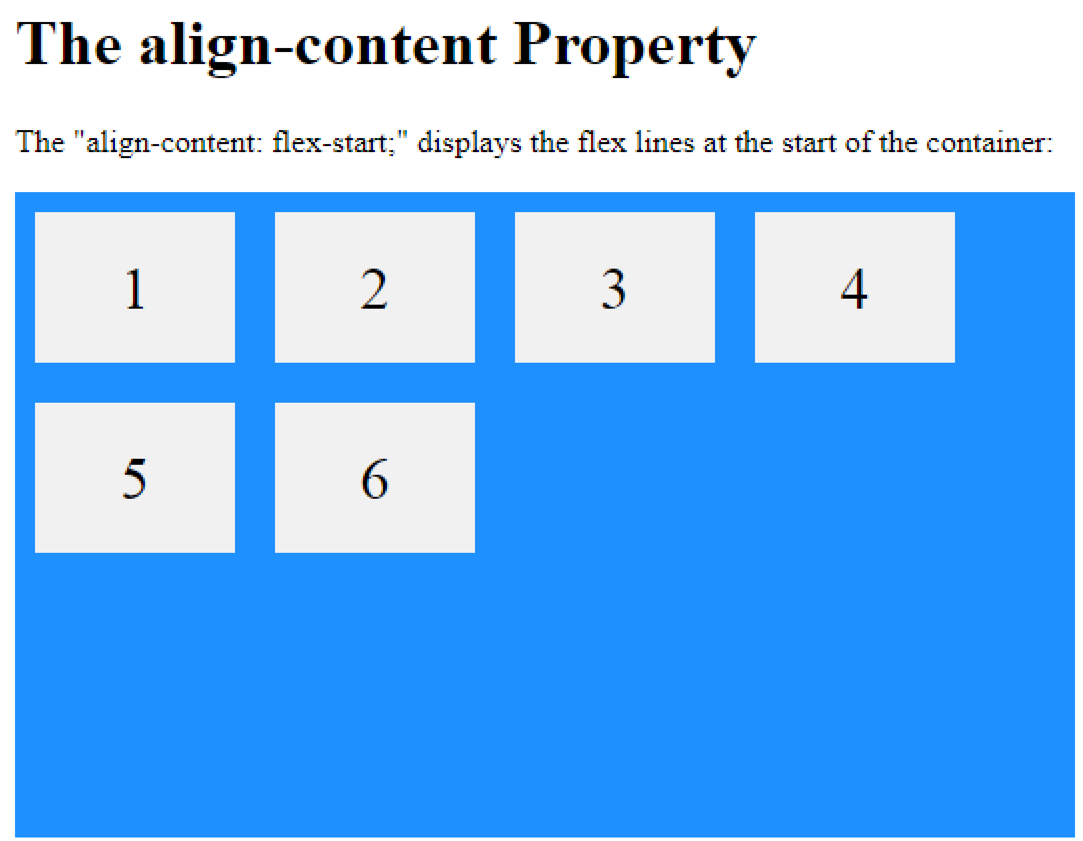
flex-start

- On passing ***flex-start*** value to the property ***align-content***, all the lines are packed at the start of the container.



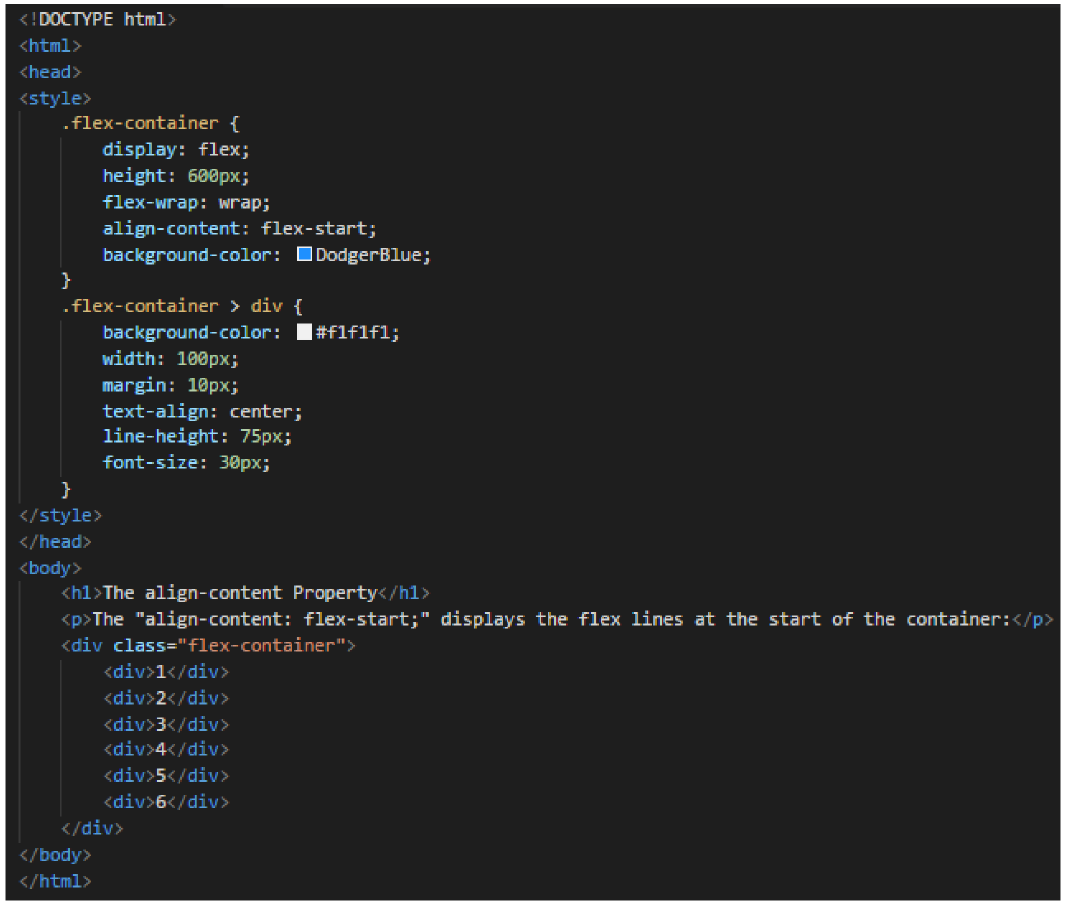
CSS Flexbox Align Content cont.

- The example demonstrates the result of passing the value ***flex-start*** to the ***align-content*** property.



The align-content Property

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

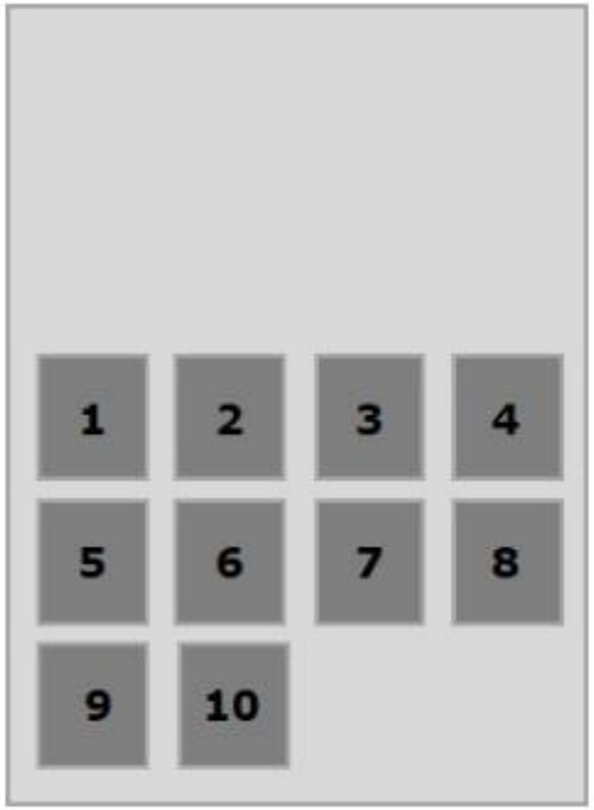


```
<!DOCTYPE html>
<html>
<head>
<style>
    .flex-container {
        display: flex;
        height: 600px;
        flex-wrap: wrap;
        align-content: flex-start;
        background-color: DodgerBlue;
    }
    .flex-container > div {
        background-color: #f1f1f1;
        width: 100px;
        margin: 10px;
        text-align: center;
        line-height: 75px;
        font-size: 30px;
    }
</style>
</head>
<body>
    <h1>The align-content Property</h1>
    <p>The "align-content: flex-start;" displays the flex lines at the start of the container:</p>
    <div class="flex-container">
        <div>1</div>
        <div>2</div>
        <div>3</div>
        <div>4</div>
        <div>5</div>
        <div>6</div>
    </div>
</body>
</html>
```

CSS Flexbox Align Content cont.

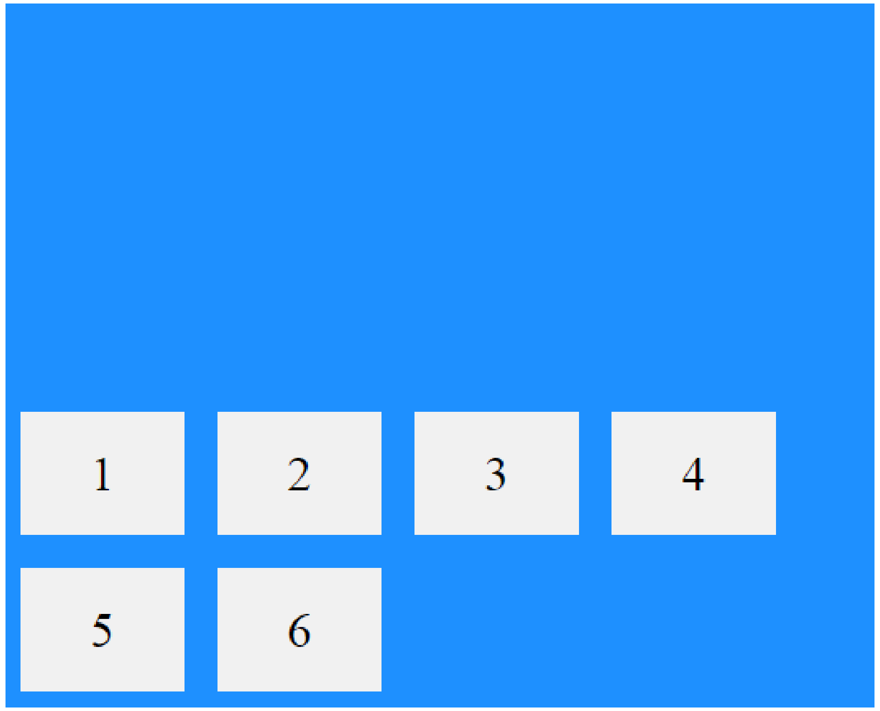
flex-end

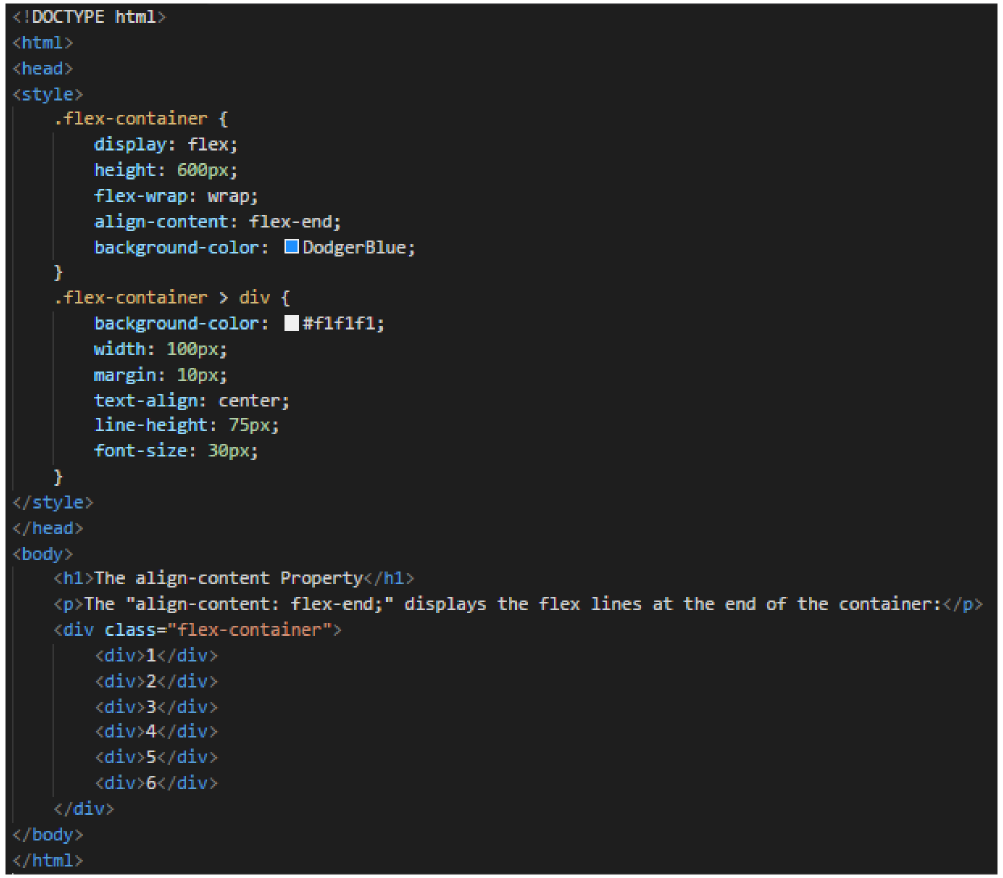
- On passing ***flex-end*** value to the property ***align-content***, all the lines are packed at the ***end of the container***.



CSS Flexbox Align Content cont.

- The example demonstrates the result of passing the value ***flex-end*** to the ***align-content*** property.



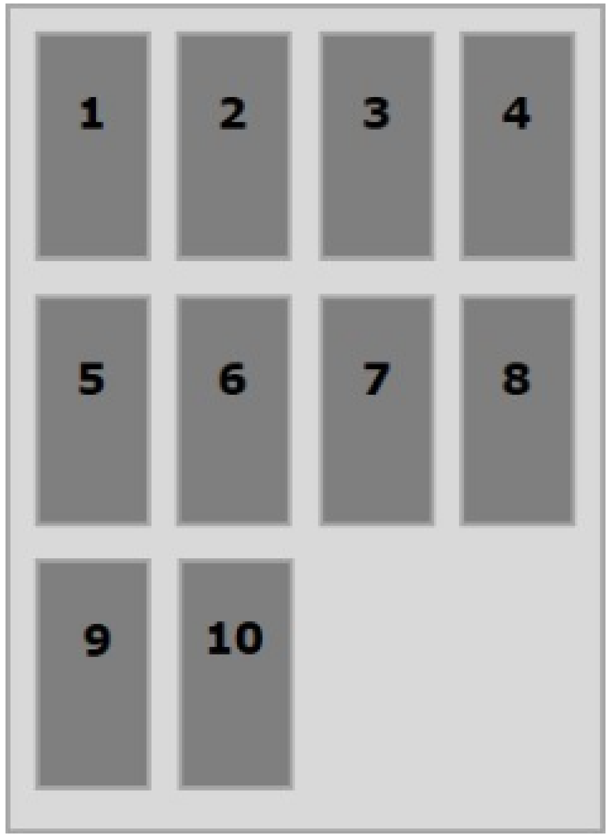


```html
<!DOCTYPE html>
<html>
<head>
<style>
    .flex-container {
        display: flex;
        height: 600px;
        flex-wrap: wrap;
        align-content: flex-end;
        background-color: DodgerBlue;
    }
    .flex-container > div {
        background-color: #f1f1f1;
        width: 100px;
        margin: 10px;
        text-align: center;
        line-height: 75px;
        font-size: 30px;
    }
</style>
</head>
<body>
    <h1>The align-content Property</h1>
    <p>The "align-content: flex-end;" displays the flex lines at the end of the container:</p>
    <div class="flex-container">
        <div>1</div>
        <div>2</div>
        <div>3</div>
        <div>4</div>
        <div>5</div>
        <div>6</div>
    </div>
</body>
</html>
```

CSS Flexbox Align Content cont.

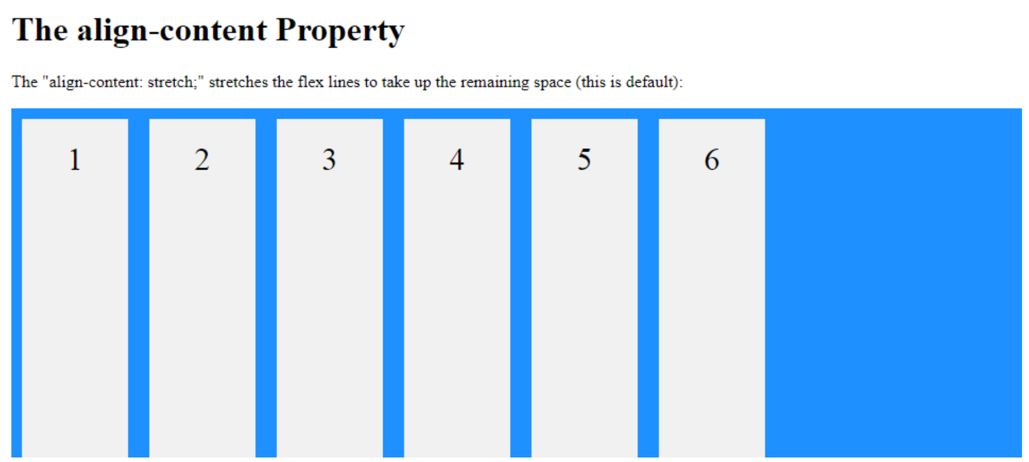
Stretch

- On passing ***stretch*** value to the property ***align-content***, the lines will stretch to fill up the remaining space.



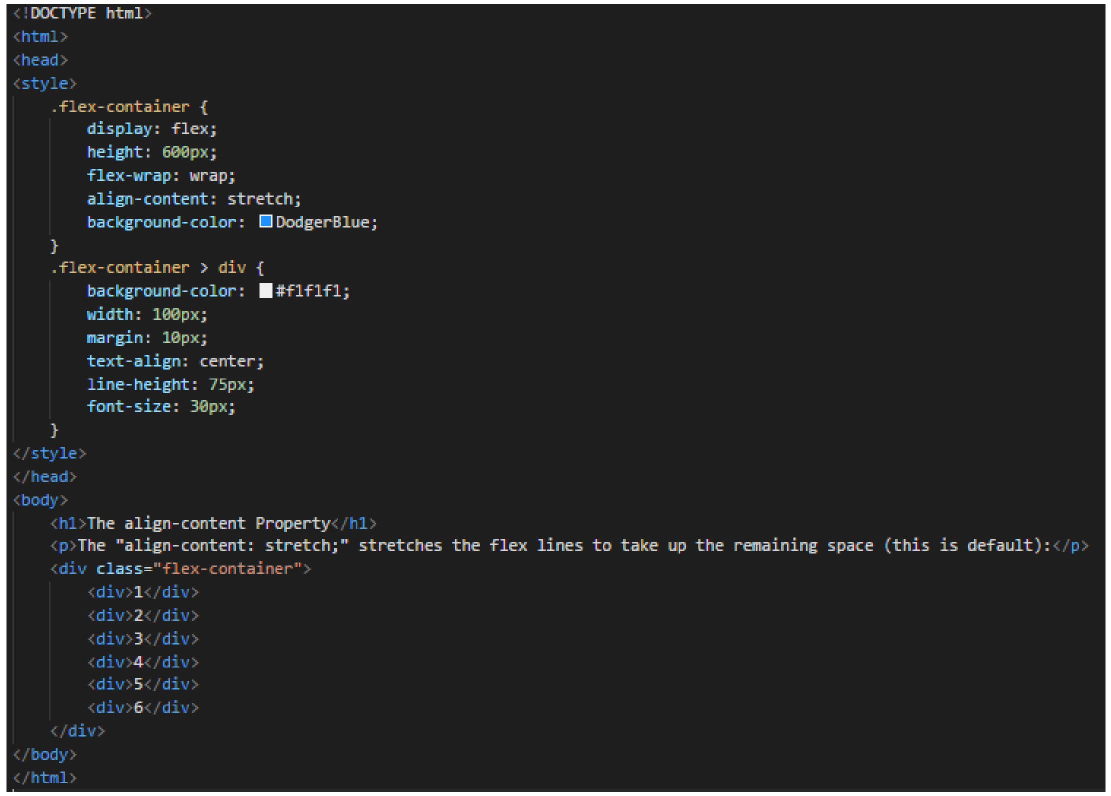
CSS Flexbox Align Content cont.

- The example demonstrates the result of passing the value ***stretch*** to the ***align-content*** property.



The align-content Property

The "align-content: stretch;" stretches the flex lines to take up the remaining space (this is default):

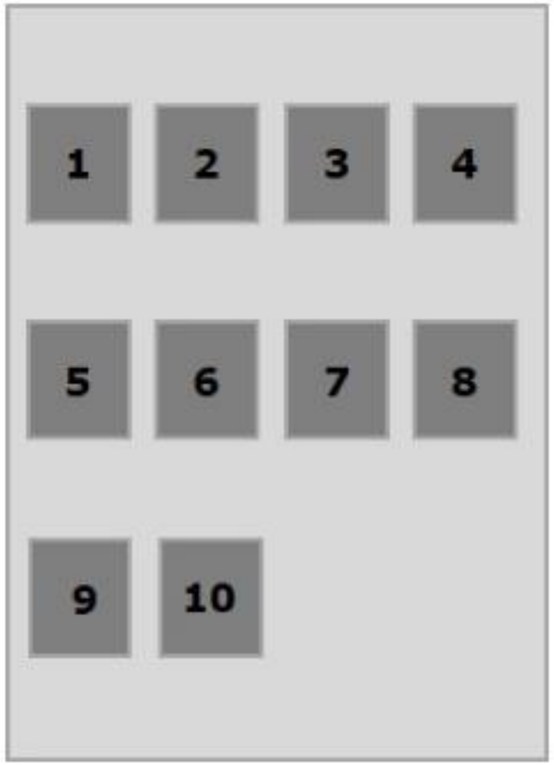


```
<!DOCTYPE html>
<html>
<head>
<style>
    .flex-container {
        display: flex;
        height: 600px;
        flex-wrap: wrap;
        align-content: stretch;
        background-color: DodgerBlue;
    }
    .flex-container > div {
        background-color: #f1f1f1;
        width: 100px;
        margin: 10px;
        text-align: center;
        line-height: 75px;
        font-size: 30px;
    }
</style>
</head>
<body>
    <h1>The align-content Property</h1>
    <p>The "align-content: stretch;" stretches the flex lines to take up the remaining space (this is default):</p>
    <div class="flex-container">
        <div>1</div>
        <div>2</div>
        <div>3</div>
        <div>4</div>
        <div>5</div>
        <div>6</div>
    </div>
</body>
</html>
```

CSS Flexbox Align Content cont.

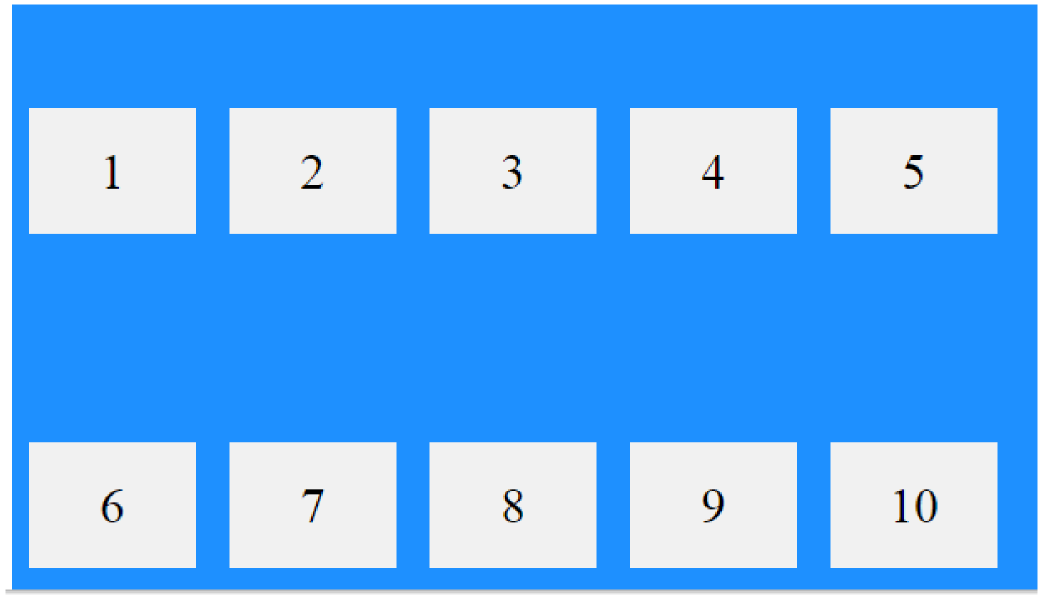
Space-around

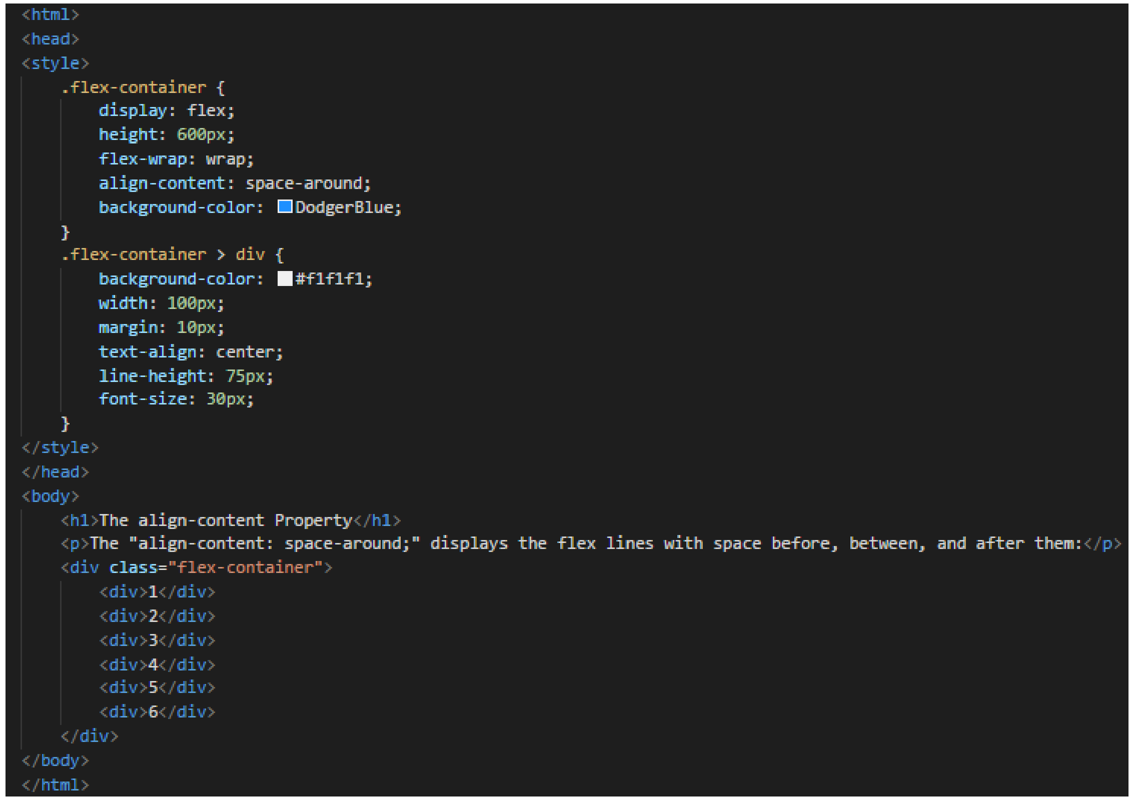
- On passing ***space-around*** value to the property ***align-content***, the extra space is distributed between the lines evenly with equal space around each line (including the first and last lines).



CSS Flexbox Align Content cont.

- The example demonstrates the result of passing the value ***space-around*** to the ***align-content*** property.



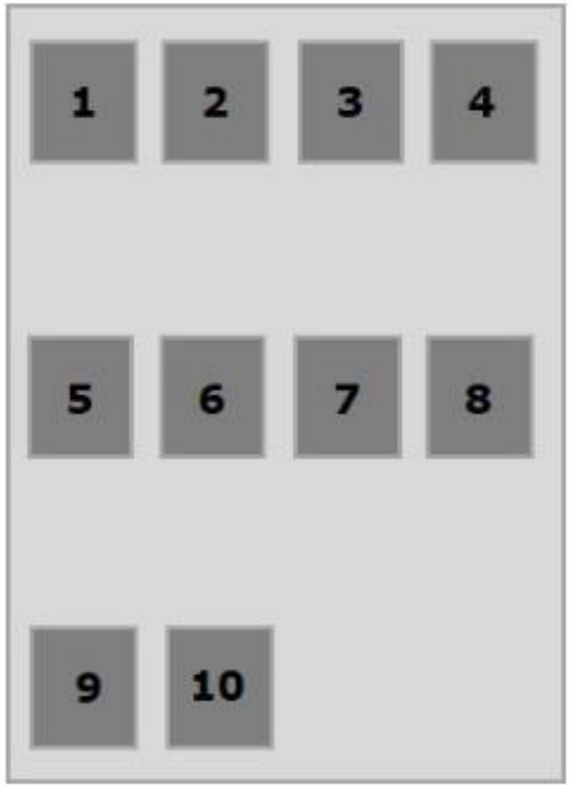


```
<html>
<head>
<style>
    .flex-container {
        display: flex;
        height: 600px;
        flex-wrap: wrap;
        align-content: space-around;
        background-color: DodgerBlue;
    }
    .flex-container > div {
        background-color: #f1f1f1;
        width: 100px;
        margin: 10px;
        text-align: center;
        line-height: 75px;
        font-size: 30px;
    }
</style>
</head>
<body>
    <h1>The align-content Property</h1>
    <p>The "align-content: space-around;" displays the flex lines with space before, between, and after them:</p>
    <div class="flex-container">
        <div>1</div>
        <div>2</div>
        <div>3</div>
        <div>4</div>
        <div>5</div>
        <div>6</div>
    </div>
</body>
</html>
```

CSS Flexbox Align Content cont.

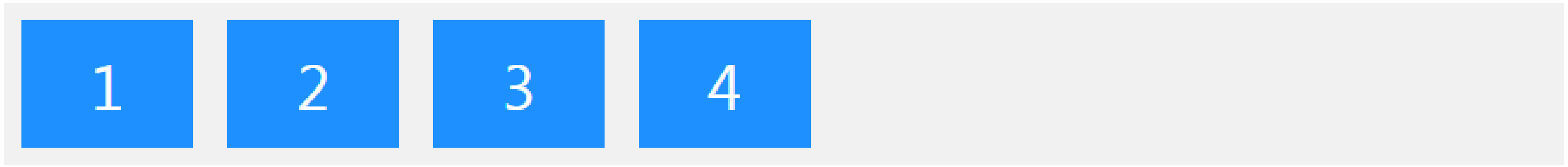
Space-between

- On passing ***space-between*** value to the property ***align-content***, the extra space is distributed between the lines evenly, where the first line will be at the top and the last line will be at the bottom of the container.



CSS Flexbox Child Elements (Items)

- The direct child elements of a flex container automatically becomes flexible (flex) items.
- The element below represents four blue flex items inside a grey flex container.

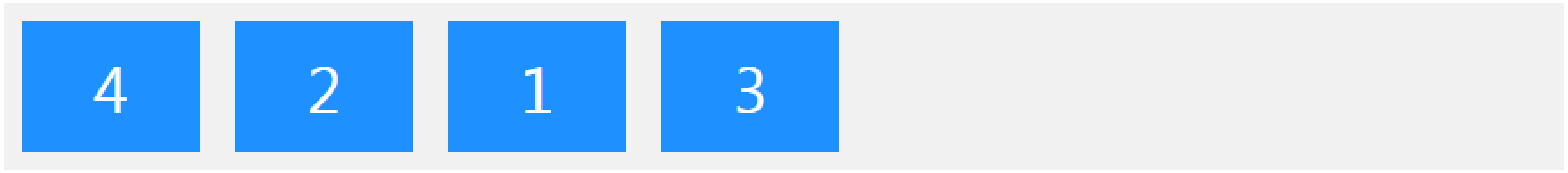


- The flex item properties are:
    - order
    - flex-grow
    - flex-shrink
    - flex-basis
    - flex
    - align-self

CSS Flexbox Child Elements cont.

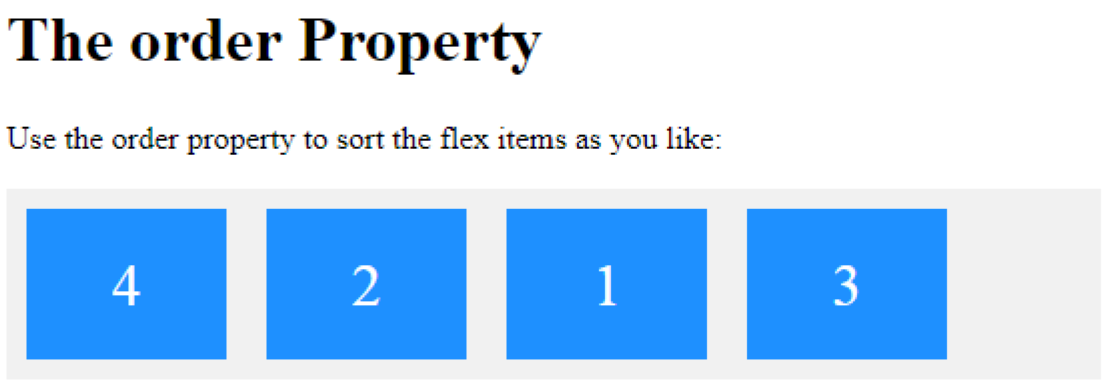
The order Property

- The ***flex-order*** property is used to define the order of the flexbox item.
- 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.



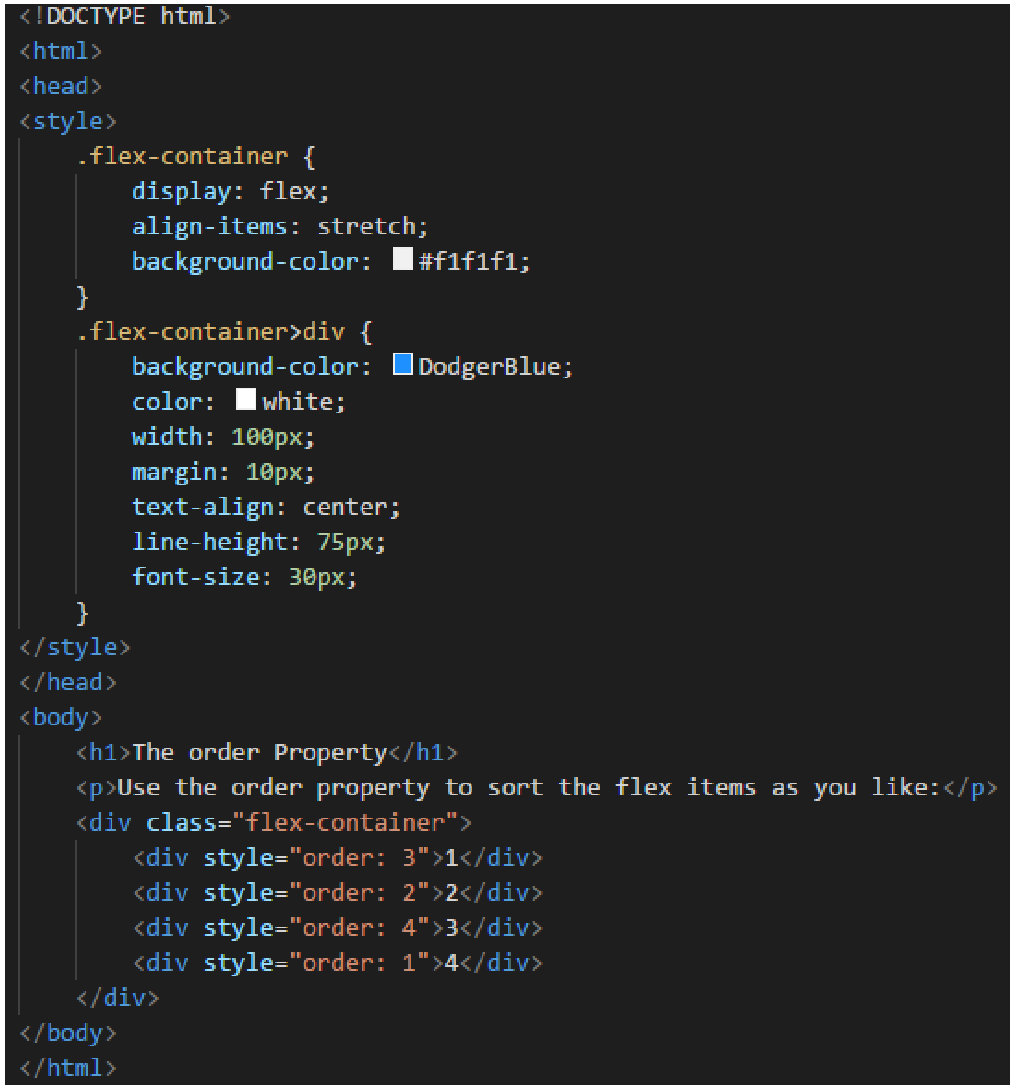
CSS Flexbox Child Elements cont.

- The example demonstrates the ***flex-order*** property.
- Here we are creating 4 boxes with the labels one, two, three, four, and we are reordering them in the order four, two, one, three, using the ***flex-order*** property.



The order Property

Use the order property to sort the flex items as you like:



```html
<!DOCTYPE html>
<html>
<head>
<style>
    .flex-container {
        display: flex;
        align-items: stretch;
        background-color: #f1f1f1;
    }
    .flex-container>div {
        background-color: DodgerBlue;
        color: white;
        width: 100px;
        margin: 10px;
        text-align: center;
        line-height: 75px;
        font-size: 30px;
    }
</style>
</head>
<body>
    <h1>The order Property</h1>
    <p>Use the order property to sort the flex items as you like:</p>
    <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>
</body>
</html>
```

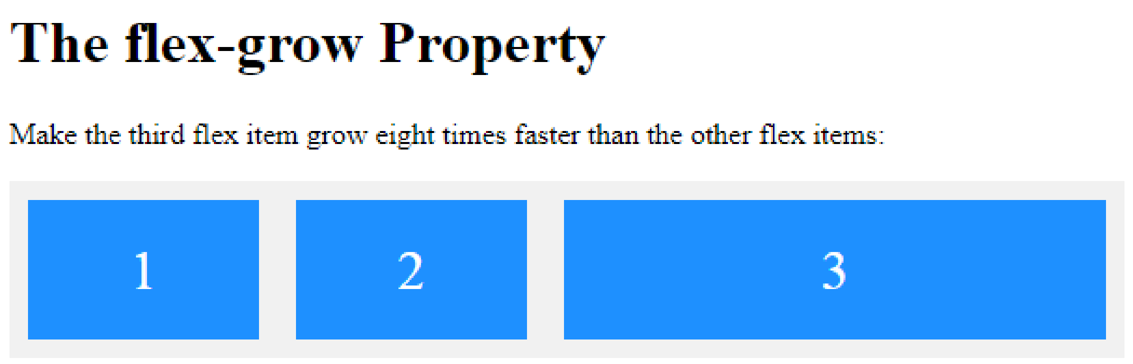
CSS Flexbox Child Elements cont.

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***.
- In case of excess space in the container, it specifies how much a particular flex-item should grow.
- Let’s look in to an example.

CSS Flexbox Child Elements cont.

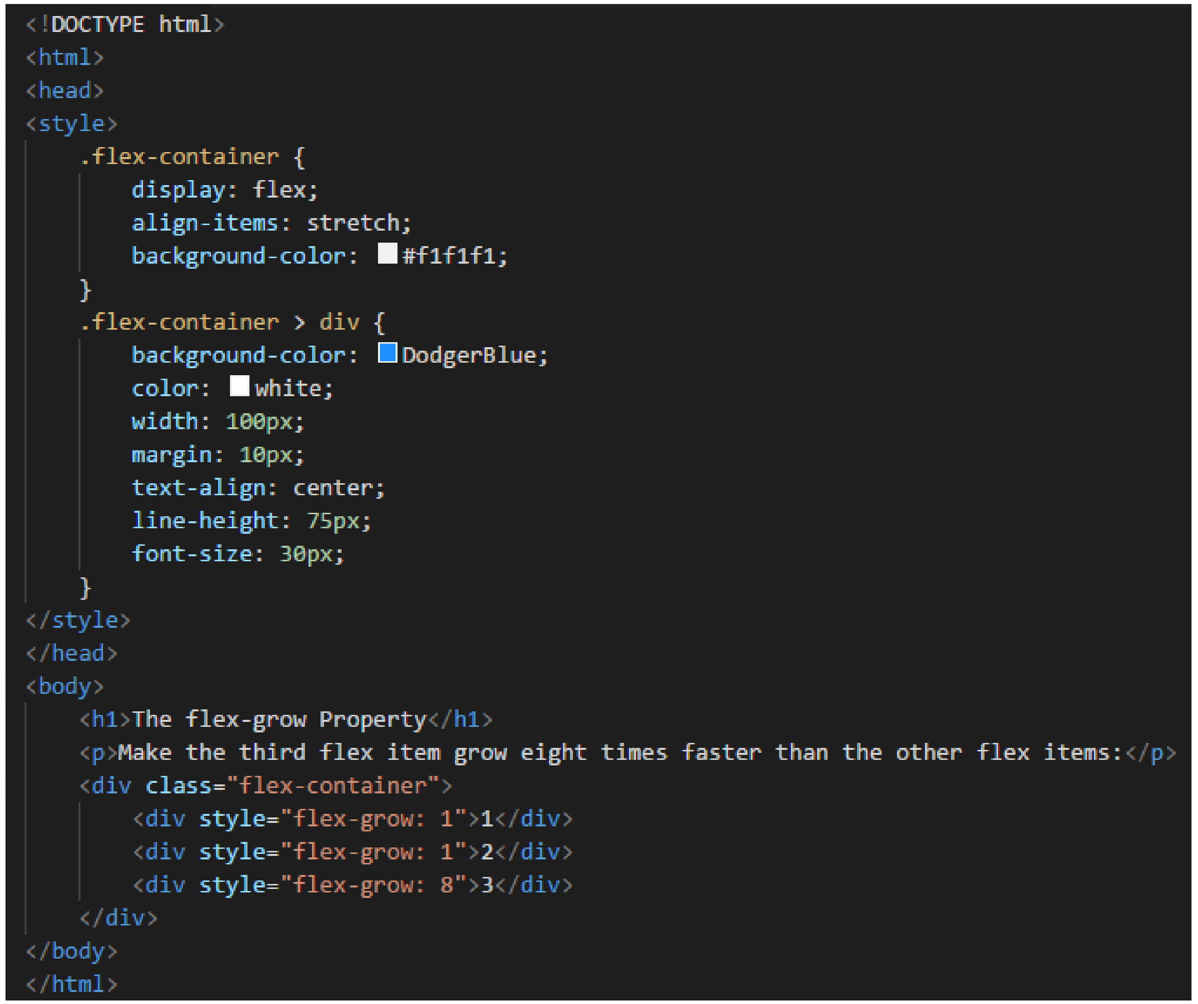
- The example demonstrates the ***flex-grow*** property.
- Here we are creating 3 boxes with the labels one, two, three, and we are making the 3 box grow eight times faster than the other flex items, using the ***flex-grow*** property.



The flex-grow Property

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

1 2 3



```html
<!DOCTYPE html>
<html>
<head>
<style>
    .flex-container {
        display: flex;
        align-items: stretch;
        background-color: #f1f1f1;
    }
    .flex-container > div {
        background-color: DodgerBlue;
        color: white;
        width: 100px;
        margin: 10px;
        text-align: center;
        line-height: 75px;
        font-size: 30px;
    }
</style>
</head>
<body>
    <h1>The flex-grow Property</h1>
    <p>Make the third flex item grow eight times faster than the other flex items:</p>
    <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>
</body>
</html>
```

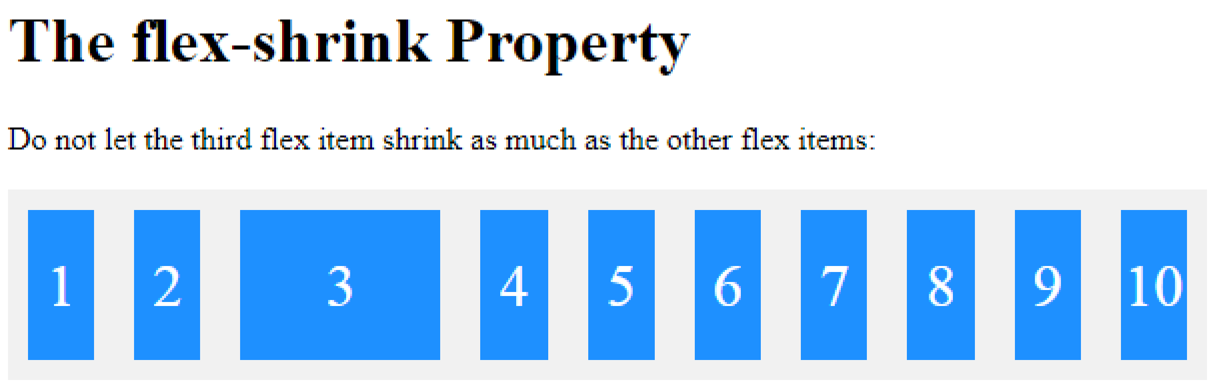
CSS Flexbox Child Elements cont.

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***.
- In case there is not enough space in the container, it specifies how much a flex-item should shrink.
- Let’s look in to an example.

CSS Flexbox Child Elements cont.

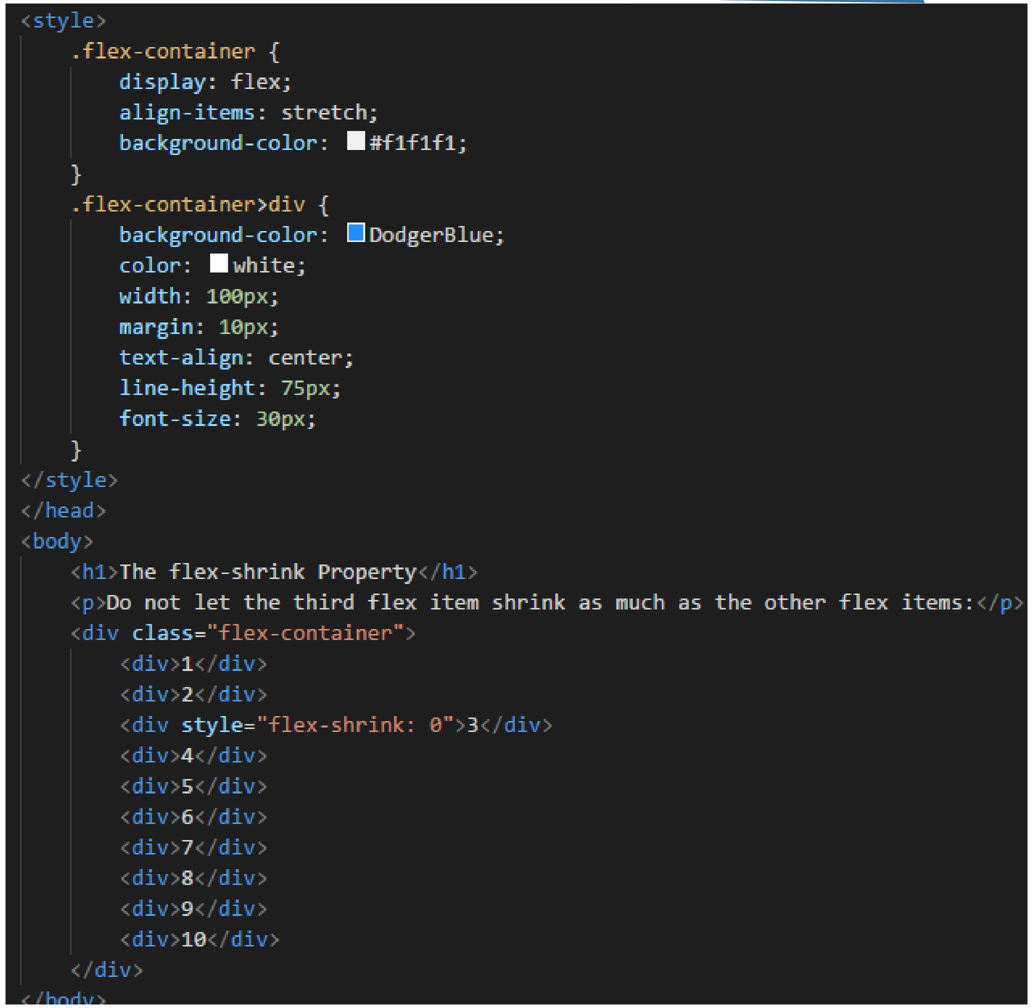
- The example demonstrates the ***flex-shrink*** property.
- Here we are creating 10 boxes with the labels and we are not allowing the third flex item to shrink as much as the other flex items using the ***flex-shrink*** property.



The flex-shrink Property

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

1 2 3 4 5 6 7 8 9 10

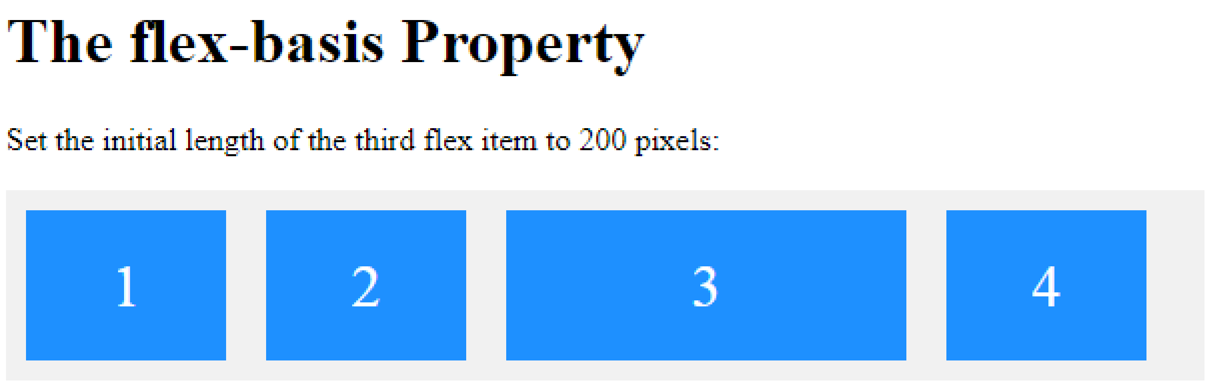


```
<style>
    .flex-container {
        display: flex;
        align-items: stretch;
        background-color: #f1f1f1;
    }
    .flex-container>div {
        background-color: DodgerBlue;
        color: white;
        width: 100px;
        margin: 10px;
        text-align: center;
        line-height: 75px;
        font-size: 30px;
    }
</style>
</head>
<body>
    <h1>The flex-shrink Property</h1>
    <p>Do not let the third flex item shrink as much as the other flex items:</p>
    <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>
</body>
```

CSS Flexbox Child Elements cont.

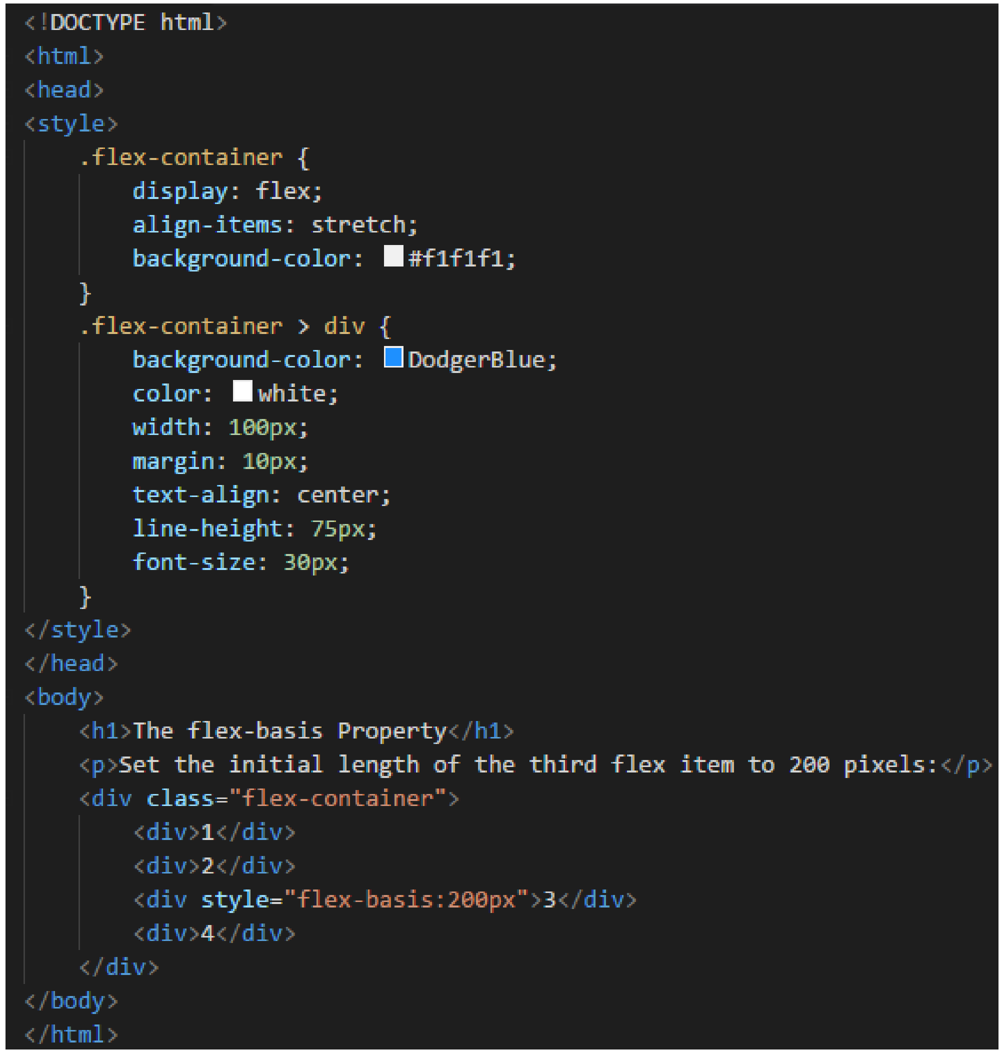
The flex-basis Property

- We use the ***flex-basis*** property to define the default size of the flex-item before the space is distributed.
- The example demonstrates the usage of the ***flex-basis*** property. Here we are creating 3 boxes and fixing the size to third flex item to 200 px.



The flex-basis Property

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



```
<!DOCTYPE html>
<html>
<head>
<style>
    .flex-container {
        display: flex;
        align-items: stretch;
        background-color: #f1f1f1;
    }
    .flex-container > div {
        background-color: DodgerBlue;
        color: white;
        width: 100px;
        margin: 10px;
        text-align: center;
        line-height: 75px;
        font-size: 30px;
    }
</style>
</head>
<body>
    <h1>The flex-basis Property</h1>
    <p>Set the initial length of the third flex item to 200 pixels:</p>
    <div class="flex-container">
        <div>1</div>
        <div>2</div>
        <div style="flex-basis:200px">3</div>
        <div>4</div>
    </div>
</body>
</html>
```

CSS Flexbox Child Elements cont.

The flex Property

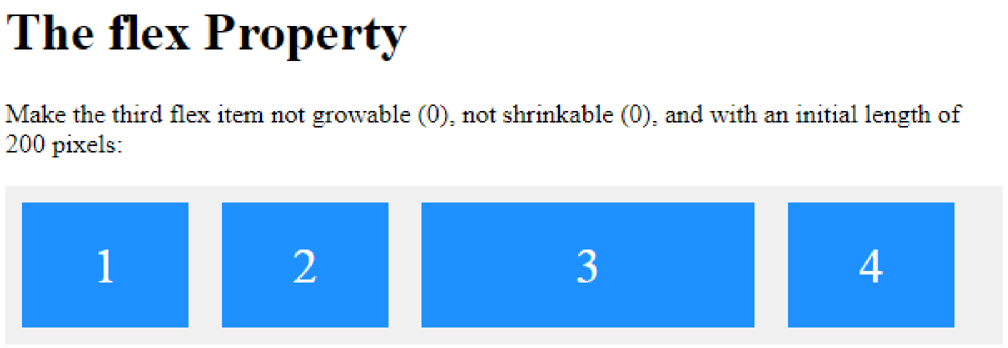
- It is a shorthand to set values to all these three properties at once. Using the ***flex*** property, you can set values to ***flex-grow***, ***flex-shrink***, and ***flex-basis*** values at once.
- Here is the syntax of this property.

```
.item {
      flex: none | [ <'flex-grow'> <'flex-shrink'>? || <'flex-basis'> ]
}
```

- Let’s look in to an example.

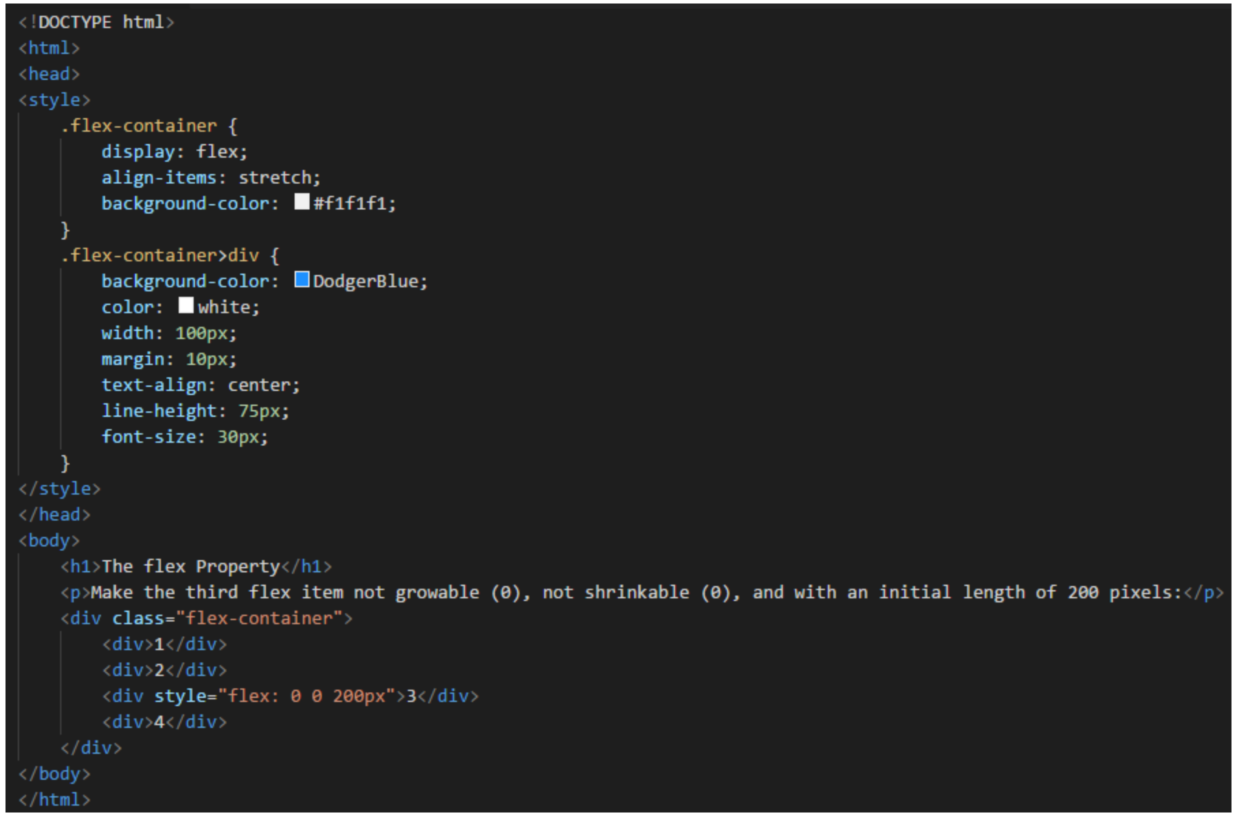
CSS Flexbox Child Elements cont.

- The example demonstrates the usage of the ***flex*** shorthand property.
- Here we are making the third flex item not growable (0), not shrinkable (0), and with an initial length of 200 pixels:



The flex Property

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



```
<!DOCTYPE html>
<html>
<head>
<style>
    .flex-container {
        display: flex;
        align-items: stretch;
        background-color: #f1f1f1;
    }
    .flex-container>div {
        background-color: DodgerBlue;
        color: white;
        width: 100px;
        margin: 10px;
        text-align: center;
        line-height: 75px;
        font-size: 30px;
    }
</style>
</head>
<body>
    <h1>The flex Property</h1>
    <p>Make the third flex item not growable (0), not shrinkable (0), and with an initial length of 200 pixels:</p>
    <div class="flex-container">
        <div>1</div>
        <div>2</div>
        <div style="flex: 0 0 200px">3</div>
        <div>4</div>
    </div>
</body>
</html>
```

CSS Flexbox Align Self

The align-self Property

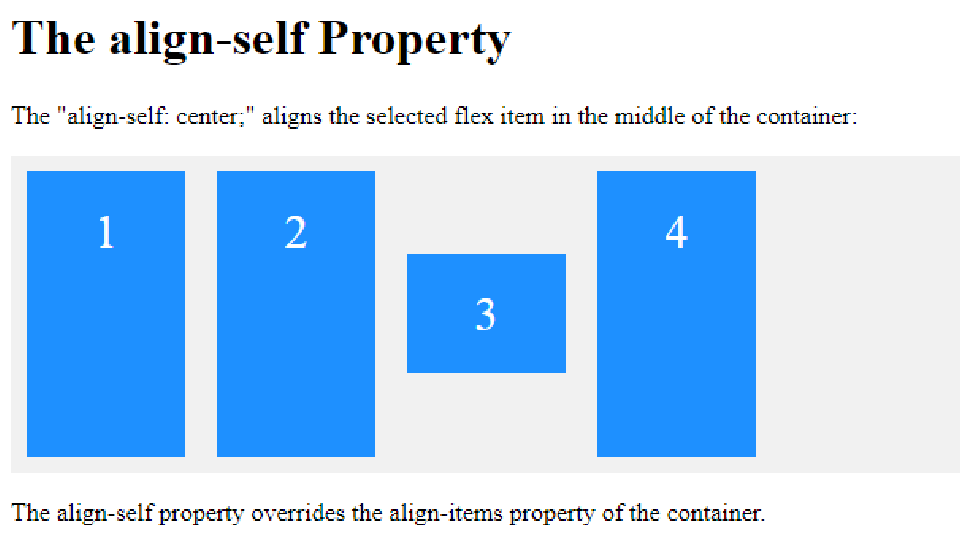
- The ***align-self*** property specifies the alignment for the selected item inside the flexible container.
- It overrides the default alignment set by the container's align-items property.
- Syntax is

  align-self: auto | flex-start | flex-end | center | baseline | stretch;

- This property accepts the following values:
  - **flex-start** − The flex item will be aligned vertically at the top of the container.
  - **flex-end** − The flex item will be aligned vertically at the bottom of the container.
  - **flex-center** − The flex item will be aligned vertically at the center of the container.

CSS Flexbox Align Self

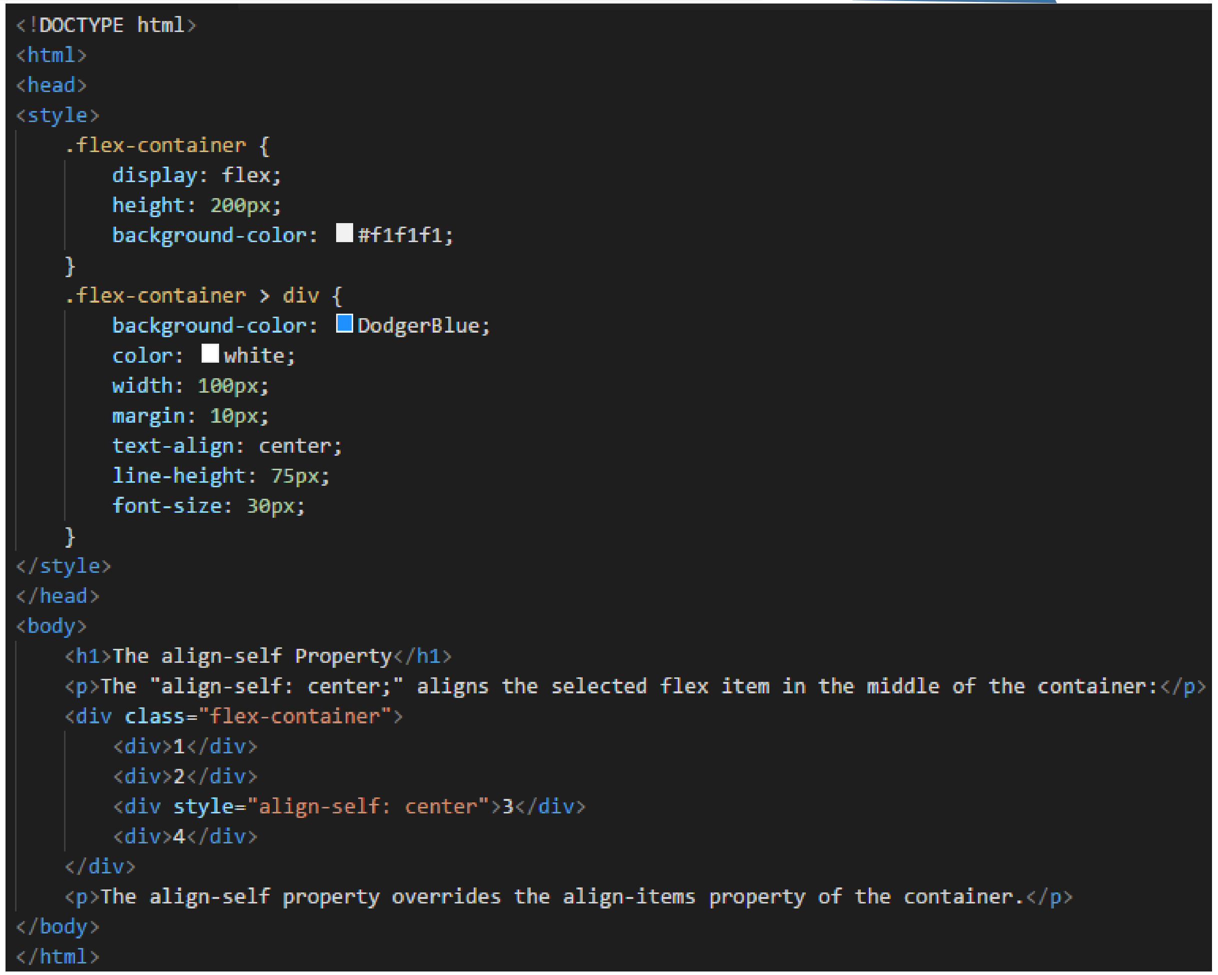
- The example demonstrates the usage of the ***align-self: center*** property.
- The "***align-self: center***;" aligns the selected flex item in the middle of the container.



The align-self Property

The "align-self: center;" aligns the selected flex item in the middle of the container:

The align-self property overrides the align-items property of the container.

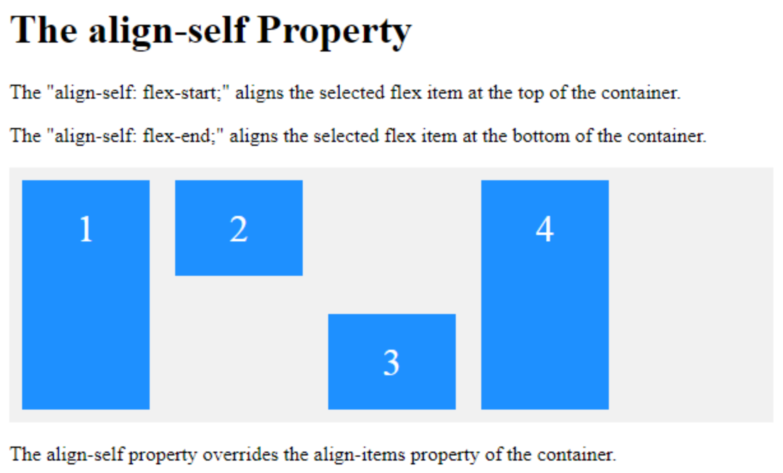


```
<!DOCTYPE html>
<html>
<head>
<style>
    .flex-container {
        display: flex;
        height: 200px;
        background-color: #f1f1f1;
    }
    .flex-container > div {
        background-color: DodgerBlue;
        color: white;
        width: 100px;
        margin: 10px;
        text-align: center;
        line-height: 75px;
        font-size: 30px;
    }
</style>
</head>
<body>
    <h1>The align-self Property</h1>
    <p>The "align-self: center;" aligns the selected flex item in the middle of the container:</p>
    <div class="flex-container">
        <div>1</div>
        <div>2</div>
        <div style="align-self: center">3</div>
        <div>4</div>
    </div>
    <p>The align-self property overrides the align-items property of the container.</p>
</body>
</html>
```

CSS Flexbox Align Self

align-self: flex-start & align-self: flex-end

- The "***align-self: flex-start***;" aligns the selected flex item at the top of the container.
- The "***align-self: flex-end***;" aligns the selected flex item at the bottom of the container.
- Let's see an example

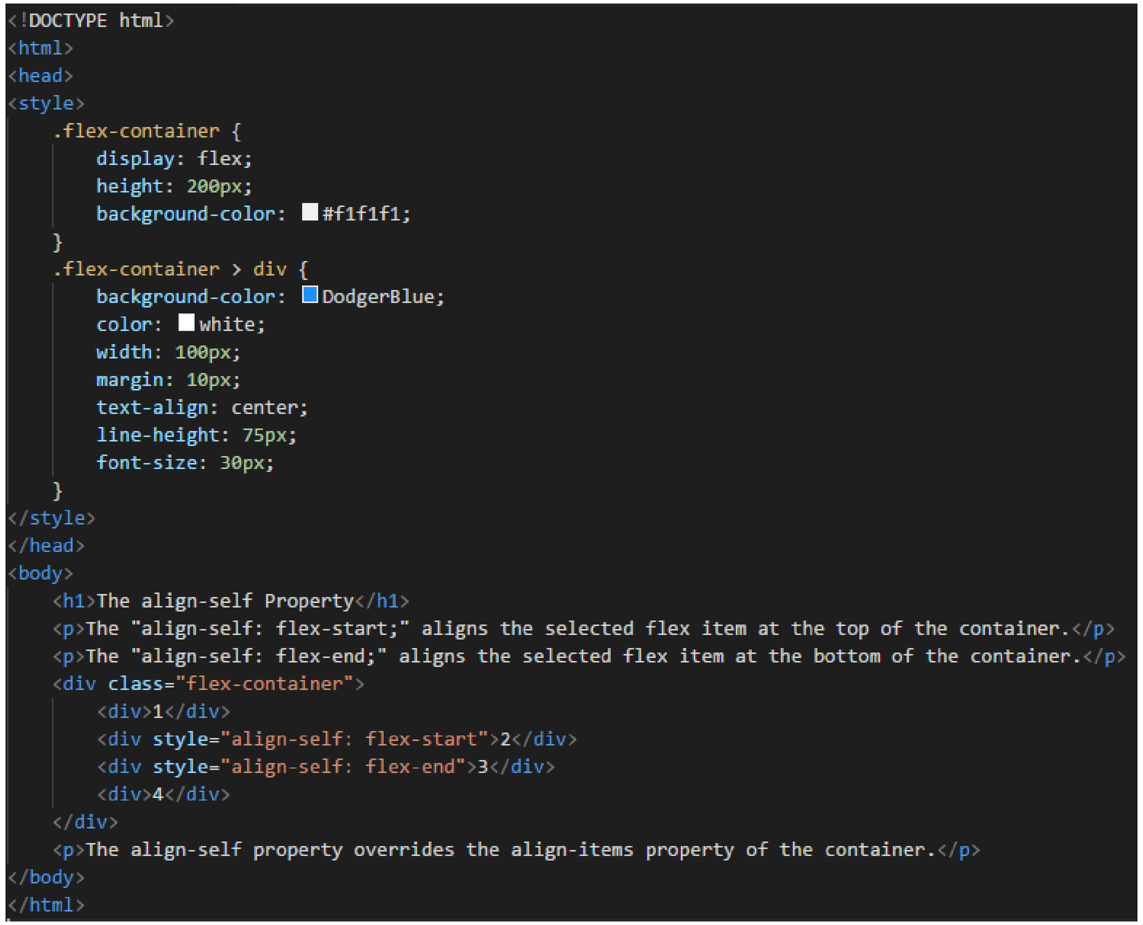


The align-self Property

The "align-self: flex-start;" aligns the selected flex item at the top of the container.

The "align-self: flex-end;" aligns the selected flex item at the bottom of the container.

The align-self property overrides the align-items property of the container.



```
<!DOCTYPE html>
<html>
<head>
<style>
    .flex-container {
        display: flex;
        height: 200px;
        background-color: #f1f1f1;
    }
    .flex-container > div {
        background-color: DodgerBlue;
        color: white;
        width: 100px;
        margin: 10px;
        text-align: center;
        line-height: 75px;
        font-size: 30px;
    }
</style>
</head>
<body>
    <h1>The align-self Property</h1>
    <p>The "align-self: flex-start;" aligns the selected flex item at the top of the container.</p>
    <p>The "align-self: flex-end;" aligns the selected flex item at the bottom of the container.</p>
    <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>
    <p>The align-self property overrides the align-items property of the container.</p>
</body>
</html>
```