# CSS

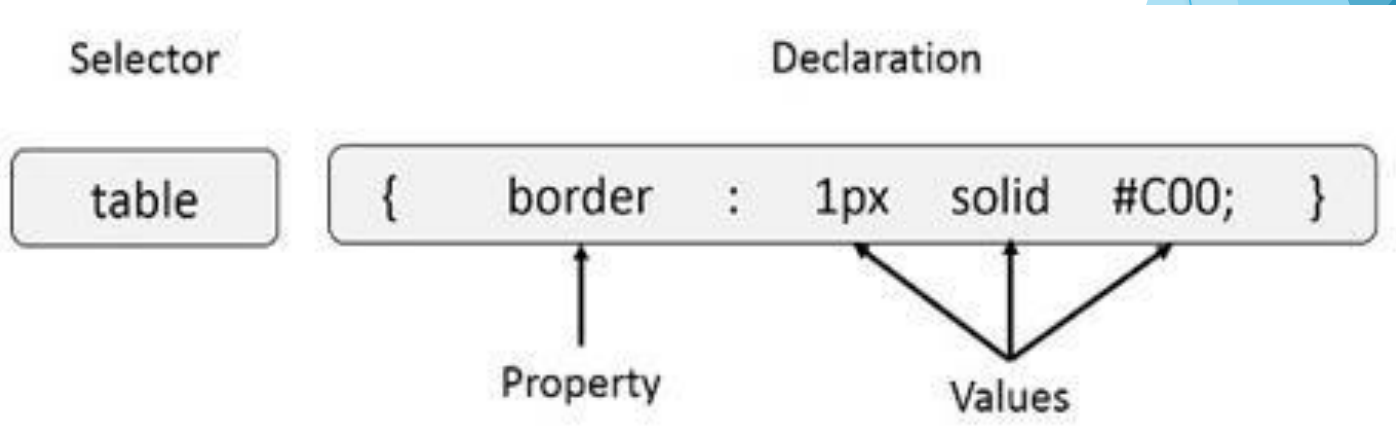
# CSS

CSS stands for **C**ascading **S**tyle **S**heet .

CSS is used to define styles for your web pages, including the design, layout for different devices and screen sizes.

CSS Syntax:

selector { property : value}



# CSS - How to use?

```
<head>
    <link rel="stylesheet" href="style.css">
</head>
```

---

```
<style>
    h1{color: red;}
    p{font-size: 15px;}
</style>
```

---

```
<h1 style="color: red;">Heading 1</h1>
<p style="font-size: 15px;">This is some text in a paragraph.</p>
```

# CSS - Selector Types

## Element Selector

```
h2 {
    color: #c70039 ;
}
```

## Universal Selector

```
* {
    color: #c70039 ;
}
```

## ID Selector

```
#content {
    color: #6E4253;
    font-size: 15px;
}
```

## Class Selector

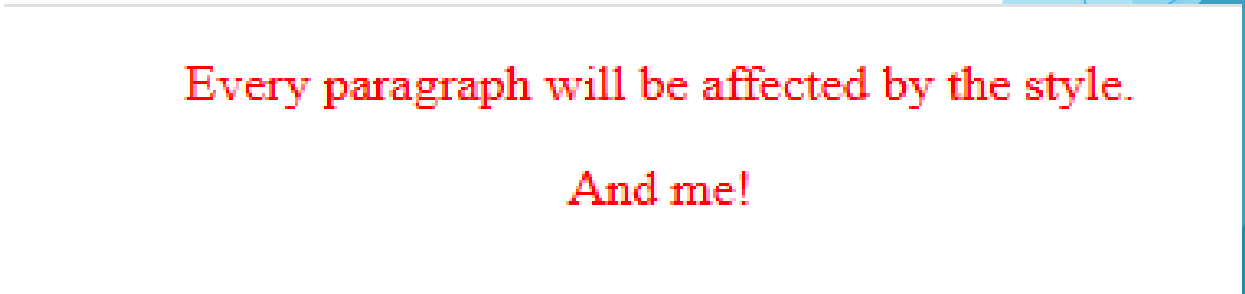
```
.main {
    margin-top: 10px
    margin-bottom: 10px
}
```

# CSS - Selector Types

- **Element Selector**

selects HTML elements based on the element name.

```
<style>
p { text-align: center;
    color: red; }
</style>

<p>Every paragraph will be affected by the style.</p>
<p>And me!</p>
```



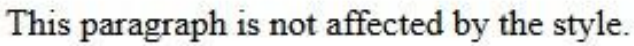
- **id Selector**

id selector uses the id attribute of an HTML element to select a specific element.

The id of an element is unique within a page, so the id selector is used to select one unique element!

To select an element with a specific id, write a hash (#) character, followed by the id of the element.

```
<style>
#para1 { text-align: center;
    color: red; }
</style>

<p id="para1">Hello World!</p>
<p>This paragraph is not affected by the style.</p>
```

- **Class Selector**

The class selector selects HTML elements with a specific class attribute.

To select elements with a specific class, write a period (.) character, followed by the class name.

```
<style>
.center {
  text-align: center;
   color: blue;
}
</style>

<h1 class="center">Red and center-aligned heading</h1>
<p class="center">Red and center-aligned paragraph.</p>
```

- **Class Selector**

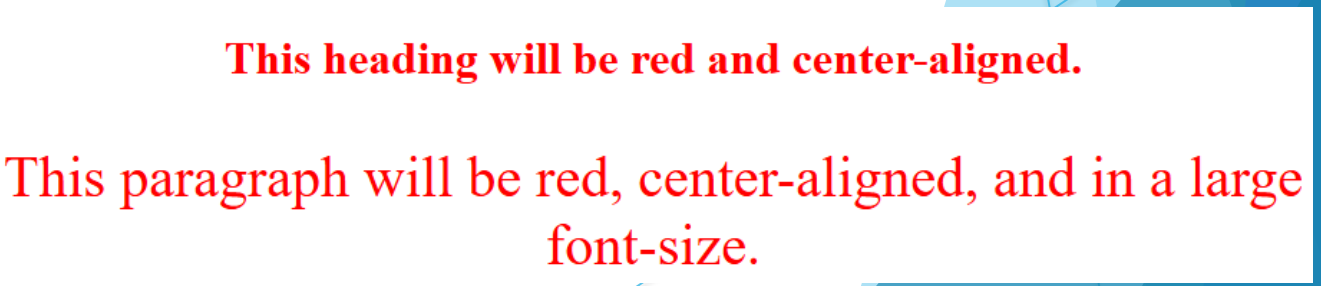
You can also specify that only specific HTML elements should be affected by a class.

```
<style>
p.center {
  text-align: center;
  color: red;
}
</style>

<h1 class="center">This heading will not be affected</h1>
<p class="center">This paragraph will be red and center-aligned.</p>
```

- **Class Selector**

HTML elements can also refer to more than one class.

```
<style>
.center {
  text-align: center;
  color: red;
}
p.large { font-size: 40px; }
</style>

<h1 class="center">This heading will be red and center-aligned.</h1>
<p class="center large">This paragraph will be red, center-aligned, and in a large font-size.</p>
```

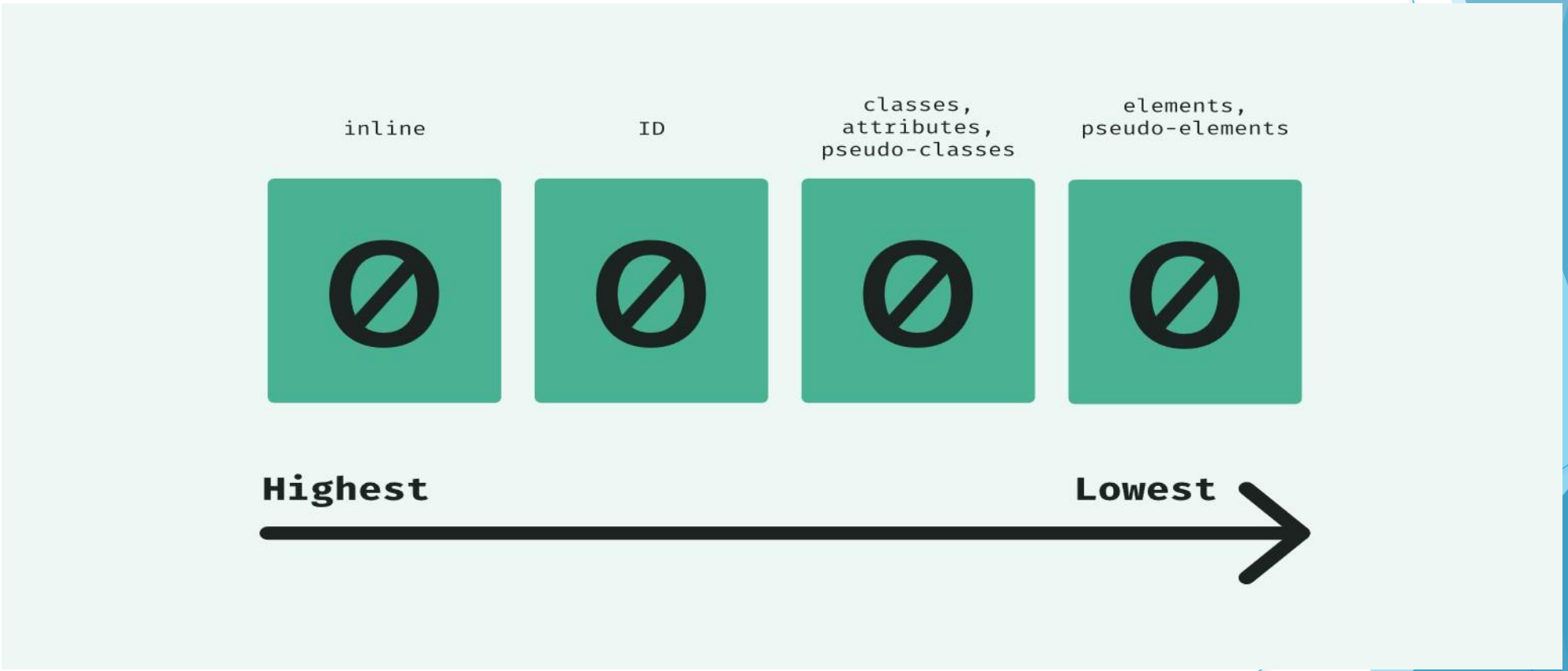


- **Universal Selector**

The universal selector (*) selects all HTML elements on the page.

```
<style>
* {
  text-align: center;
  color: blue;
}
</style>

<h1>Hello world!</h1>
<p>Every element on the page will be affected by the style.</p>
<p id="para1">Me too!</p>
```

# Specificity



- **Grouping Selector**

selects all the HTML elements with the same style definitions.

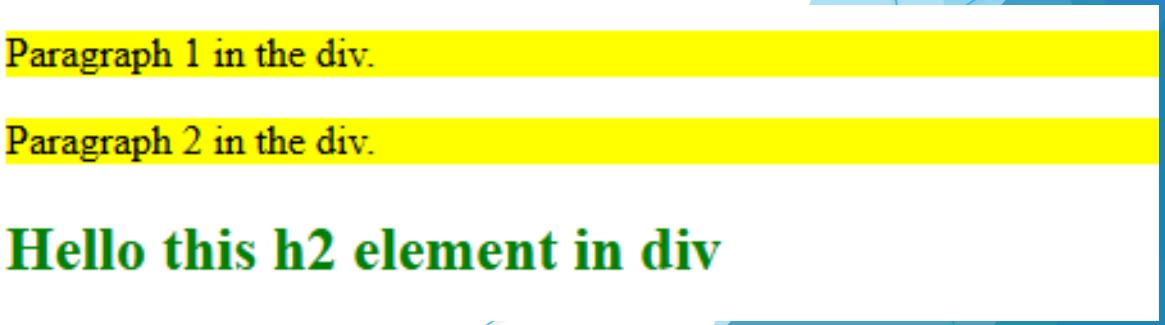
To group selectors, separate each selector with a comma (,).

```
<style>
h1, h2, p {
  text-align: center;
  color: green;
}
</style>

<h1>Hello World!</h1>
<h2>Smaller heading!</h2>
<p>This is a paragraph.</p>
```

- **Descendant Selector**

matches all elements that are descendants of a specified element.

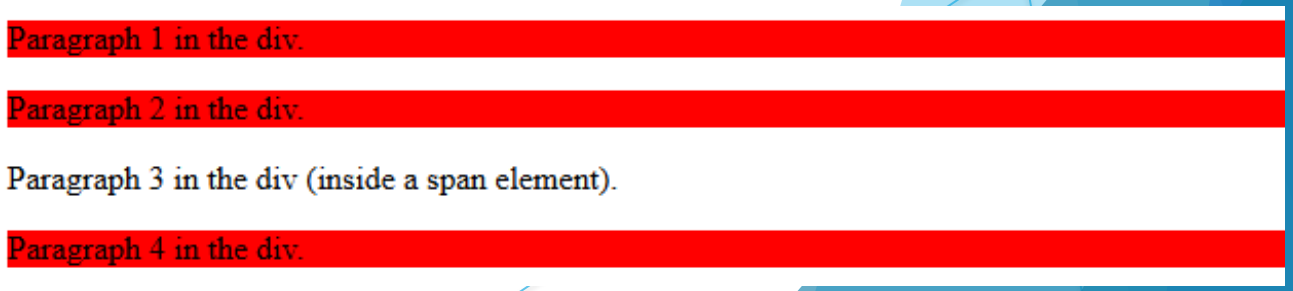
```
<style>
div p {
  background-color: yellow;
}
div h2 {
  color:green;
}
</style>

<div>
  <p>Paragraph 1 in the div.</p>
  <p>Paragraph 2 in the div.</p>
  <h2>Hello this h2 element in div</h2>
</div>
```



- **Child Selector**

child selector (>) selects all elements that are the children of a specified element.

```
<style>
div > p {
  background-color: red;
}
</style>

<div>
  <p>Paragraph 1 in the div.</p>
  <p>Paragraph 2 in the div.</p>
  <span>
    <!-- not Child but Descendant -->
    <p>Paragraph 3 in the div (inside a span element).</p>
  </span>
  <p>Paragraph 4 in the div.</p>
</div>
```



- **[attribute] Selector**

is used to select elements with a specified attribute.

```
<style>
a[target] {
  background-color: yellow;
}
</style>

<a href="https://www.w3schools.com">w3schools.com</a>
<a href="http://www.disney.com" target="_blank">disney.com</a>
```

w3schools.com disney.com

- **[attribute="value"] Selector**

is used to select elements with a specified attribute and value.

```
<style>

a[target="_blank"] { background-
color: red;
}
</style>

<a href="https://www.w3schools.com">w3schools.com</a>
<a href="http://www.disney.com" target="_blank">disney.com</a>
<a href="http://www.wikipedia.org" target="_top">wikipedia.org</a>
```

w3schools.com disney.com wikipedia.org

- **pseudo-class selector**

is used to define a special state of an element, such as :hover is used to style an element when hovered.

Syntax:

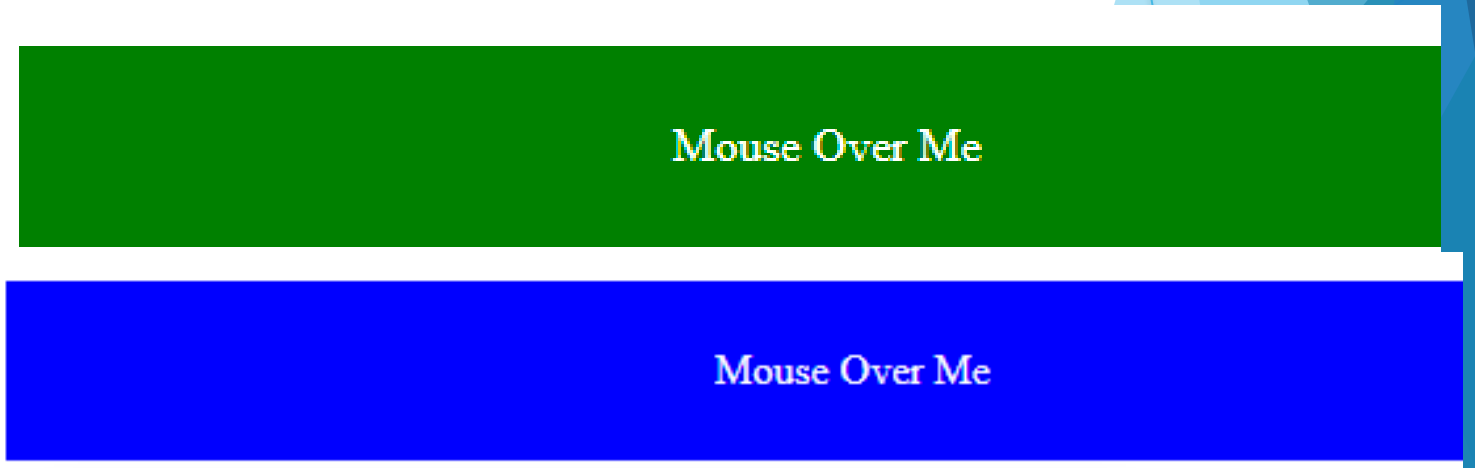
```
selector: pseudo-class{
    property: value;
}
```

more about pseudo-class visit link :

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

- **pseudo-class selector**

```
 <style>
div {
  background-color: green;
  color: white;
  padding: 25px;
  text-align: center;
}
div:hover {
  background-color: blue;
}
</style>

<div>Mouse Over Me</div>
```



- **Pseudo-Element selector**

is used to style specific parts of an element,Style the first letter or line, of an element .Insert content before or after an element

Syntax:

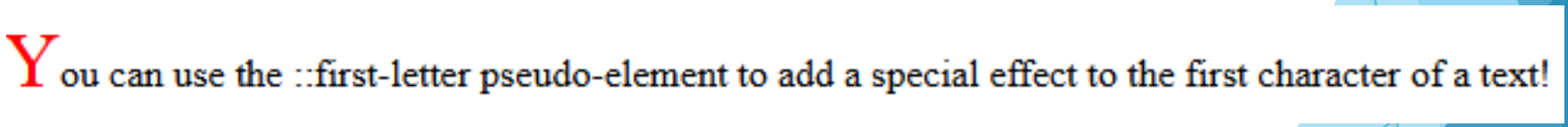
```
selector::pseudo-element {
  property: value;
}
```

more about pseudo-elements visit link :

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

- **pseudo-element selector**

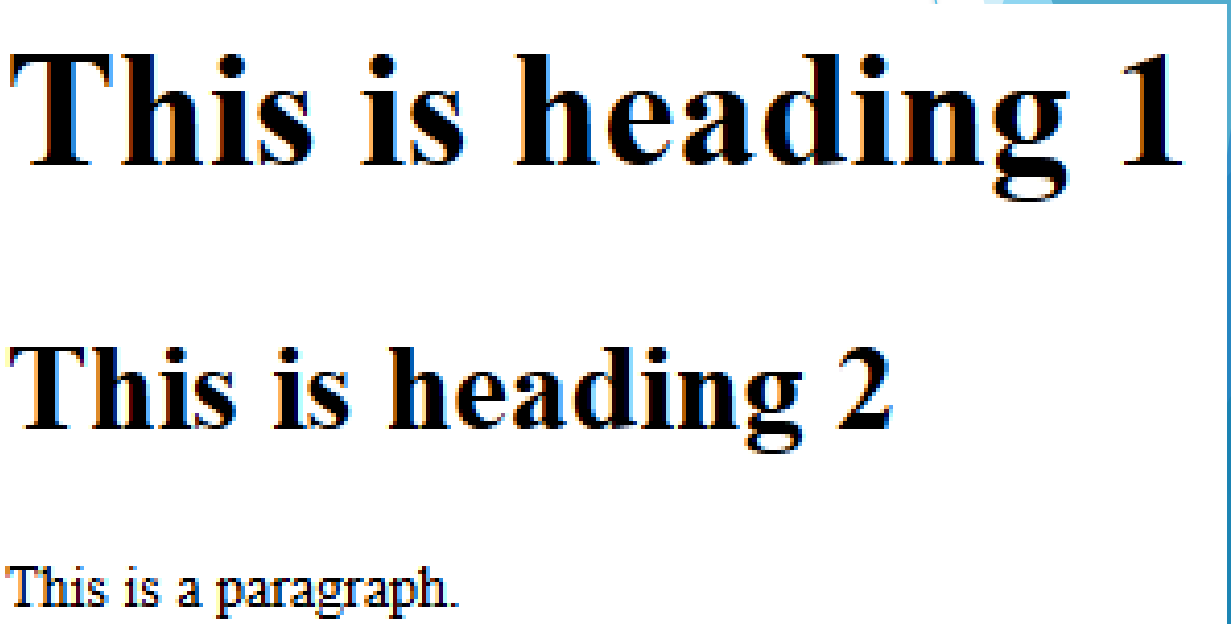
```
 <style>
p::first-letter {
 color: #ff0000;
 font-size: xx-large;
}
</style>

<p>You can use the ::first-letter pseudo-element to add a special effect to the first character of a text!</p>
```



# CSS - font-size

font-size property sets the size of the text.However, you should not use font size adjustments to make paragraphs look like headings, or headings look like paragraphs.

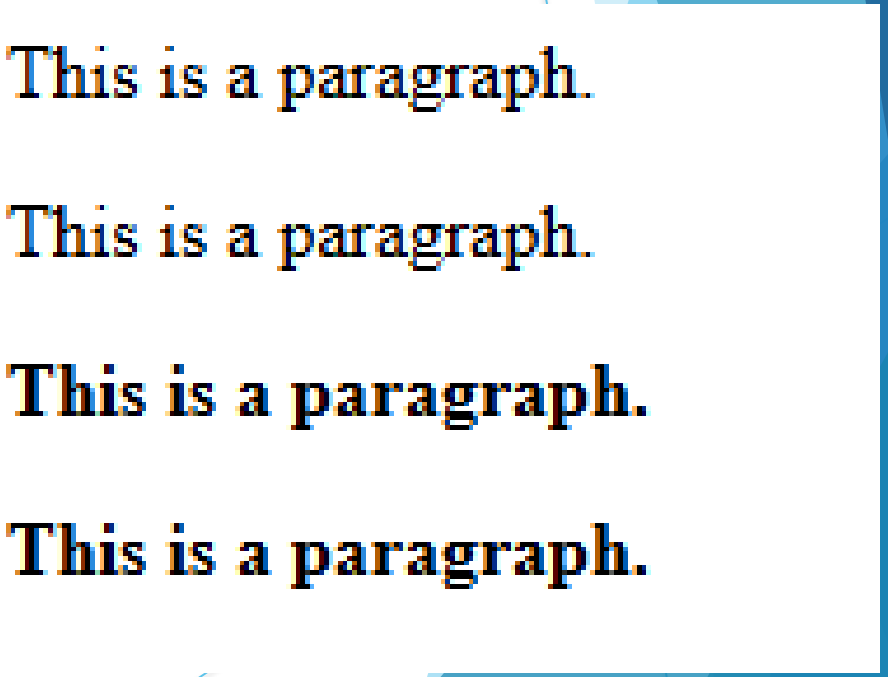
```
<style>
h1 { font-size: 40px; }
h2 { font-size: 30px; }
p { font-size: 14px; }
</style>

<h1>This is heading 1</h1>
<h2>This is heading 2</h2>
<p>This is a paragraph.</p>
```



# CSS - Font Weight

font-weight property specifies the weight of a font.

```
<style>
p.normal {font-weight: normal;}
p.light { font-weight: lighter;}
p.thick {font-weight: bold;}
p.thicker {font-weight: 900;}
</style>

<h1>The font-weight property</h1>
<p class="normal">This is a paragraph.</p>
<p class="light">This is a paragraph.</p>
<p class="thick">This is a paragraph.</p>
<p class="thicker">This is a paragraph.</p>
```

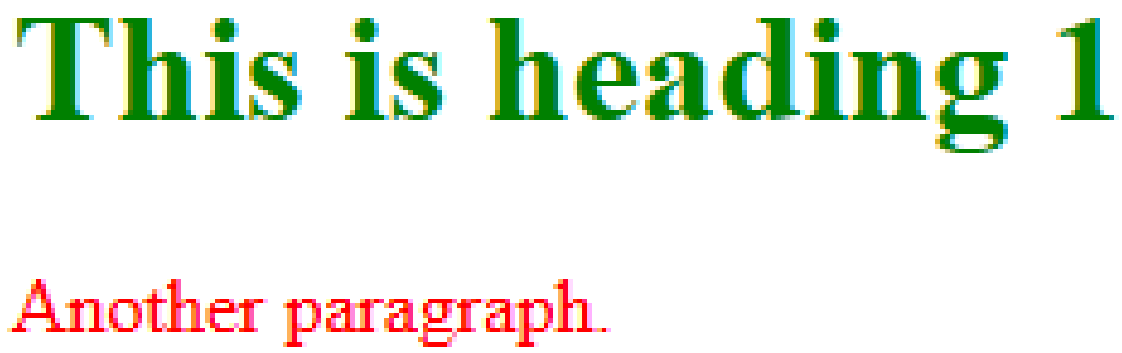


# CSS - Text Color

The color property is used to set the color of the text. The color is specified by:

- a color name - like "red"
- a HEX value - like "#ff0000"
- an RGB value - like "rgb(255,0,0)"

```
<style>
p {color: #ff0000;}
h1 { color: green;}
</style>

<h1>This is heading 1</h1>
<p>Another paragraph.</p>
```



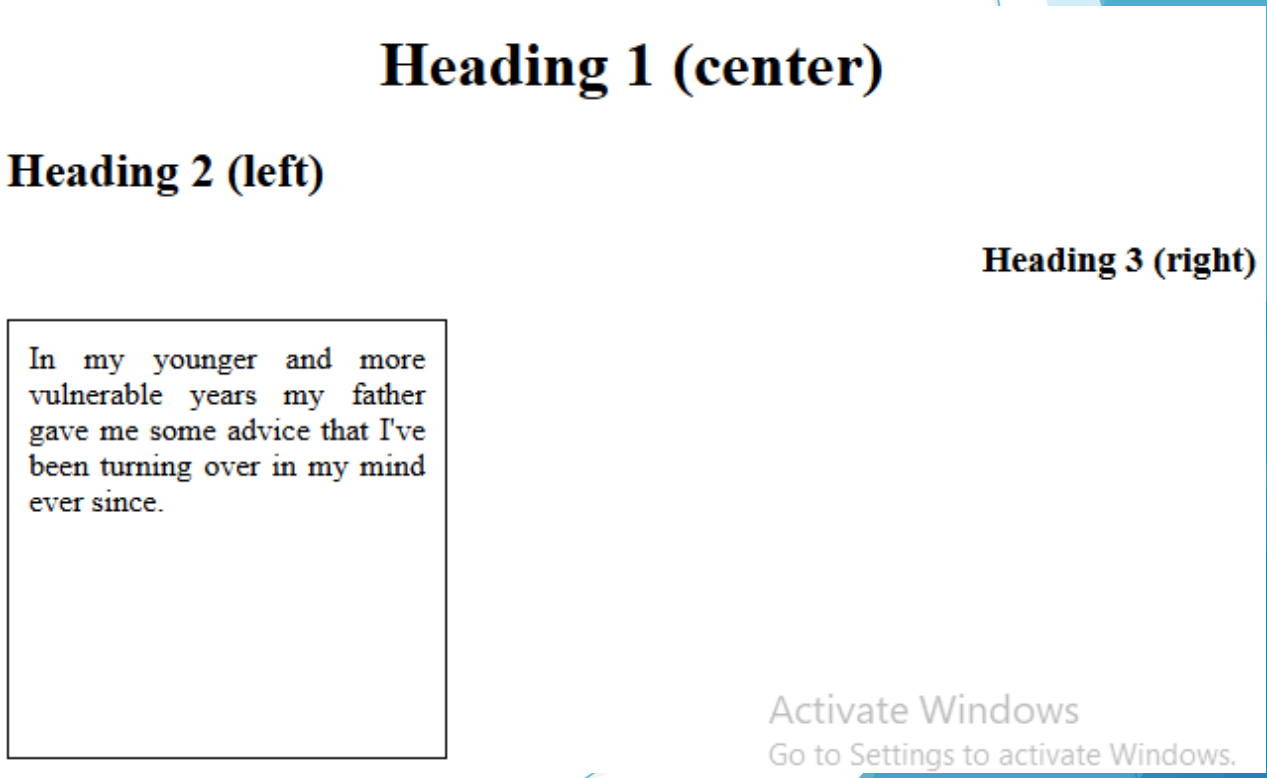
# CSS - Text Alignment

The text-align property is used to set the horizontal alignment of a text.

A text can be left or right aligned, centered, or justified.

```
<style>
h1 {text-align: center;}
h2 {text-align: left;}
h3 { text-align: right;}
div {
  border: 1px solid black;
  padding: 10px;
  width: 200px;
  height: 200px;
  text-align: justify;
}
</style>
```

```
<h1>Heading 1 (center)</h1>
<h2>Heading 2 (left)</h2>
<h3>Heading 3 (right)</h3>

<div>
In my younger and more vulnerable years my father gave me some advice that I've been turning over in my mind ever since.
</div>
```



# CSS - Styling Images

```
<style>
img {
  border-radius: 8px;
  border: 2px solid #000;
  border-top-left-radius: 20%;
  width:300px;
  height:50%;
  opacity: 0.7;
}
//Flip an Image
img:hover {
  transform: scaleX(-1);
}
</style>

<img src="paris.jpg" alt="Paris" >
```

# CSS - Background

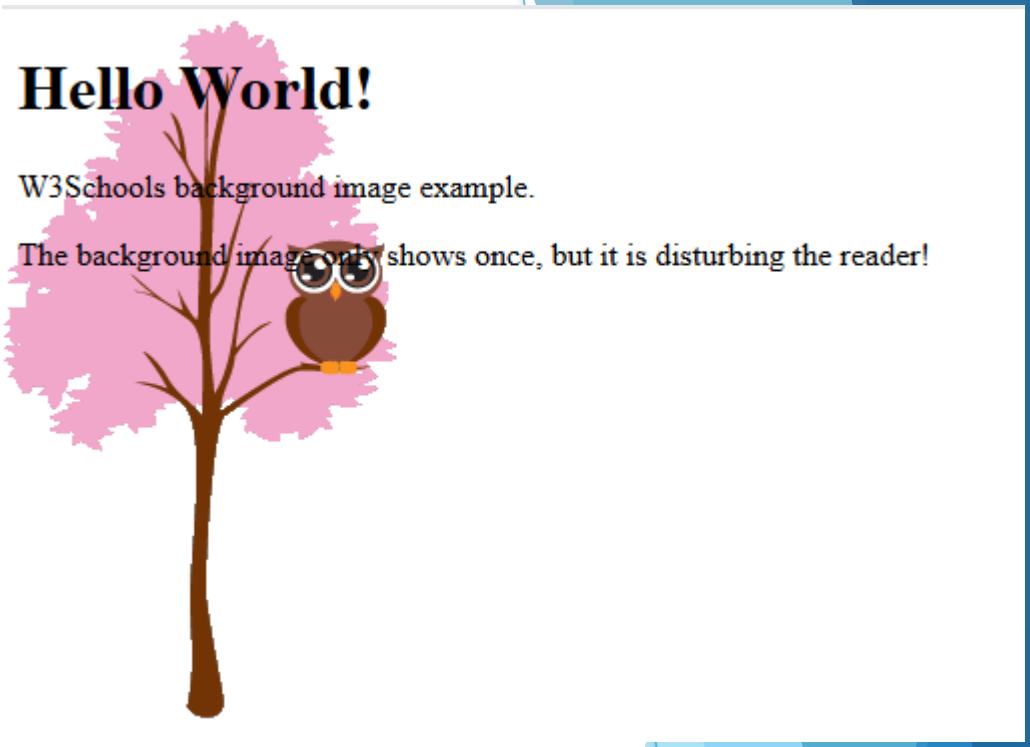
The background property of CSS is used to set the background of an element. It can be used to apply a single background image or multiple background images, as well as defining the background color, size, position, repeat behavior, and other related properties.

- **Background Image**

```
<style>
body {
  background-image: url("paper.gif");
}
</style>
</head>
<body>
<h1>Hello World!</h1>
<p>This page has an image as the background!</p>
</body>
```

- **Background-repeat**

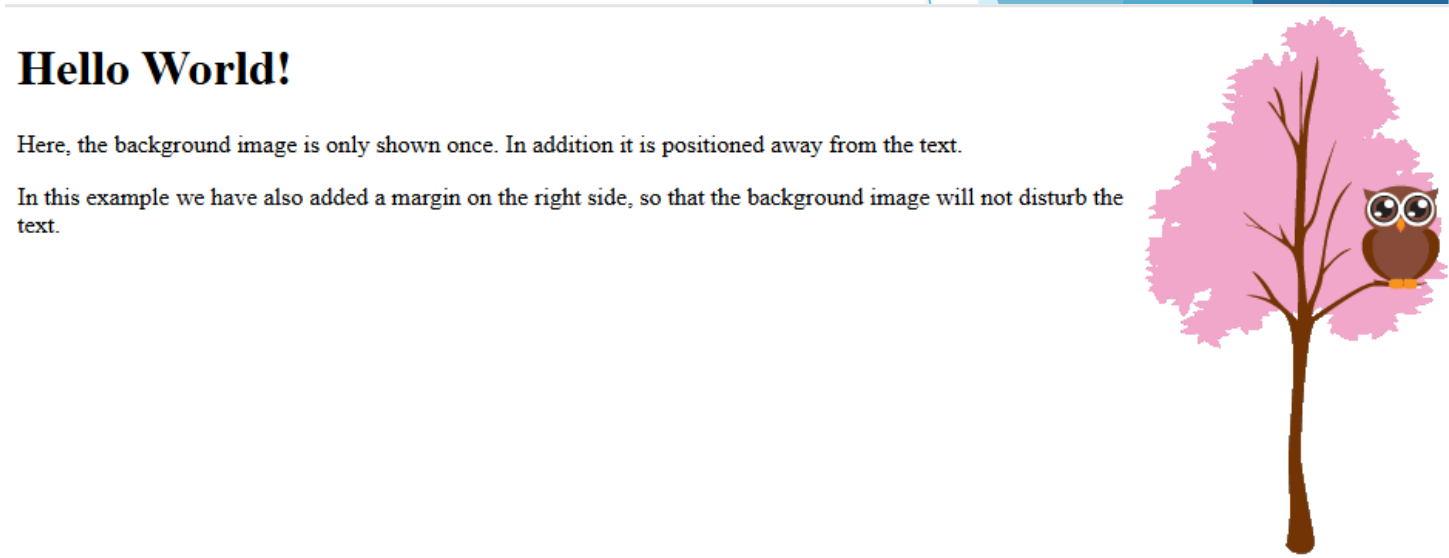
```
<style>
body {
  background-image: url("img_tree.png");
  background-repeat: no-repeat;
  //background-repeat: repeat-x;

}
</style>

<h1>Hello World!</h1>
<p>W3Schools background image example.</p>
<p>The background image only shows once, but it is disturbing the reader!</p>
```



- **background-position**

property is used to set the starting position of the background image.
By default, a background-image is placed at the top-left corner of an element.

```
<style>
body {
  background-image: url("img_tree.png");
  background-repeat: no-repeat;
  background-position: right top;
  margin-right: 200px;
}
</style>
```

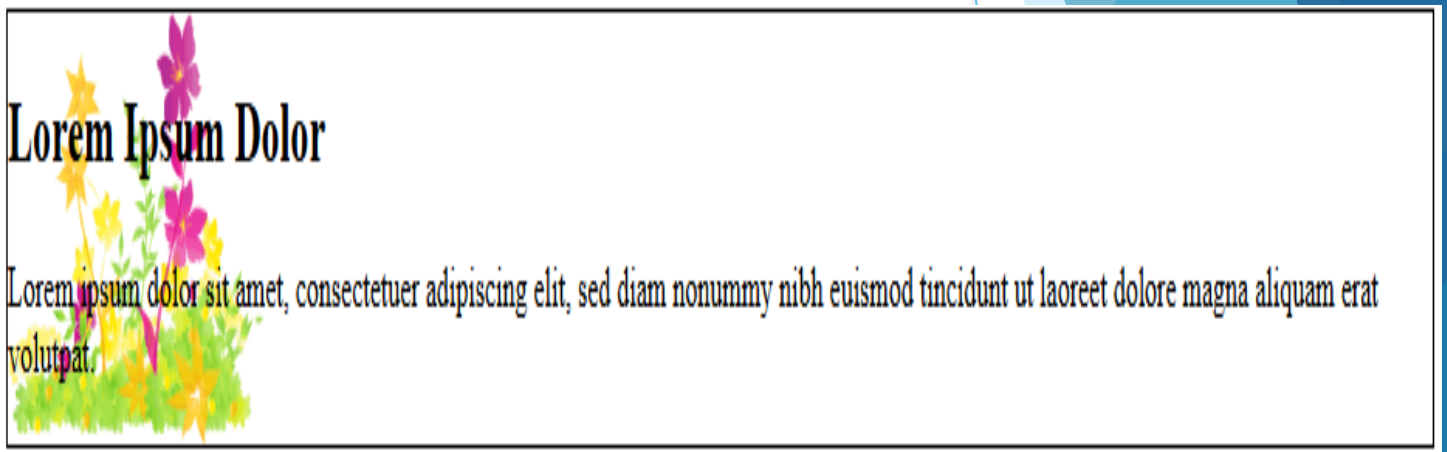



```
<h1>Hello World!</h1>
<p>Here, the background image is only shown once. In addition it is positioned away from the text.</p>
<p>In this example we have also added a margin on the right side, so that the background image will
not disturb the text.</p>
```

- **background-size**

property allows you to specify the size of background images.
The background size can be specified in lengths, percentages, or by using one of the keywords: auto, contain, or cover.

```
<style>
#div1 {
  border: 1px solid black;
  background-image: url(img_flwr.gif);
  background-repeat: no-repeat;
  background-size: contain;
//background-size: cover;
//background-size: auto;
}
<div id="div1">
  <h2>Lorem Ipsum Dolor</h2>
  <p>Lorem ipsum dolor sit amet, consectetuer adipiscing elit, sed diam nonummy nibh euismod
tincidunt ut laoreet dolore magna aliquam erat volutpat.</p>
</div>
```



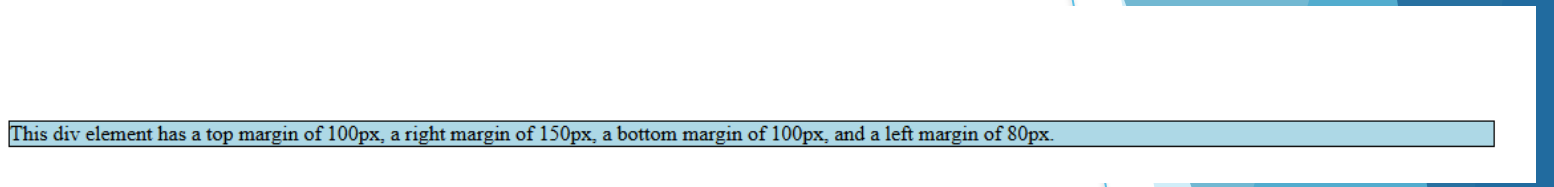
# CSS - Margins

Margins are used to create space around elements, outside of any defined borders.

| Property | Description |
| --- | --- |
| margin | a shorthand property that sets the margin properties in one declaration |
| margin-top | sets the top margin of the element |
| margin-right | sets the right margin of the element |
| margin-bottom | sets the bottom margin of the element |
| margin-left | sets the left margin of the element |

# CSS - Margins

```
<style>
div {
  border: 1px solid black;
  margin-top: 100px;
  margin-bottom: 100px;
  margin-right: 150px;
  margin-left: 80px;
  background-color: lightblue;
}
</style>

<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>
```



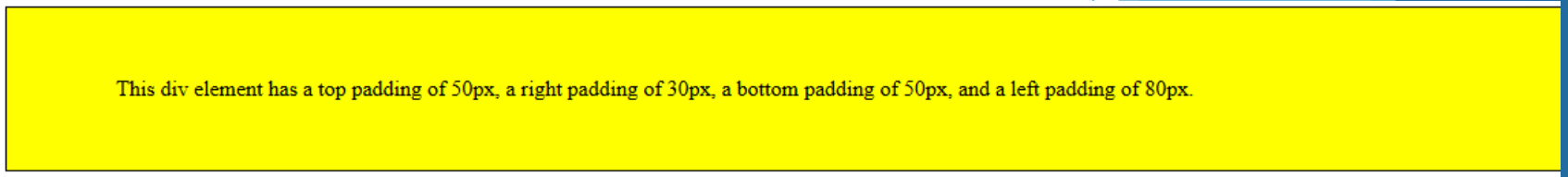
# CSS - Padding

Padding is used to create space around an element's content, inside of any defined borders.

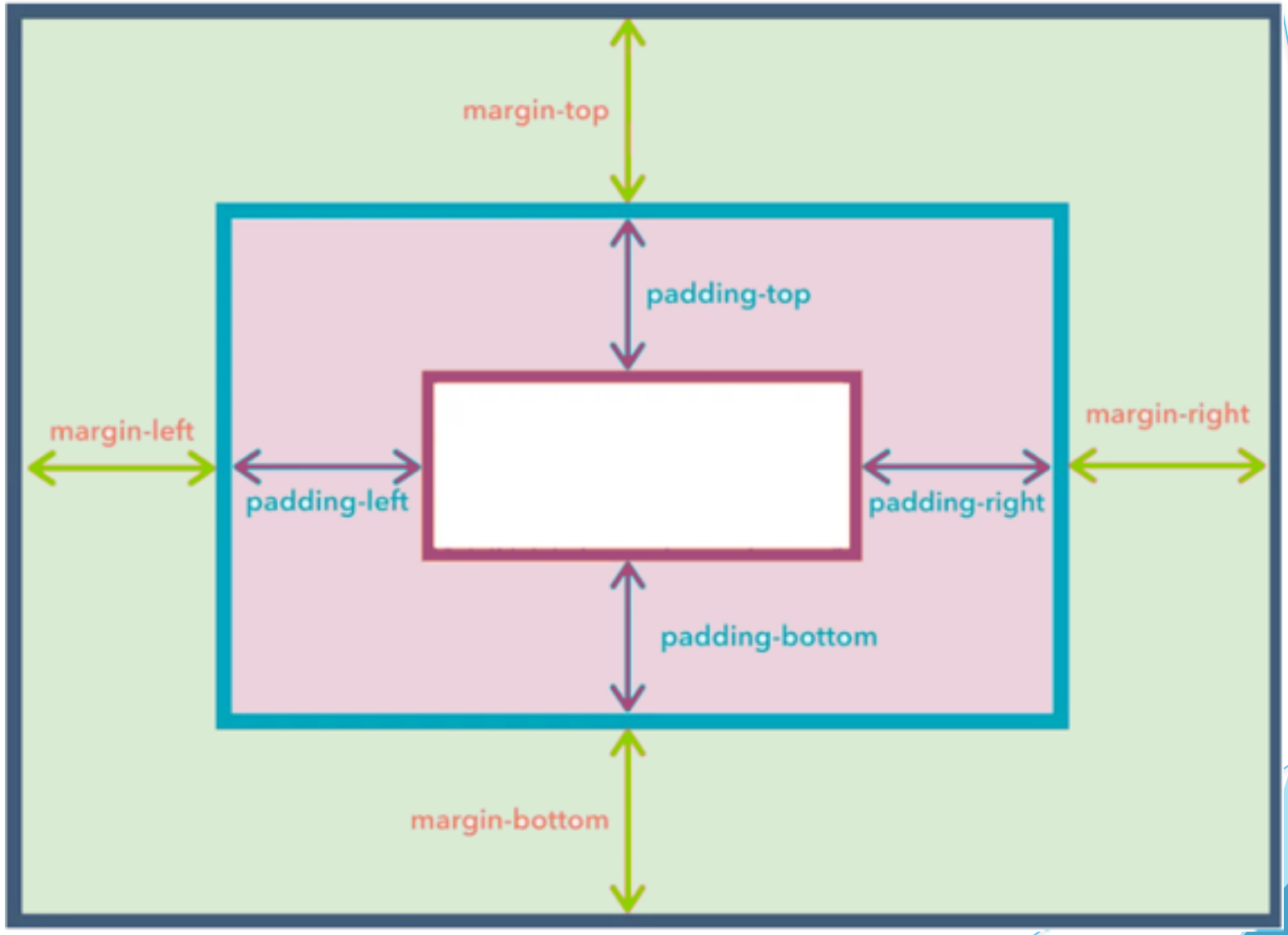
| Property | Description |
| --- | --- |
| padding | A shorthand property that is used for setting all the padding properties in one declaration. |
| padding-top | Sets the top padding of an element. |
| padding-right | Sets the right padding of an element. |
| padding-bottom | Sets the bottom padding of an element. |
| padding-left | Sets the left padding of an element. |

# CSS - Padding

```
<style>
div {
  border: 1px solid black;
  background-color: yellow;
  padding-top: 50px;
  padding-right: 30px;
  padding-bottom: 50px;
  padding-left: 80px;
}
</style>

<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>
```



# CSS – Box Model



# CSS – Float

- The CSS float property controls the positioning and formatting of content on the page.
- It positions an element on the right or left side of the container, letting text and other inline elements to wrap around it.
- Values:

  - **none**: The element does not float. This is the default value.
  - **left**: The element floats to the left of its container..
  - **right**: The element floats to the right of its container.

# Float-Example

```
<div>
<img src="pineapple.jpg" alt="Pineapple" width="300" height="300">

<p>Lorem ipsum dolor sit amet, consectetur adipiscing elit. Phasellus imperdiet, nulla et dictum interdum, nisi lorem egestas odio, vitae scelerisque enim ligula venenatis dolor. Maecenas nisl est, ultrices nec congue eget, auctor vitae massa. Fusce luctus vestibulum augue ut aliquet. Mauris ante ligula, facilisis sed ornare eu, lobortis in odio. Praesent convallis urna a lacus interdum ut hendrerit risus congue. </p>

</div>

img {
  float: right;
  margin-right: 15px;
  width: 200px;
  height: 200px;
}
```

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Phasellus imperdiet, nulla et dictum interdum, nisi lorem egestas odio, vitae scelerisque enim ligula venenatis dolor. Maecenas nisl est, ultrices nec congue eget, auctor vitae massa. Fusce luctus vestibulum augue ut aliquet. Mauris ante ligula, facilisis sed ornare eu, lobortis in odio.

# CSS - Positions

- **Position Static**

HTML elements are positioned static by default.

Static positioned elements are not affected by the top, bottom, left, and right properties.

```
<div class=”parent”>
<div class=”one”>one </div>
<div class=”two”>two</div>
<div class=”three”>three</div>
 </div>
```

```
.parent{
background-color: skyblue;
height:200px;
width:200px;
padding: 10px;
}
.parent div {height : 50px;}
.one{background-color: tomato;}
.two{background-color: gold;}
.three{background-color: darkviolet;}
```

# CSS - Positions

- **Position Static**

# CSS - Positions

- **Position Relative**

is positioned relative to its normal position.

Setting the top, right, bottom, and left properties of a relatively-positioned element will cause it to be adjusted away from its normal position.

```
.parent{
background-color: skyblue;
height:200px;  width:200px;
padding: 10px; }
.parent div {height : 50px;}
.one{background-color: tomato;}
.two{ background-color: gold; position:relative;
top:70px; left:40px; }
.three{background-color: darkviolet;}
```

# CSS - Positions

- **Position Relative**

# CSS - Positions

- **Position Fixed**

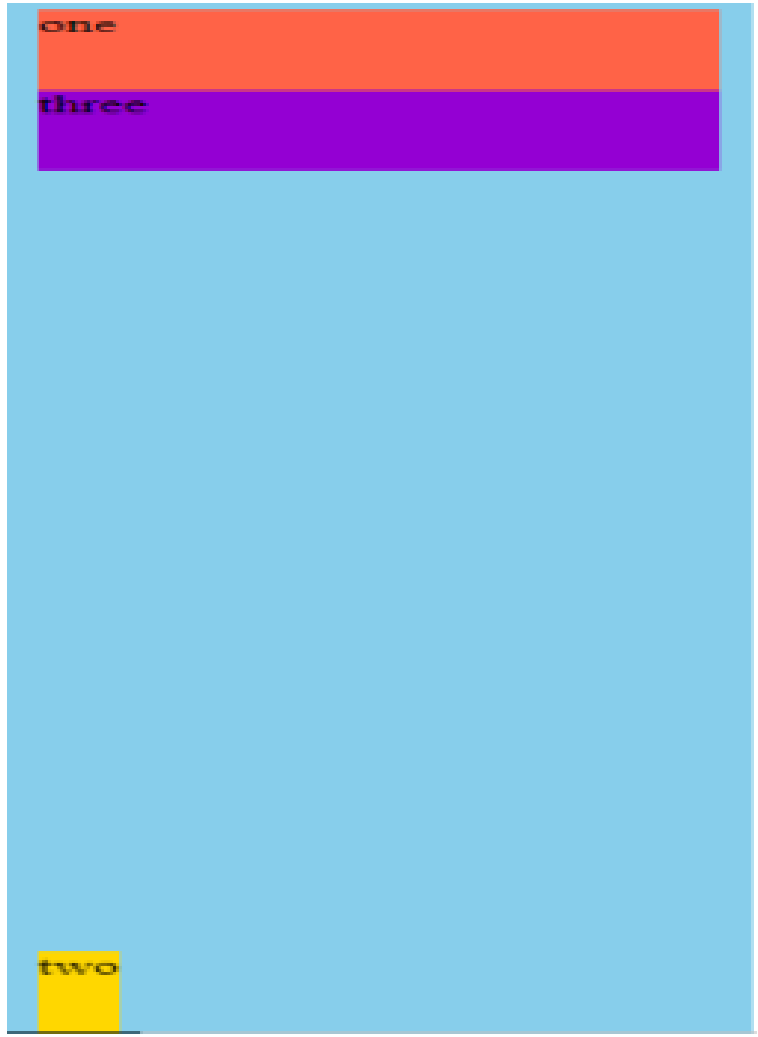
is positioned relative to the viewport, which means it always stays in the same place even if the page is scrolled.

A fixed element does not leave a gap in the page where it would normally have been located.

```
.parent{
background-color: skyblue;
height:200px;  width:200px;
padding: 10px; }
.parent div {height : 50px;}
.one{background-color: tomato;}
.two{ background-color: gold;
 position:fixed;  bottom:0; }
.three{background-color: darkviolet;}
```

# CSS - Positions

- **Position Fixed**



# CSS - Positions

- **Position Sticky**

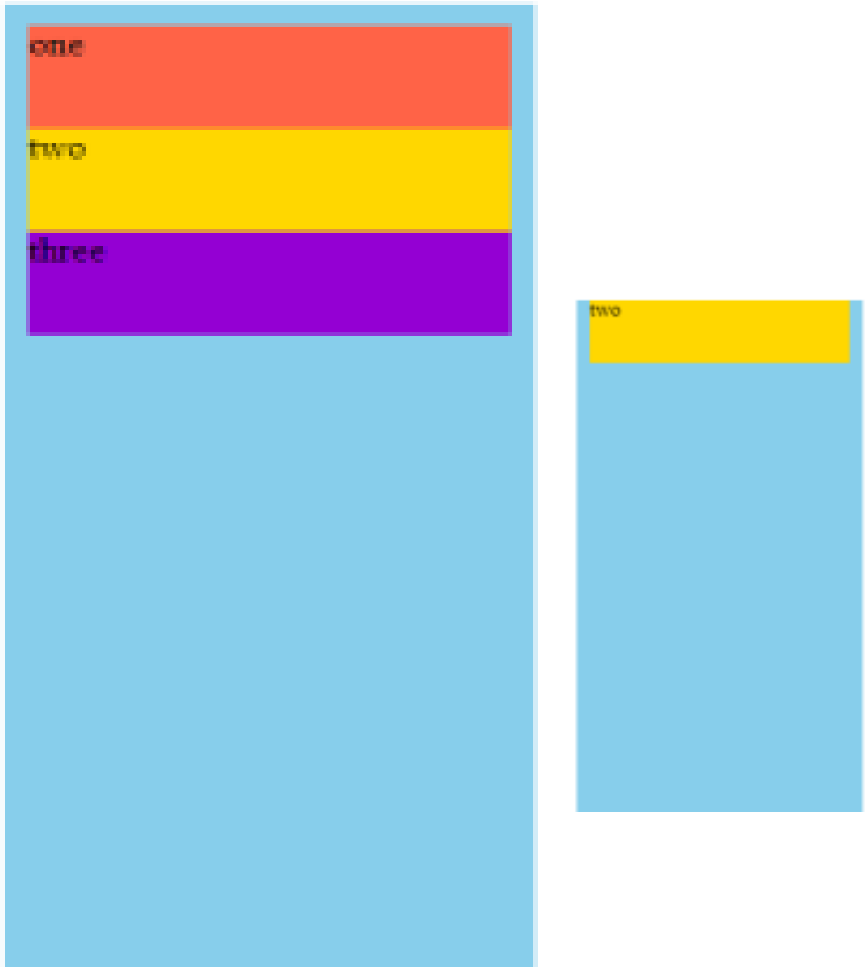
is positioned based on the user's scroll position.

It is positioned relative until a given offset position is met in the viewport - then it "sticks" in place (like position:fixed).

```
.parent{
background-color: skyblue;
height:200px;  width:200px;
padding: 10px; }
.parent div {height : 50px;}
.one{background-color: tomato;}
.two{ background-color: gold;
 position:sticky;  bottom:0; top:0; }
.three{background-color: darkviolet;}
```

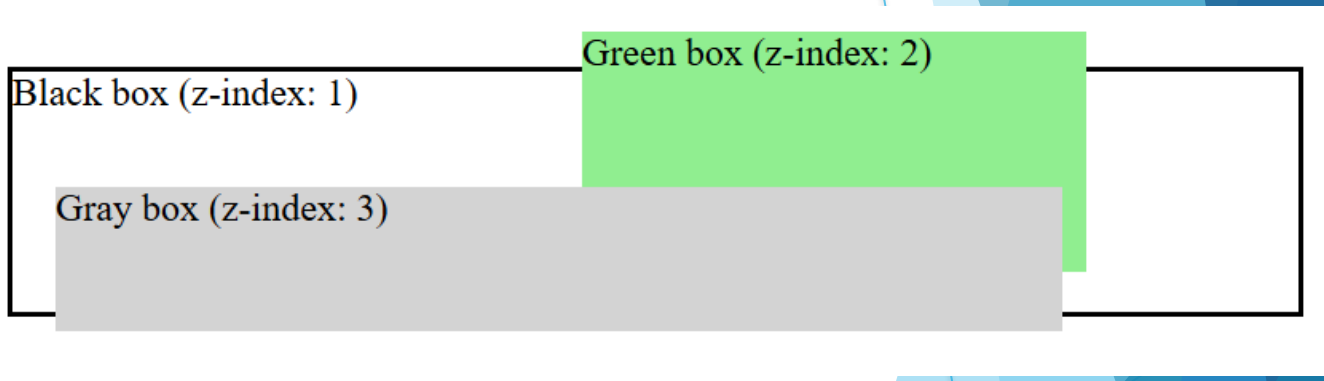
# CSS - Positions

- **Position Sticky**



# Z-Index

```
<div class="container">
  <div class="black-box">Black box (z-index: 1)</div>
  <div class="gray-box">Gray box (z-index: 3)</div>
  <div class="green-box">Green box (z-index: 2)</div>
</div>

.container { position: relative;}
.black-box {
  position: relative; z-index: 1;
  border: 2px solid black;
  height: 100px;  margin: 30px;
}
.gray-box {
  position: absolute;  z-index: 3;
  background: lightgray;  height: 60px;  width: 70%;
  left: 50px; top: 50px; }
.green-box {
  position: absolute; z-index: 2;
  background: lightgreen; width: 35%;
  left: 270px;  top: -15px;  height: 100px; }
```



# CSS - Display Elements

- **Display none**

is commonly used with JavaScript to hide and show elements without deleting and recreating them.

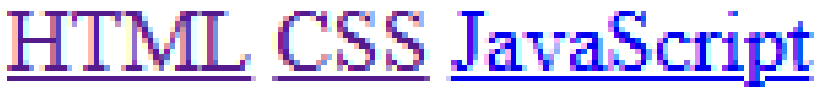
```
<style>
h1.hidden {
  display: none;
}
</style>
<h1>This is a visible heading</h1>
<h1 class="hidden">This is a hidden heading</h1>
<p>Notice that the h1 element with display: none; does not take up any space.</p>
```

# CSS - Display Elements

- **Display inline**

Changing an inline element to a block element, or vice versa, can be useful for making the page look a specific way

```
<style>
li {
  display: inline;
}
</style>
<ul>
  <li><a href="/html/default.asp" target="_blank">HTML</a></li>
  <li><a href="/css/default.asp" target="_blank">CSS</a></li>
  <li><a href="/js/default.asp" target="_blank">JavaScript</a></li>
</ul>
```



# CSS - Display Elements

- **Display block**

```
<style>
a {
  display: block;
}
</style>

<a href="/html/default.asp" target="_blank">HTML</a>
<a href="/css/default.asp" target="_blank">CSS</a>
<a href="/js/default.asp" target="_blank">JavaScript</a>
```

# CSS- Flex box

Flexbox is a one-dimensional layout system used to align and distribute space among items in a container.

**why we use it:**

- Easier alignment (center elements vertically & horizontally).
- Responsive layouts without float or positioning tricks.

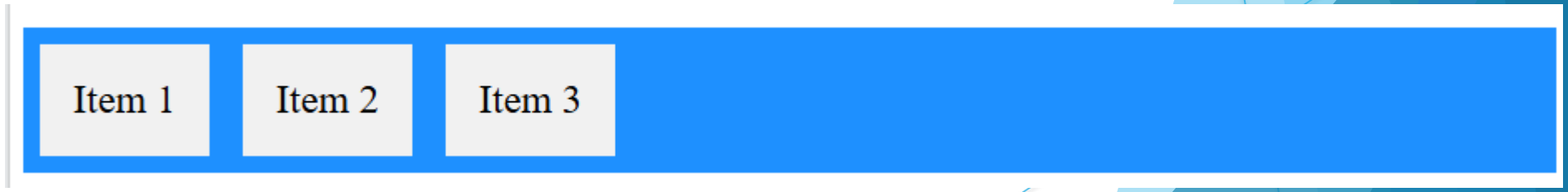
**A flexbox always consists of:**

- A Flex Container : The parent (container) element, where the display property is set to flex or inline-flex
- One or more Flex Items : The direct children of the flex container automatically becomes flex items

# CSS – Flex box

```
<style>
.container {
  display: flex;
  background-color: DodgerBlue; }
.container div {
  background-color: #f1f1f1;
  margin: 10px;
  padding: 20px;
  font-size: 24px; }
</style>
```

```
<h1>Create a Flex Container</h1>
<div class="container">
  <div>Item 1</div>
  <div>Item 2</div>
  <div>Item 3</div>
</div>
```



# CSS – Flex Container

**The flex container element can have the following properties:**

- **flex-direction** - Sets the display-direction of flex items
- **flex-wrap** - Specifies whether the flex items should wrap or not
- **justify-content** - Aligns the flex items when they do not use all available space on the main-axis (horizontally)
- **align-items** - Aligns the flex items when they do not use all available space on the cross-axis (vertically)
- **align-content** - Aligns the flex lines when there is extra space in the cross axis and flex items wrap

**flex-direction** - Sets the display-direction of flex items

This property can have one of the following **values**:

- row (default)
- column
- row-reverse
- column-reverse

```
<style>
.container {
 display: flex;     flex-direction: row;
 background-color: DodgerBlue; }
.container div {
 background-color: #f1f1f1;
 width: 100px;   margin: 10px;  padding: 10px;
 text-align: center;    font-size: 30px; }
</style>
```

```
<p>flex-direction: row; displays the flex items
horizontally (from left to right):</p>
<div class="container">
 <div>1</div>
 <div>2</div>
 <div>3</div>
</div>
```

**flex-wrap** - property specifies whether the flex items should wrap or not, if there is not enough room for them on one flex line.

This property can have one of the following **values**:

- nowrap (default)
- wrap
- wrap-reverse

```
<style>
.container {
 display: flex;    flex-wrap: nowrap;
 background-color: DodgerBlue;  }
.container div {
 background-color: #f1f1f1;
 width: 100px;   margin: 10px; padding: 10px;
 text-align: center;  font-size: 30px; }
</style>
```

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

**justify-content** - is used to align the flex items when they do not use all available space on the main-axis (horizontally).

This property can have one of the following **values**:

- center
- flex-start (default)
- flex-end
- space-around
- space-between
- space-evenly

```
<style>
.container {
  display: flex; justify-content: center;
  background-color: DodgerBlue;  }
.container div {
  background-color: #f1f1f1;
  width: 100px;   margin: 10px;   padding: 10px;
  text-align: center;  font-size: 30px; }
</style>
```

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

**align-items** is used to align the flex items when they do not use all available space on the cross-axis (vertically).

This property can have one of the following **values**:

- center
- flex-start
- flex-end
- stretch
- normal (default)

```
<style>
.container {
  display: flex;  height: 200px; align-items: center;
  background-color: DodgerBlue;  }
.container div {
  background-color: #f1f1f1;
  width: 100px;   margin: 10px;   padding: 10px;
  text-align: center;  font-size: 30px; }
</style>
```

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

**align-content** is similar to align-items, but instead of aligning the flex items, it aligns the flex lines.

This property can have one of the following **values**:

- center
- stretch (default)
- flex-start
- flex-end
- space-around
- space-between
- space-evenly

```
.container {
 display: flex;  height: 400px;  flex-wrap: wrap;
 align-content: center;  overflow: scroll;
 background-color: DodgerBlue; }
.container div {
 background-color: #f1f1f1;
 width: 100px;   margin: 10px;   padding: 10px;
 text-align: center;  font-size: 30px; }
```

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