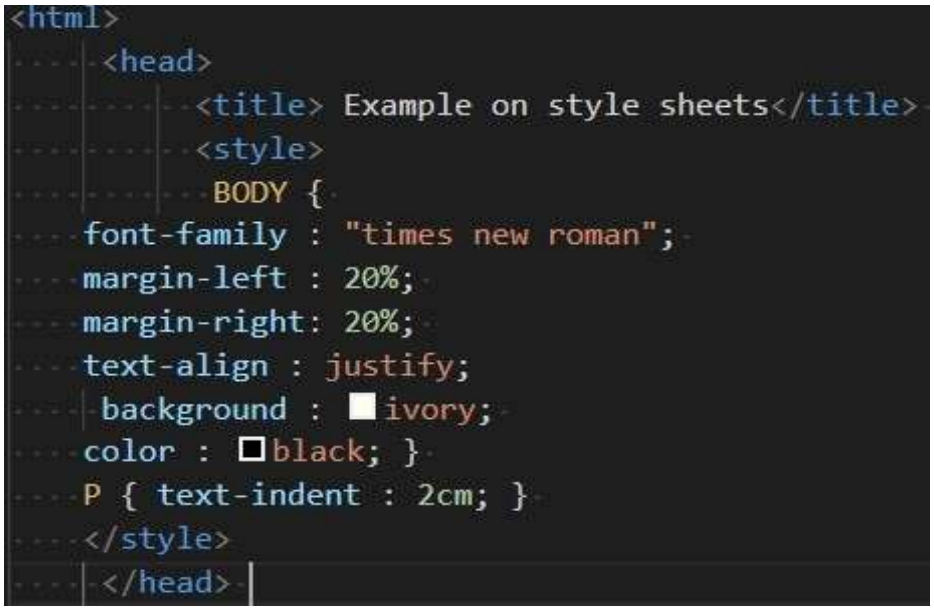
UNIT 1: HTML, CSS & Client Side Scripting

Cascading Style Sheets

CSS stands for Cascading Style Sheets and it is the language used to style the visual presentation of web pages. CSS is the language that tells web browsers how to render the different parts of a web page.

Every item or element on a web page is part of a document written in a mark-up language. In most cases, HTML is the mark-up language, but there are other languages in use such as XML. CSS is the language that defines the presentation of a web page. It is used to add colour, background images, and textures, and to arrange elements on the page. It is also used to enhance the usability of a website. The style sheet standard supported by modern browsers is called cascading style sheets, or CSS. CSS files contain a set of rules for the formatting of HTML documents.



```
<html>
    <head>
        <title> Example on style sheets</title>
        <style>
         BODY {
    font-family : "times new roman";
    margin-left : 20%;
    margin-right: 20%;
    text-align : justify;
     background : ▢ivory;
    color : ■black; }
    P { text-indent : 2cm; }
    </style>
    </head>
```

A style sheet is a collection of rules that describe the format of the HTML tags. In the above example there are six rules describing the format of the BODY tag, and one rule for the P tag.

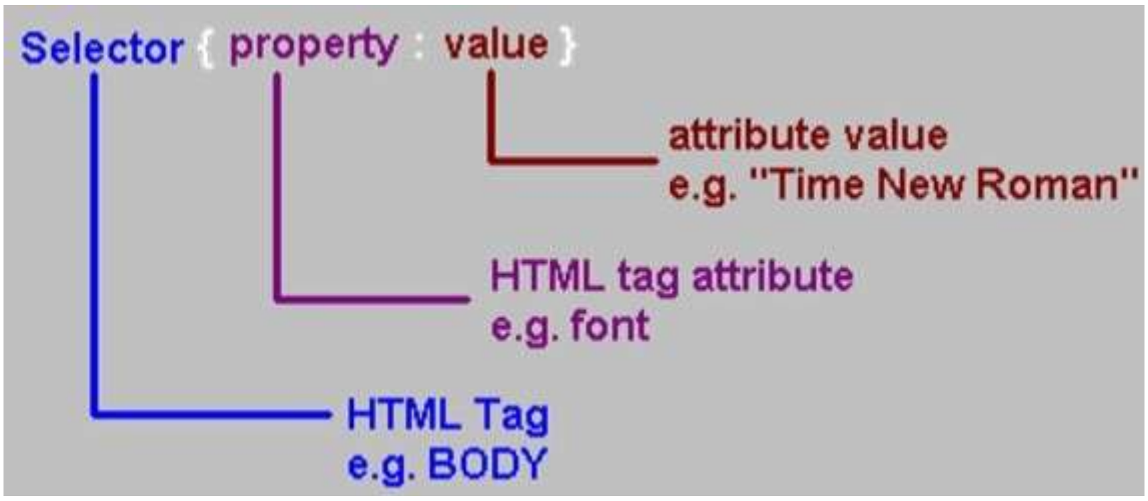
There are two ways to write style sheets:

the technically easier rule-based approach, and an approach that procedurally constructs a style sheet.

Below is a description of the rules used in the above example.

- Body - bgcolor set to "ivory", left and right margins indented relatively by 20%, font set to "Times New Roman" and text colour set to black.
- P to indent the first line by and absolute value of two centimetres.

There are three parts to a style sheet rule, shown in the figure below



The selector represents the HTML element that you want to style. For example:

h1 { color: blue }

This code tells the browser to render all occurrences of the HTML <h1> element in blue.

Grouping Selectors

You can apply a style to many selectors if you like. Just separate the selectors with a comma.

h1, h2, h3, h4, h5, h6 { color: blue }

Applying Multiple Properties

To apply more than one property separate each declaration with a semi-colon.

h1 { color:blue; font-family: arial, helvetica, "sans serif" }

Note:

CSS are a little different than their HTML counterparts. Instead of the <!-----> syntax, CSS ignores everything between /* and */ characters.

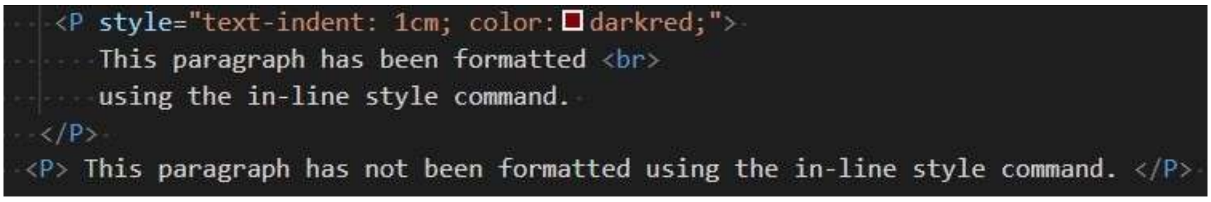
There are three ways to apply the above CSS rules to an HTML file:

1. Inline styles: add them in-line to the HTML file itself, attached directly to the relevant HTML tag.
2. Internal stylesheets : embed the rules into the HTML file
3. External stylesheets : link the CSS file to the HTML file

In-line Styles

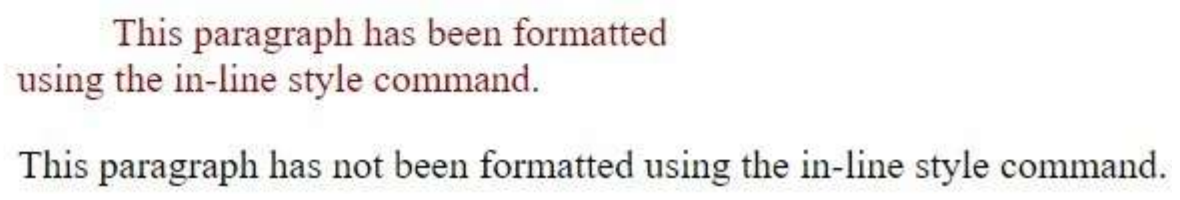
In-Line styles are added to individual tags and are usually avoided. Like the FONT tag they clog up HTML documents, making them larger and increasing their download times.

An example of an in-line style is given below:



```
    <P style="text-indent: 1cm; color:■darkred;">
        This paragraph has been formatted <br>
        using the in-line style command.
    </P>
  <P> This paragraph has not been formatted using the in-line style command. </P>
```

Result:

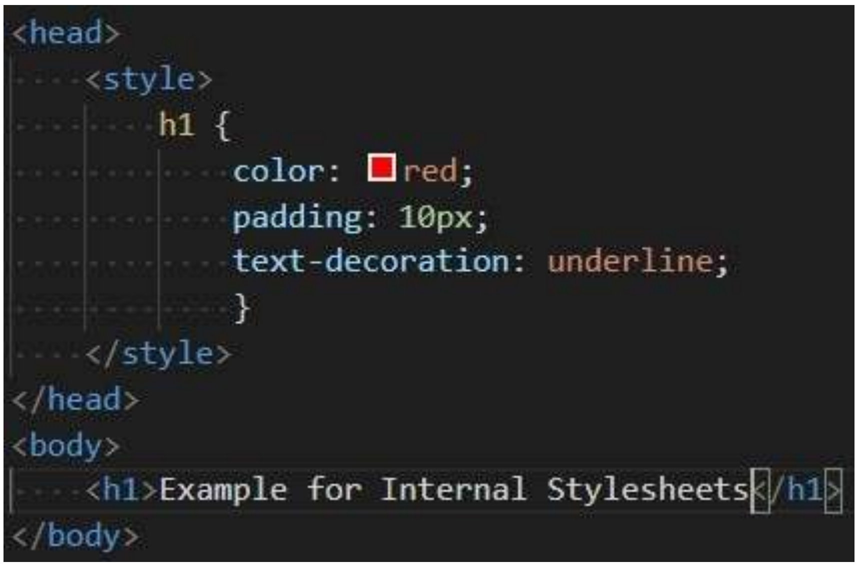


This paragraph has been formatted
using the in-line style command.

This paragraph has not been formatted using the in-line style command.

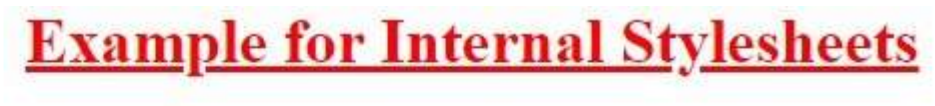
Internal Stylesheets

An internal stylesheet is a block of CSS added to an HTML document head element. The style element is used between the opening and closing head tags, and all CSS declarations are added between the style tags.



```
<head>
    <style>
        h1 {
            color: red;
            padding: 10px;
            text-decoration: underline;
            }
    </style>
</head>
<body>
    <h1>Example for Internal Stylesheets</h1>
</body>
```

Results:



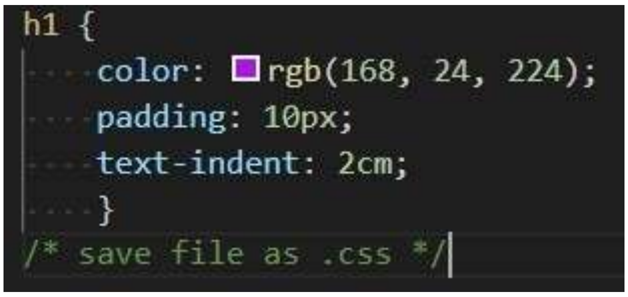
Example for Internal Stylesheets

External Stylesheets

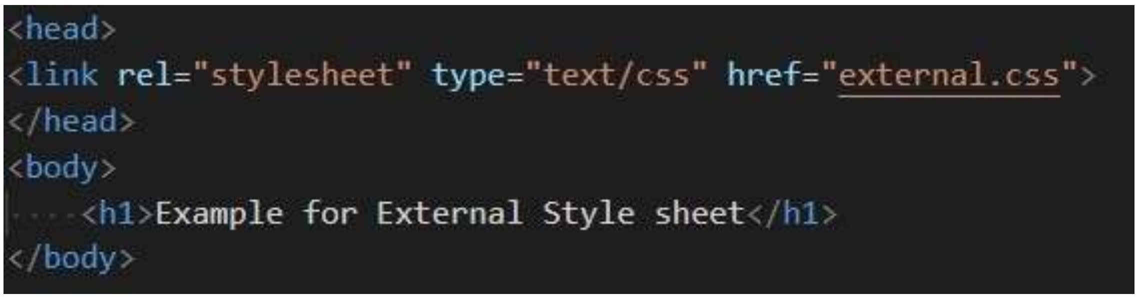
External stylesheets are documents containing nothing other than CSS statements. The rules defined in the document are linked to one or more HTML documents by using the link tag within the head element of the HTML document.

To use an external stylesheet, first create the CSS document and link it to an HTML document using the link element.

When this HTML document is loaded the link tag will cause the styles in the file styles.css to be loaded into the web page. As a result, all level 1 heading elements will appear with red text, underlined, and with 10 pixels of padding applied to every side.

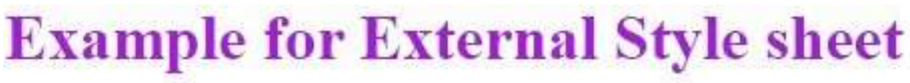


```
h1 {
    color: rgb(168, 24, 224);
    padding: 10px;
    text-indent: 2cm;
    }
/* save file as .css */
```



```
<head>
<link rel="stylesheet" type="text/css" href="external.css">
</head>
<body>
    <h1>Example for External Style sheet</h1>
</body>
```

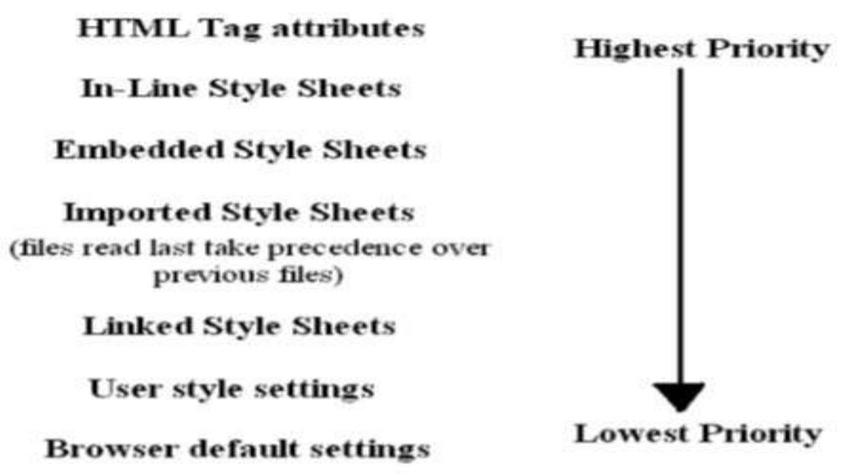
Results:



Example for External Style sheet

The cascading style sheet standard supplies very powerful tools to control Web page formatting. For instance, consider a university with many departments each with their own individual design criteria that is producing a website. It is possible to create a hierarchy of style sheets that allows each department's website to maintain formatting consistency with all the other university sites, while allowing each department to deviate from the format where needed.

The hierarchical (cascading) structure of style sheets can be used to do this. The figure below illustrates the style hierarchy design by W3C.



The Style Element

The <style> element is a "metadata” type element, which means its purpose is to provide setup for how the rest of the document will be displayed. This element is used to embed style declarations within our HTML document, rather than linking to an external dedicated stylesheet. It is essentially a form of inline styling.

In addition to the global HTML attributes, the style element accepts the following attributes -

Attribute	Description	Default
type	A valid MIME type that specifies the language for the style	“text/css”
media	A valid media query	“all”
title	Specifies a title for an alternate style element	None
scoped*	Allows us to place a **<style>** element within the body and limit the scope of the styles to the parent element	none/false

By default, most browsers apply a single line of styling to the <style> element -

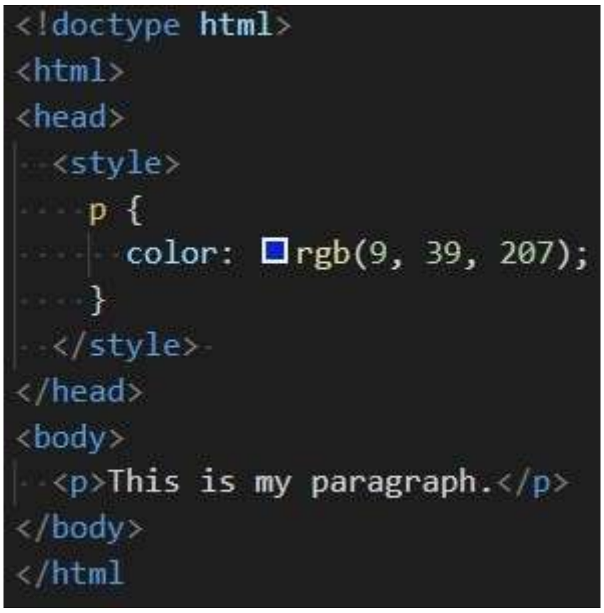
style {

display: none;

}

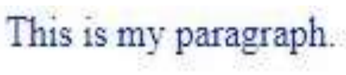
A simple stylesheet :

In the following example, apply a very simple stylesheet to a document:



```
<!doctype html>
<html>
<head>
  <style>
    p {
      color: ■rgb(9, 39, 207);
    }
  </style>
</head>
<body>
  <p>This is my paragraph.</p>
</body>
</html
```

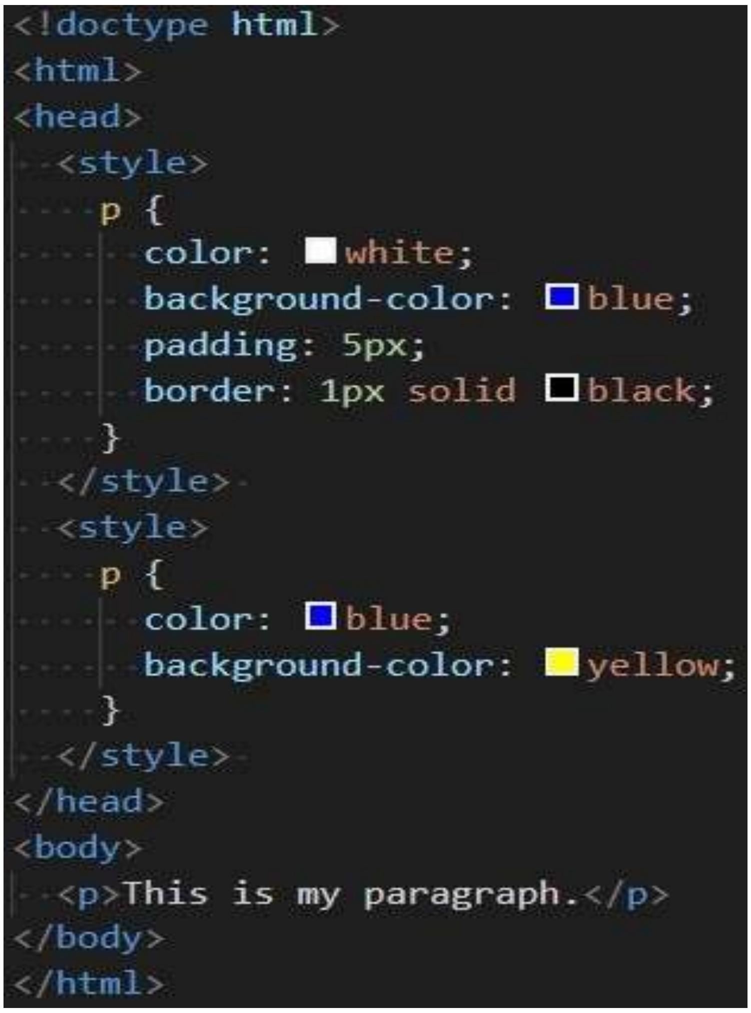
Result:



This is my paragraph.

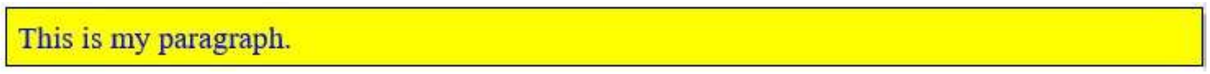
Multiple style elements

Two <style> elements have been included in the example let’s see how the conflicting declarations in the later <style> element override those in the earlier one, if they have equal specificity.



```
<!doctype html>
<html>
<head>
  <style>
    p {
      color: white;
      background-color: blue;
      padding: 5px;
      border: 1px solid black;
    }
  </style>
  <style>
    p {
      color: blue;
      background-color: yellow;
    }
  </style>
</head>
<body>
  <p>This is my paragraph.</p>
</body>
</html>
```

Results:



This is my paragraph.

MIME type

MIME stands for "Multipurpose Internet Mail Extensions. It's a way of identifying files on the Internet according to their nature and format. For example, using the "Content-type" header value defined in a HTTP response, the browser can open the file with the proper extension/plug-in.

A "Content-type" is simply a header defined in many protocols, such as HTTP, that makes use of MIME types to specify the nature of the file currently being handled.

Simplest MIME type consists of a type and a subtype; these are each strings which, when concatenated with a slash (/) between them, comprise a MIME type. No whitespace is allowed in a MIME type:

type/subtype

The type represents the general category into which the data type falls, such as video or text. The subtype identifies the exact kind of data of the specified type the MIME type represents. For example, for the MIME type text, the subtype might be plain (plain text), html (HTML source code), or calendar (for iCalendar/.ics) files.

Each type has its own set of possible subtypes, and a MIME type always has both a type and a subtype, never just one or the other.

There are two classes of type: **discrete** and **multipart**.

Discrete types are types which represent a single file or medium, such as a single text or music file, or a single video.

A *multipart* type is one which represents a document that's comprised of multiple component parts, each of which may have its own individual MIME type; or, a multipart type may encapsulate multiple files being sent together in one transaction. For example, multipart MIME types are used when attaching multiple files to an email.

text/plain

This is the default for textual files. Even if it really means "unknown textual file," browsers assume they can display it.

text/css

CSS files used to style a Web page must be sent with text/css. If a server doesn't recognize the .css suffix for CSS files, it may send them with text/plain or application/octet-stream MIME types.

text/html

All HTML content should be served with this type

image/jpeg

JPEG images

image/png

PNG images

Cascading in CSS

CSS: The class selector

The class selector is a way to select all of the elements with the specified class name, and apply styles to each of the matching elements. Declare a CSS class by using a dot (.) followed by the class name. You make up the class name yourself. After the class name you simply enter the properties/values that you want to assign to your class. The browser will look for all tags in the page that have a class attribute containing that class name.

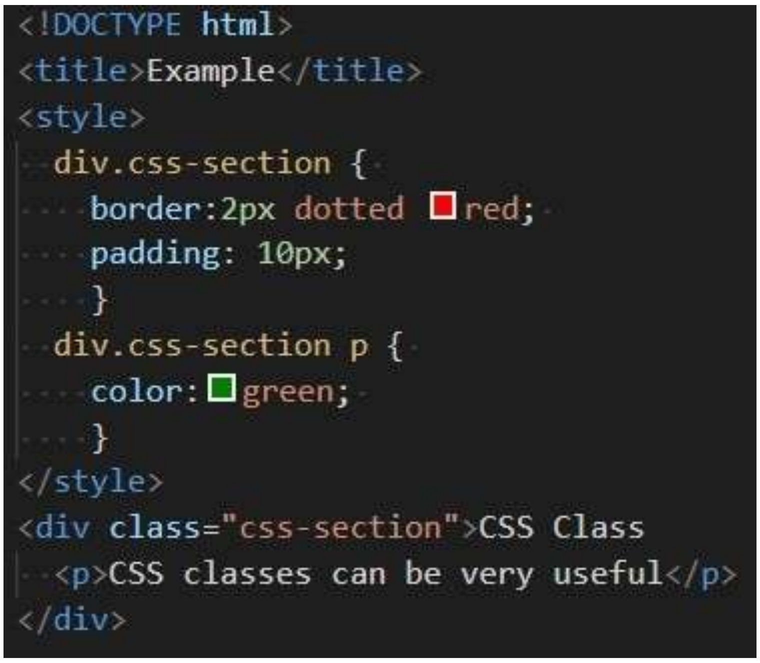
.class-name { property:value; }

If you want to use the same class name for multiple elements, but each with a different style, you can prefix the dot with the HTML element name.

For example:

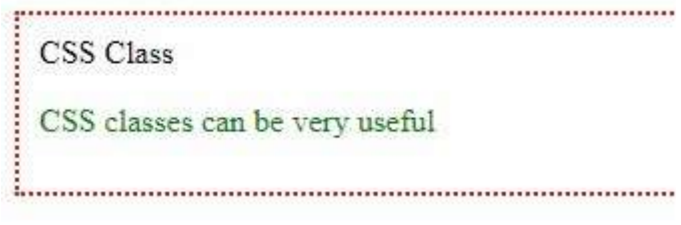
p.large { font-size: 2em; }

The class name can't contain a space, but it can contain hyphens or underscores. Any tag can have multiple space-separated class names.



```
<!DOCTYPE html>
<title>Example</title>
<style>
  div.css-section {
    border:2px dotted 🟥red;
    padding: 10px;
    }
  div.css-section p {
    color:🟩green;
    }
</style>
<div class="css-section">CSS Class
  <p>CSS classes can be very useful</p>
</div>
```

Result:



CSS Class

CSS classes can be very useful

CSS Class

CSS classes can be very useful

CSS ID Selectors

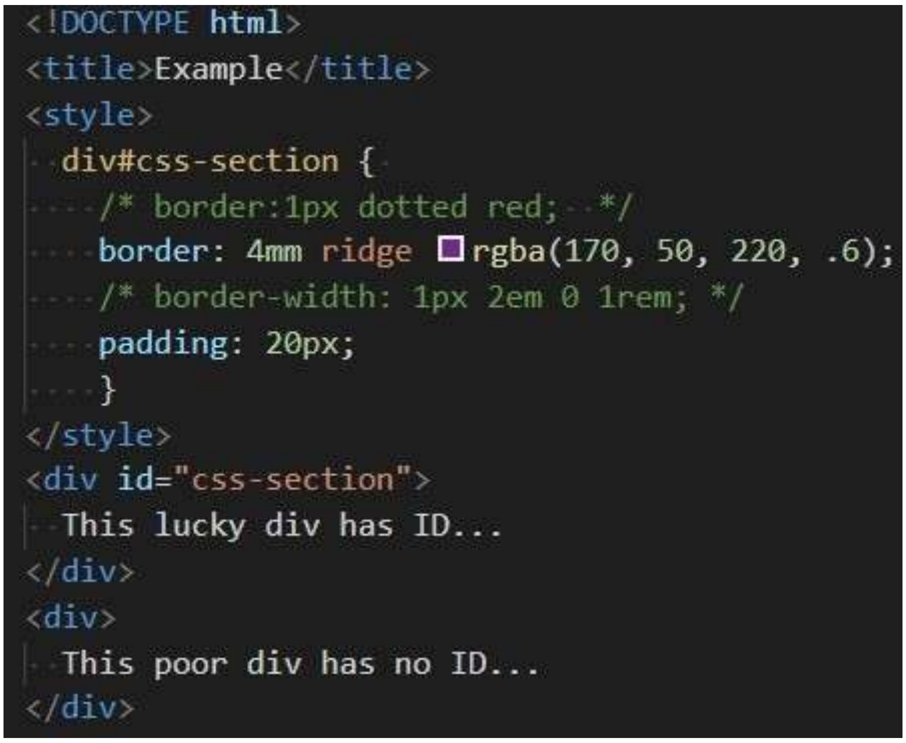
The id selector is a way to select only the element with the specified id, and apply styles to that element. The selector must start with a pound sign (#) and then the value of the id attribute. The browser will then look for a tag in the page that has an id attribute equal to that id.

The spelling and the casing must be exactly the same - #soon_gone is different from #Soon_Gone. The page should not have multiple tags with the same id- every id should be unique.

#id-name { property:value; }

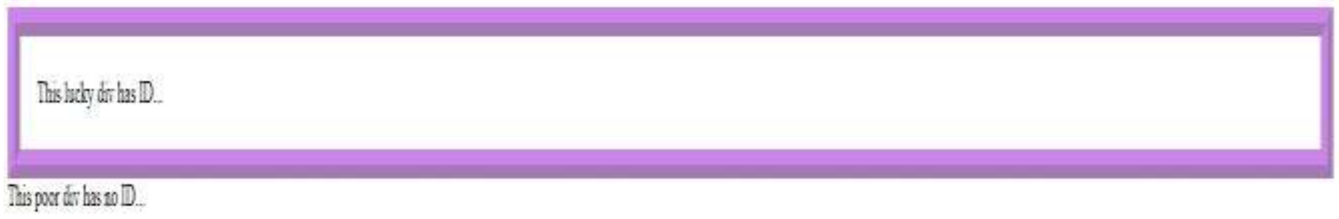
Again, similar to classes, if you want to use the same id name for multiple elements, but each with a different style, you can prefix the hash with the HTML element name.

p#intro { font-size: 2em; }



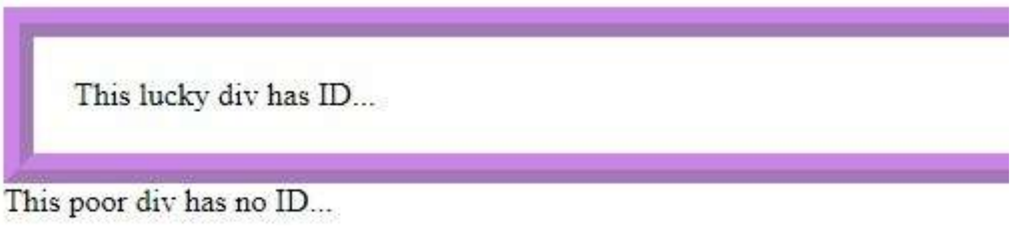
```
<!DOCTYPE html>
<title>Example</title>
<style>
  div#css-section {
    /* border:1px dotted red;  */
    border: 4mm ridge rgba(170, 50, 220, .6);
    /* border-width: 1px 2em 0 1rem; */
    padding: 20px;
    }
</style>
<div id="css-section">
  This lucky div has ID...
</div>
<div>
  This poor div has no ID...
</div>
```

Result:



This lucky div has ID...

This poor div has no ID...



This lucky div has ID...

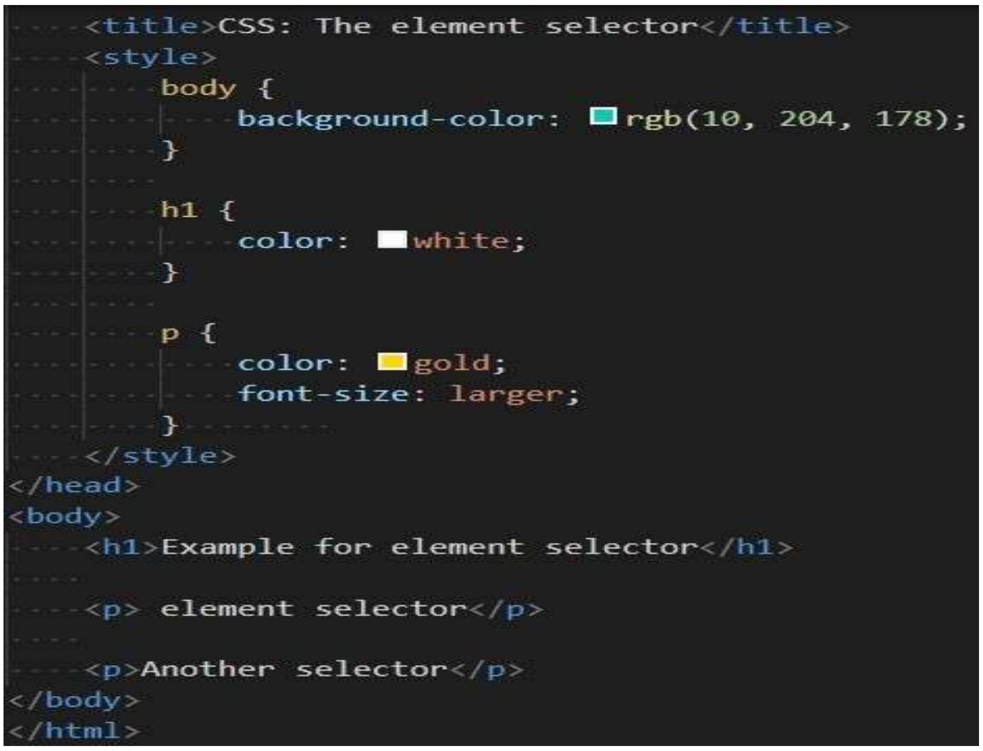
This poor div has no ID...

Note: ID applies to EVERY single thing with that specified element, while class applies only to the classes you gave to everything, so ID would make every h2 purple in ID for example, but class would only make the one h2 you gave the class to purple.

CSS: The element selector

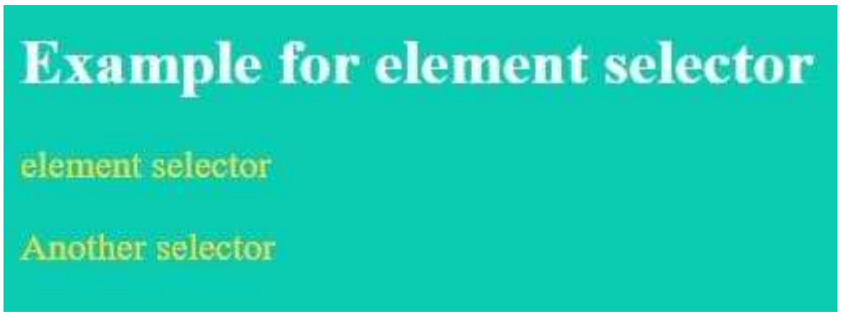
The element selector is a way to select all the elements with a given tag name in a document, and apply the same styles to each element with the tag name.

Note that you should only write the tag name, and not write brackets around the tag name — h1, not <h1>.



```
    <title>CSS: The element selector</title>
    <style>
        body {
            background-color: rgb(10, 204, 178);
        }

        h1 {
            color: white;
        }

        p {
            color: gold;
            font-size: larger;
        }
    </style>
</head>
<body>
    <h1>Example for element selector</h1>

    <p> element selector</p>

    <p>Another selector</p>
</body>
</html>
```

Results:



Example for element selector

element selector

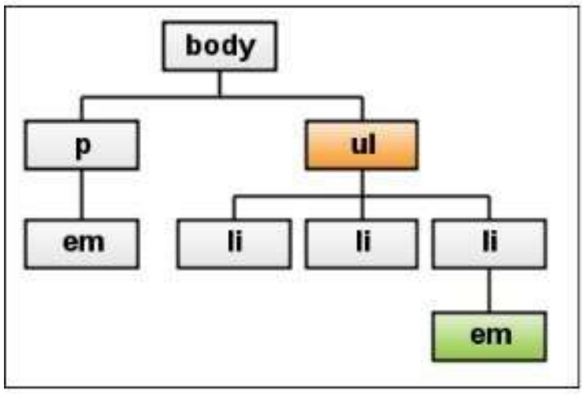
Another selector

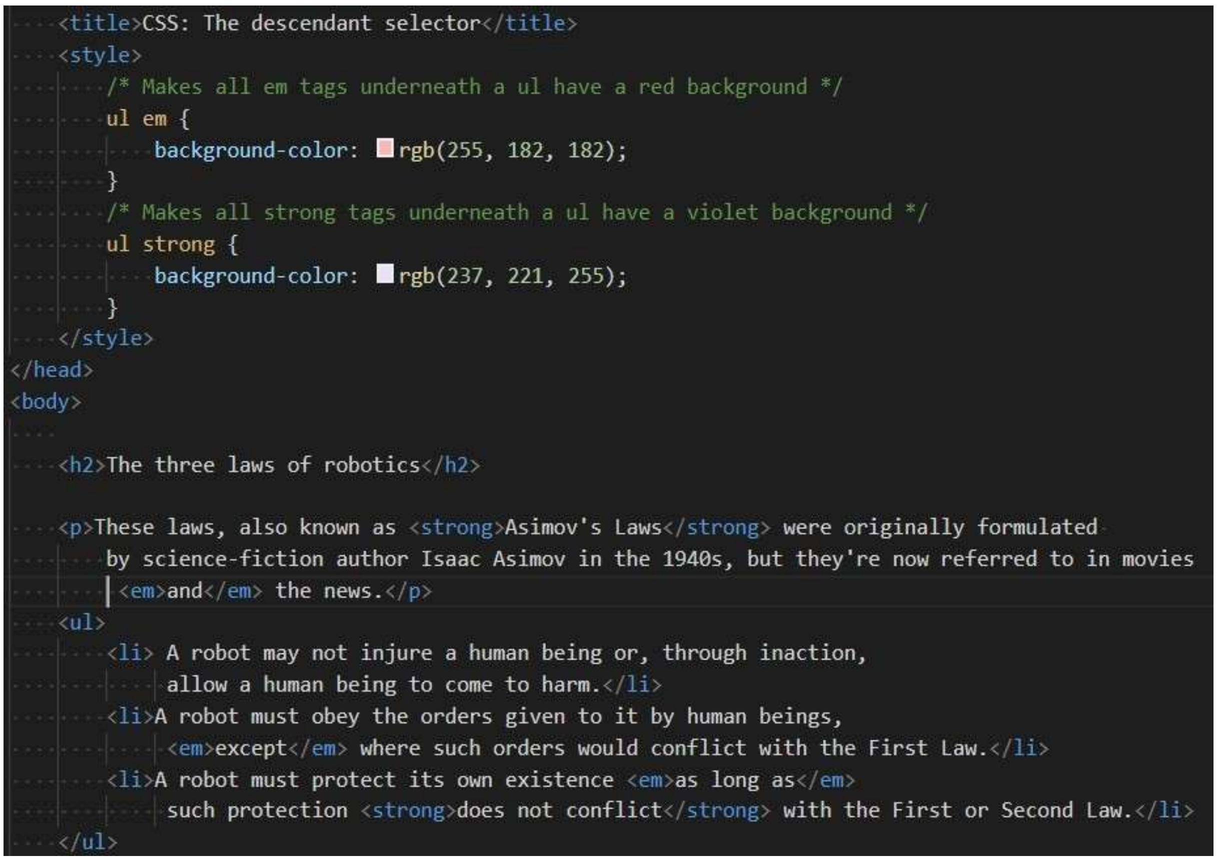
CSS: The descendant selector

The descendant selector is a way to select elements that are located somewhere underneath other elements, according to the tree structure of the webpage. This selector is actually multiple selectors combined together, but separated by a space. The first selector is the "ancestor", the element(s) that must be located higher in the tree, and the second selector is the "descendant", the element(s) that must be located somewhere underneath the ancestor in the tree.

To understand ancestor/descendant relationships, you need to think about the webpage is a tree.

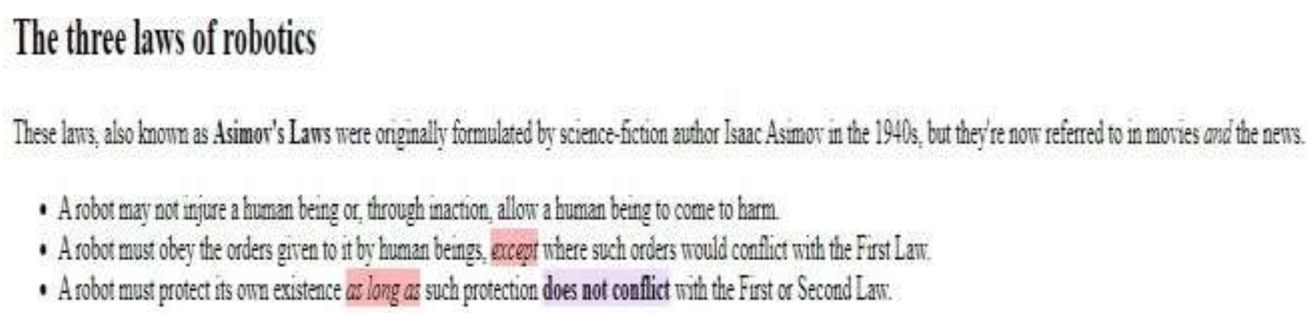
An example tree structure is shown in the image to the right. The ul is an ancestor of the em below it, but it is not an ancestor of the other em. That means that a ul em selector will only select a single em, not both of them.





```
    <title>CSS: The descendant selector</title>
    <style>
        /* Makes all em tags underneath a ul have a red background */
        ul em {
            background-color: rgb(255, 182, 182);
        }
        /* Makes all strong tags underneath a ul have a violet background */
        ul strong {
            background-color: rgb(237, 221, 255);
        }
    </style>
</head>
<body>
    
    <h2>The three laws of robotics</h2>

    <p>These laws, also known as <strong>Asimov's Laws</strong> were originally formulated
        by science-fiction author Isaac Asimov in the 1940s, but they're now referred to in movies
        <em>and</em> the news.</p>
    <ul>
        <li> A robot may not injure a human being or, through inaction,
             allow a human being to come to harm.</li>
        <li>A robot must obey the orders given to it by human beings,
             <em>except</em> where such orders would conflict with the First Law.</li>
        <li>A robot must protect its own existence <em>as long as</em>
             such protection <strong>does not conflict</strong> with the First or Second Law.</li>
    </ul>
```

Result:



The three laws of robotics

These laws, also known as **Asimov's Laws** were originally formulated by science-fiction author Isaac Asimov in the 1940s, but they're now referred to in movies *and* the news.

- A robot may not injure a human being or, through inaction, allow a human being to come to harm.
- A robot must obey the orders given to it by human beings, *except* where such orders would conflict with the First Law.
- A robot must protect its own existence *as long as* such protection **does not conflict** with the First or Second Law.

Pseudo-classes

A CSS pseudo-class is a keyword added to a selector that specifies a special state of the selected element(s). For example, ***:hover*** can be used to change a button's color when the user's pointer hovers over it.

/* Any button over which the user's pointer is hovering */

```
button:hover {
color: blue;
}
```

Pseudo-classes let you apply a style to an element not only in relation to the content of the document tree, but also in relation to external factors like the history of the navigator (:visited, for example), the status of its content (like :checked on certain form elements), or the position of the mouse (like :hover, which lets you know if the mouse is over an element or not). A few common pseudo-classes are :link, :visited, :hover, :active, :first-child and :nth-child.

CSS pseudo-classes and pseudo-elements can certainly be a handful.

Syntax:

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

Single Or Double Colon For Pseudo-Elements?

The double colon (::) was introduced in CSS3 to differentiate pseudo-elements such as ::before and ::after from pseudo-classes such as :hover and :active. All browsers support double colons for pseudo-elements except Internet Explorer (IE) 8 and below.

Some pseudo-elements, such as ::backdrop, accept only a double colon, though.

Dynamic pseudo-classes

These are the link-related pseudo-class states which were included in CSS1. Each of these states can be applied to an element, usually <a>. They include;

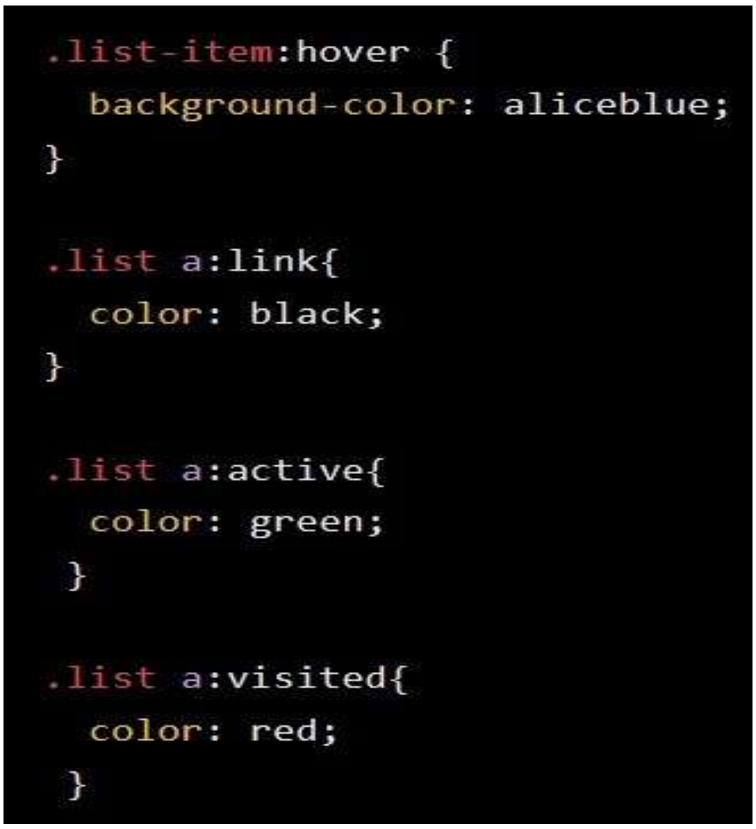
:link - This only selects <a> tags with href attributes. It will not work if it is missing.

:active - Selects the link while it is being activated (being clicked on or otherwise activated). For example, for the “pressed” state of a button-style link.

:visited - Selects links that have already been visited by the current browser.

:hover - This is the most commonly used state. When the mouse cursor rolls over a link, that link is in its hover state and this will select it.

Referring to index.html, it can like to change the background of <li> when hovered, give specific colours to all links, active and visited links.



```
.list-item:hover {
  background-color: aliceblue;
}

.list a:link{
  color: black;
}

.list a:active{
  color: green;
 }

.list a:visited{
  color: red;
 }
```

Structural pseudo-classes

These exciting positioning states/selectors were introduced in CSS2. They target elements according to their position in the document tree and relation to other elements. They include

root - This selects the element that is at the root of the document specifically the <html> element unless you are specifically working in some other environment that somehow also allows CSS.

:first-child - Selects the first element within a parent.

:last-child - Selects the last element within a parent.

:nth-child() - Selects elements based on a provided algebraic expression (e.g. “2n” or “4n-1”). For example, you could use ‘2n’ for selecting even positions and ‘2n-1’ for odd positions. Has the ability to do other things like select “every fourth element”, “the first six elements”, and things like that.

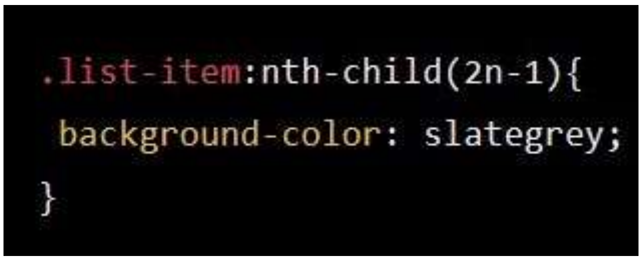
:first-of-type - Selects the first element of this type within any parent. If for example, you have two divs, each with a paragraph, link, paragraph, link. Then div a:first-of-type would select the first link inside the first div and the first link inside the second div.

:*last-of-type* - This works the same as above but it then selects the last element instead of the first element.

:nth-of-type() - Works like :nth-child, but it is used in places where the elements at the same level are of different types. For example, if inside a div you had a number of paragraphs and links. You wanted to select all the odd paragraphs. :nth-child wouldn’t work in this scenario, therefore, you use div p:nth-of-type(odd).

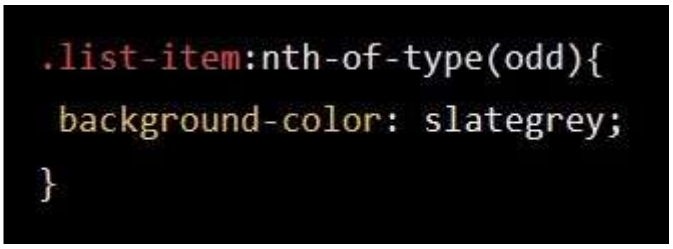
:only-of-type - Selects the element if and only if it is one of its kind within the current parent.

Method one: Using the :nth-child()



```
.list-item:nth-child(2n-1){
 background-color: slategrey;
}
```

Method two: Using the :nth-of-type()



```
.list-item:nth-of-type(odd){
 background-color: slategrey;
}
```

Pseudo-Elements

Content-related pseudo-elements effectively create new elements that are not specified in the mark-up of the document and can be manipulated much like a regular element. This introduces huge benefits for creating cool effects with minimal mark-up, also aiding significantly in keeping the presentation of the document out of the HTML and in CSS where it belongs.

Difference between Pseudo-classes and Pseudo-elements

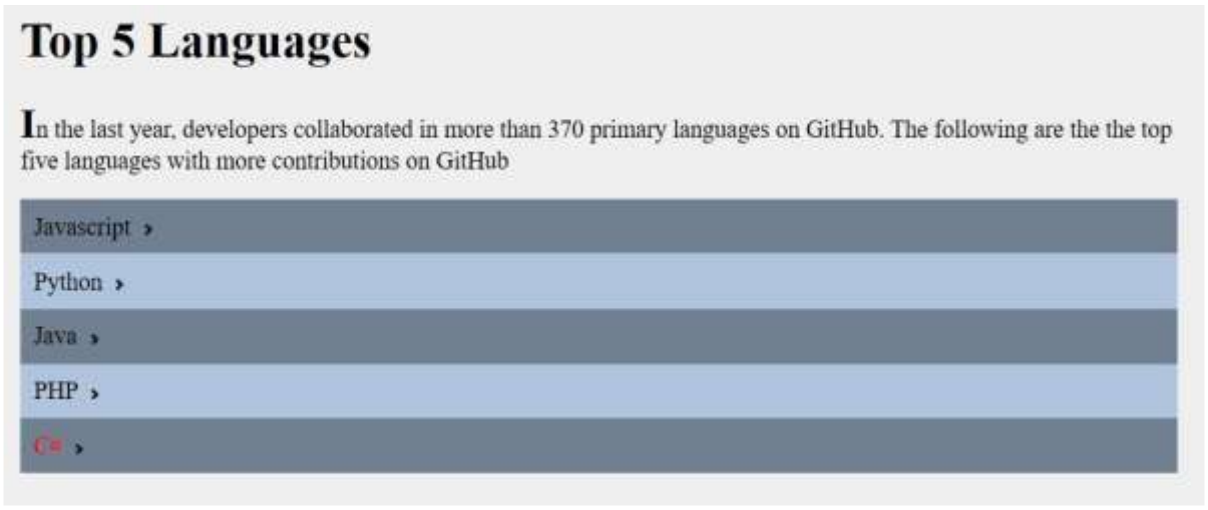
A pseudo-class is a selector that assists in the selection of something that cannot be expressed by a simple selector, for example: hover. A pseudo-element, however, allows us to create items that do not normally exist in the document tree, for example ::after. So you could simply identify a pseudo-class by a single colon (:) and a pseudo-element by two colons (::).

Pseudo-elements include

::before - This enables us to add content before a certain element. For example, adding an opening quote before a blockquote.

::after - This enables us to add content after a certain element. For example, a closing quotes to a blockquote. Also used commonly for the clearfix, where an empty space is added after the element which clears the float without any need for extra HTML mark-up.

::first-letter - This is used to add a style to the first letter of the specified selector. For example, to create a drop cap.



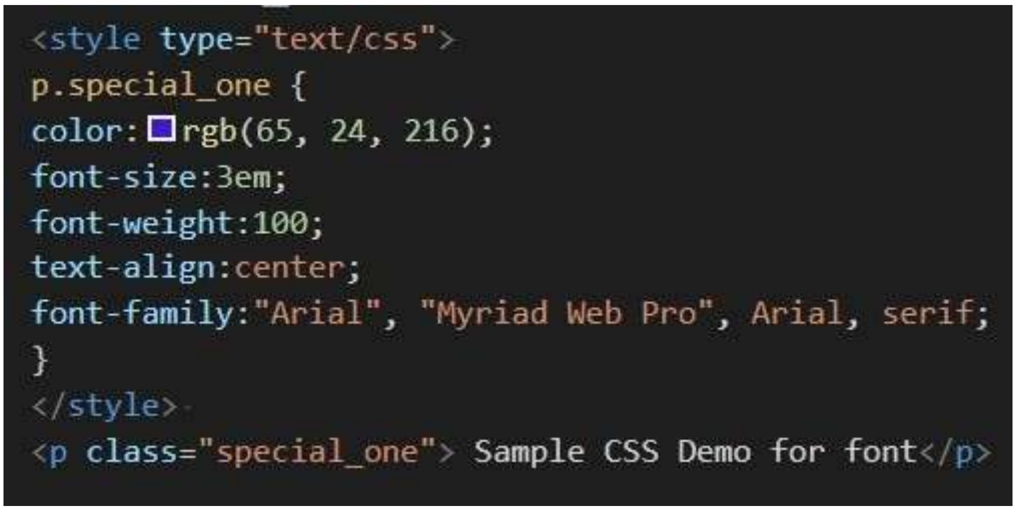
Top 5 Languages

In the last year, developers collaborated in more than 370 primary languages on GitHub. The following are the the top five languages with more contributions on GitHub

- Javascript ›
- Python ›
- Java ›
- PHP ›
- C# ›

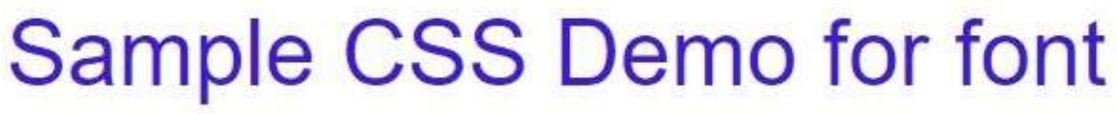
Font and Spacing
CSS has a lot of properties. CSS can do so many things, but often the font and spacing are overlooked. Common font related properties:

- color – sets the font color
- font-size – sets the font size
- font-weight – sets the font weight (skinny to very bold)
- text-align – sets the font position inside element (left, centre, right, justify)
- font-family – sets the actual font type



```
<style type="text/css">
p.special_one {
color:■rgb(65, 24, 216);
font-size:3em;
font-weight:100;
text-align:center;
font-family:"Arial", "Myriad Web Pro", Arial, serif;
}
</style>
<p class="special_one"> Sample CSS Demo for font</p>
```

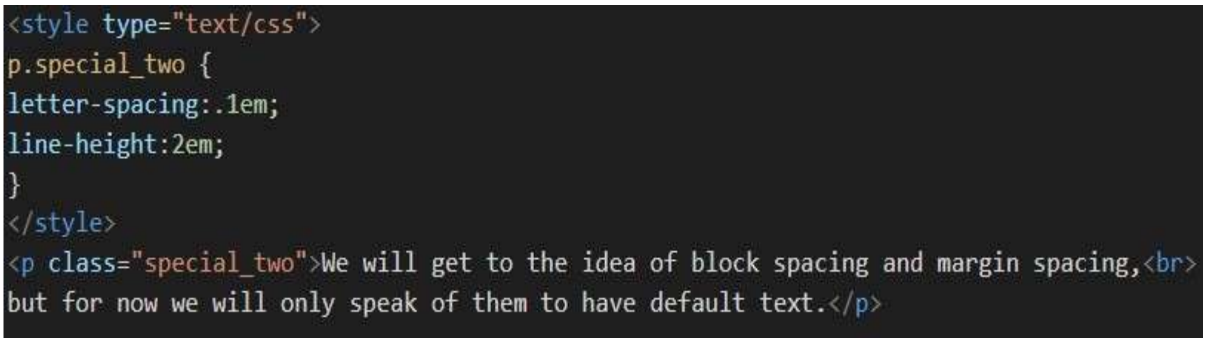
Results:



```
Sample CSS Demo for font
```

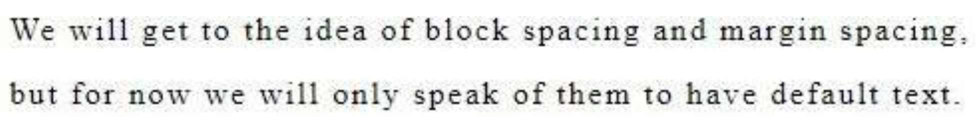
Text Spacing

Text spacing, refers to characters in relation to other characters. The first important property is letter-spacing, which is the horizontal spacing between characters. The second and final is the line-height that is the vertical spacing between text lines.



```
<style type="text/css">
p.special_two {
letter-spacing:.1em;
line-height:2em;
}
</style>
<p class="special_two">We will get to the idea of block spacing and margin spacing,<br>
but for now we will only speak of them to have default text.</p>
```

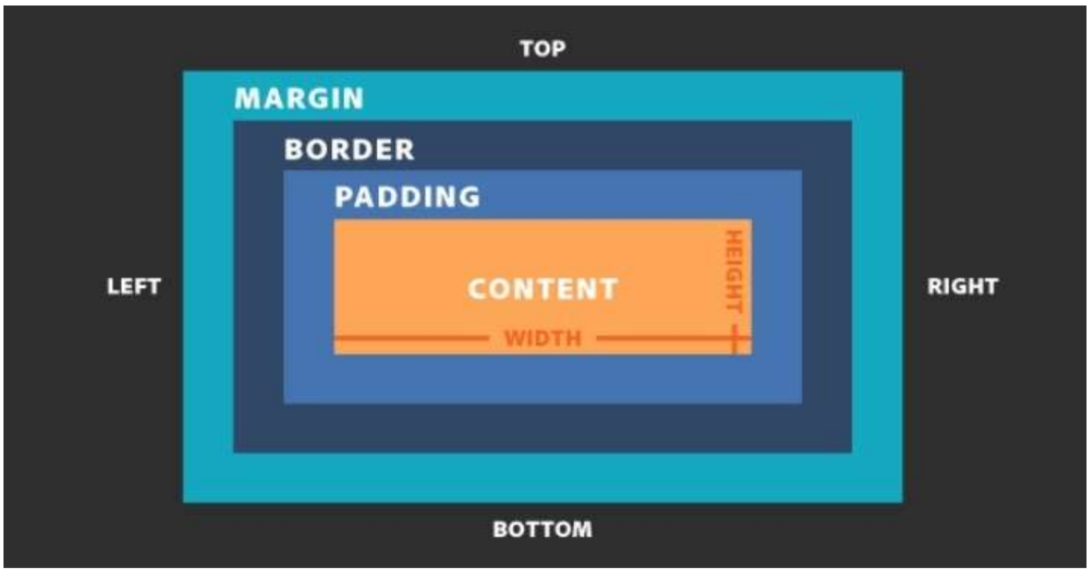
Results:



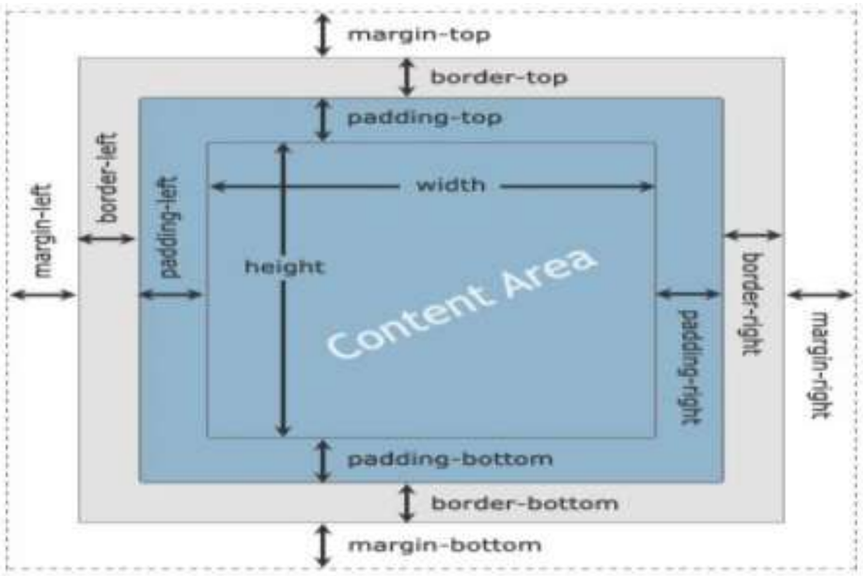
```
We will get to the idea of block spacing and margin spacing,

but for now we will only speak of them to have default text.
```

CSS box model

When laying out a document, the browser's rendering engine represents each element as a rectangular box according to the standard CSS basic box model. CSS determines the size, position, and properties (color, background, border size, etc.) of these boxes.

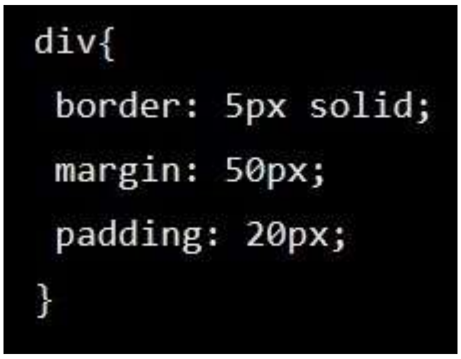


Every box is composed of four parts (or areas), defined by their respective edges: the content edge, padding edge, border edge, and margin edge. Every box has a content area and an optional surrounding margin, padding, and border.



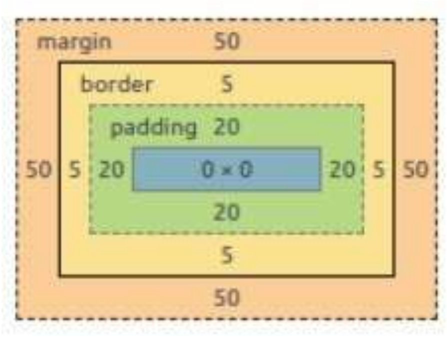
1. The innermost rectangle is the content box. The width and height of this depend on the element’s content (text, images, videos, any child elements).
2. Then have the padding box (defined by the padding property). If there is no padding width defined, the padding edge is equal to the content edge.
3. Next, the border box (defined by the border property). If there is no border width defined, the border edge is equal to the padding edge.
4. The outermost rectangle is the margin box. If there is no margin width defined, the margin edge is equal to the border edge.

Consider the following example:



```
div{
 border: 5px solid;
 margin: 50px;
 padding: 20px;
}
```

This CSS styles all div elements to have a top, right, bottom and left border of 5px in width and a top, right, bottom and left margin of 50px; and a top, right, bottom, and left padding of 20px. Ignoring content, our generated box will look like this:



CSS Position

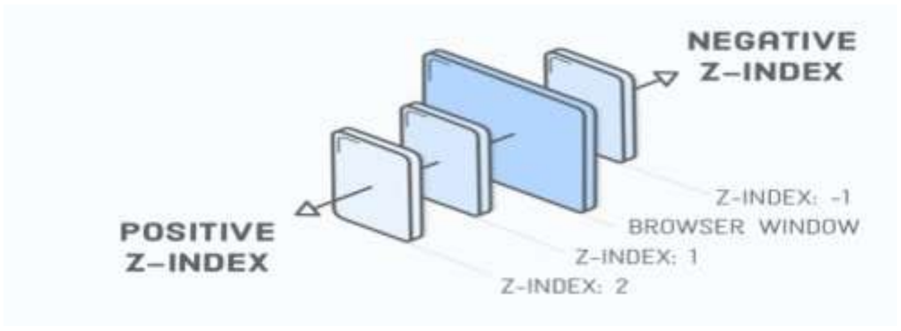
The position CSS property sets how an element is positioned in a document. The top, right, bottom, and left properties determine the final location of positioned elements. There are 5 main values of the Position Property:

position: static , relative , absolute , fixed , sticky

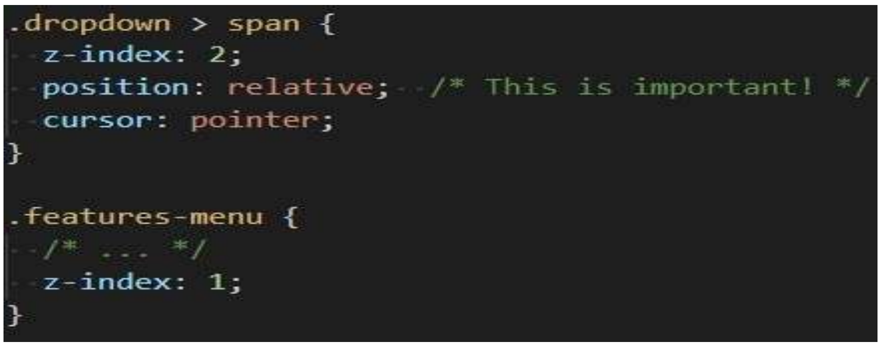
and additional properties for setting the coordinates of an element : **top , right , bottom , left AND the z-index.**

What is this z-index?

Consider height and width (x, y) as 2 dimensions. Z is the 3rd dimension. An element in the webpage comes in front of other elements as its z-index value increases. Z-index doesn't work with position: static or without a declared position.

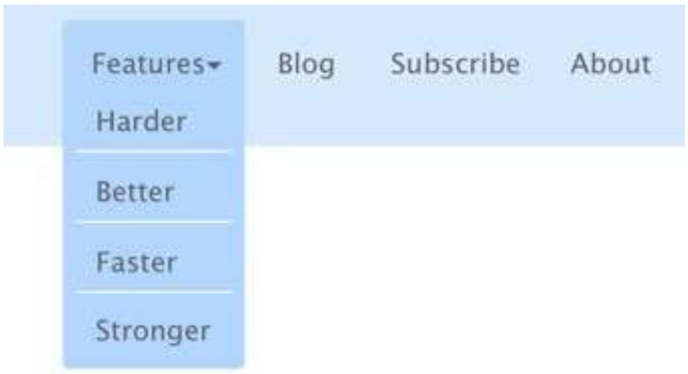


The .features-menu element needs to have a lower z-index than the Features label. The default z-index value is 0, so make both of them higher than that. It can conveniently wrapped the Features label in a <span>, allowing to style it via a child selector, like this



```
.dropdown > span {
  z-index: 2;
  position: relative;  /* This is important! */
  cursor: pointer;
}

.features-menu {
  /* ... */
  z-index: 1;
}
```

The Features label appear on top of the submenu. The position: relative; line. It’s required because only positioned elements pay attention to their z-index property.

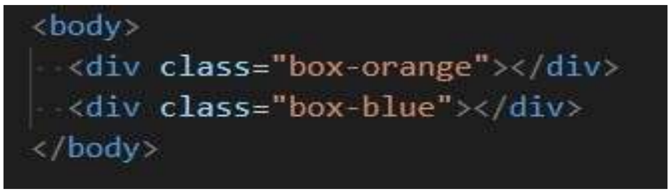


Now, lets see the Position Property:

1. Static

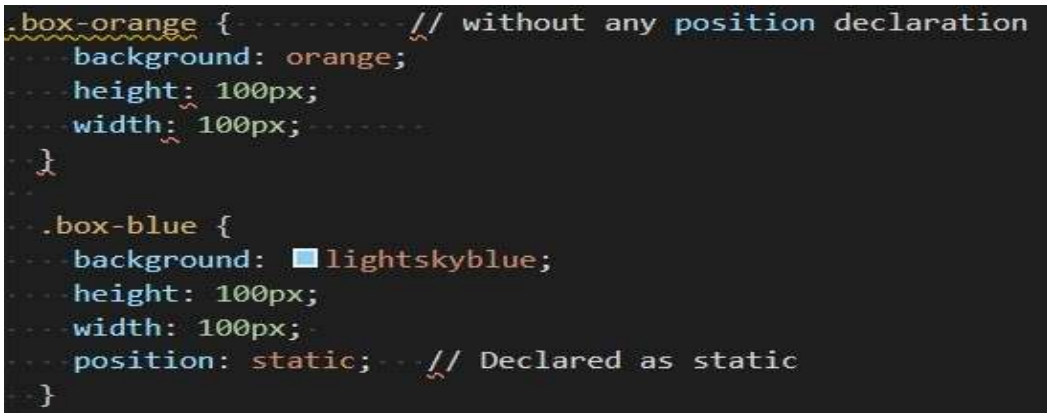
position: static is the default value. Whether declare it or not, elements are positioned in a normal order on the webpage. Let’s give an example:

First, define our HTML structure:



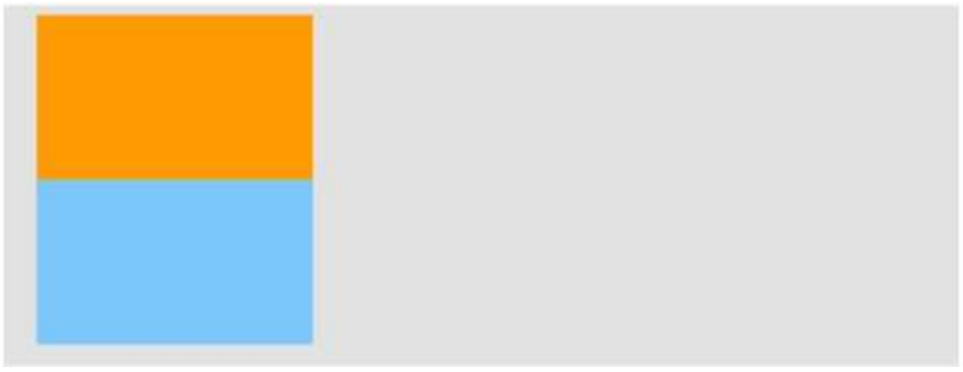
```
<body>
  <div class="box-orange"></div>
  <div class="box-blue"></div>
</body>
```

Then, create 2 boxes and define their widths, heights & positions:



```
.box-orange {          // without any position declaration
    background: orange;
    height: 100px;
    width: 100px;
  }

  .box-blue {
    background: lightskyblue;
    height: 100px;
    width: 100px;
    position: static;   // Declared as static
  }
```

Results:

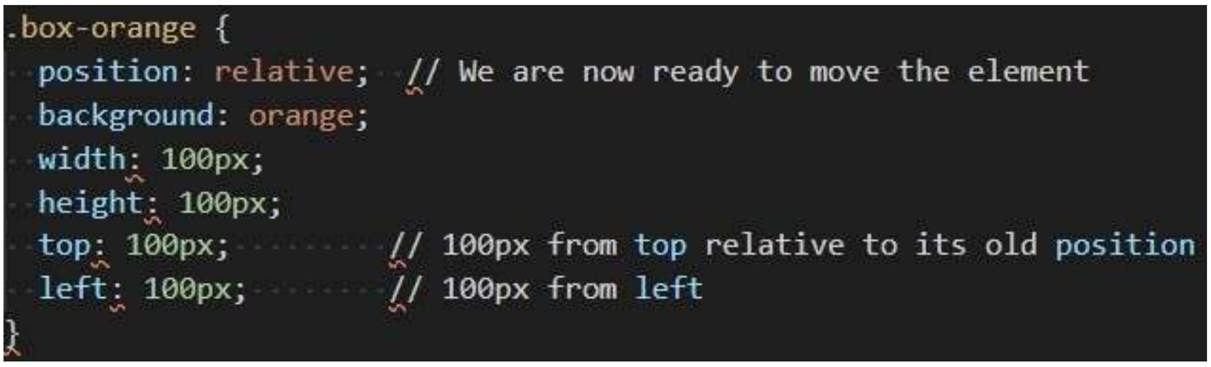


Hence from the result it can observe defining position: static or not doesn't make any difference. The boxes are positioned according to the normal document flow.

2. Relative

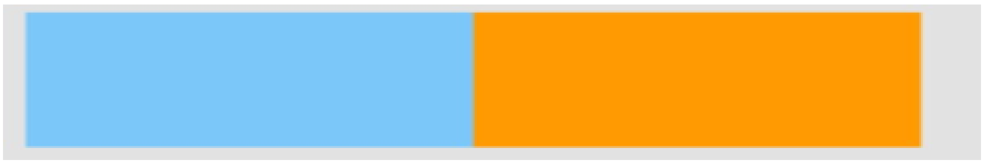
position: relative: An element’s new position relative to its normal position.

Starting with position: relative and for all non-static position values, able to change an element’s default position by using the helper properties. Move the orange box next to the blue one. Orange box is moved 100px to bottom & right, relative to its normal position.



```
.box-orange {
  position: relative;  // We are now ready to move the element
  background: orange;
  width: 100px;
  height: 100px;
  top: 100px;          // 100px from top relative to its old position
  left: 100px;         // 100px from left
}
```

Results:

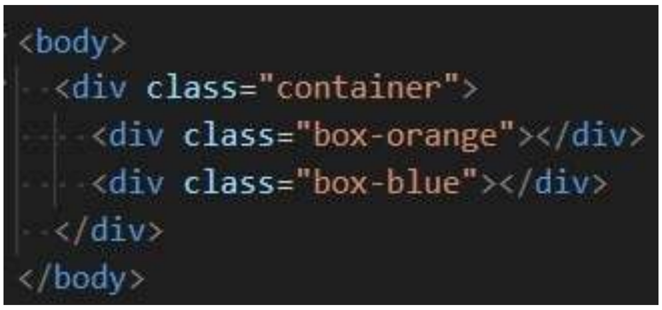


3.Absolute

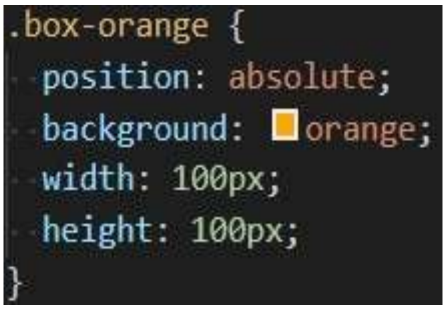
In position: relative, the element is positioned relative to itself. However, an absolutely positioned element is relative to its parent.

An element with position: absolute is removed from the normal document flow. It is positioned automatically to the starting point (top-left corner) of its parent element. If it doesn't have any parent elements, then the initial document <html> will be its parent.

Since position: absolute removes the element from the document flow, other elements are affected and behave as the element is removed completely from the webpage. Add a container as parent element.

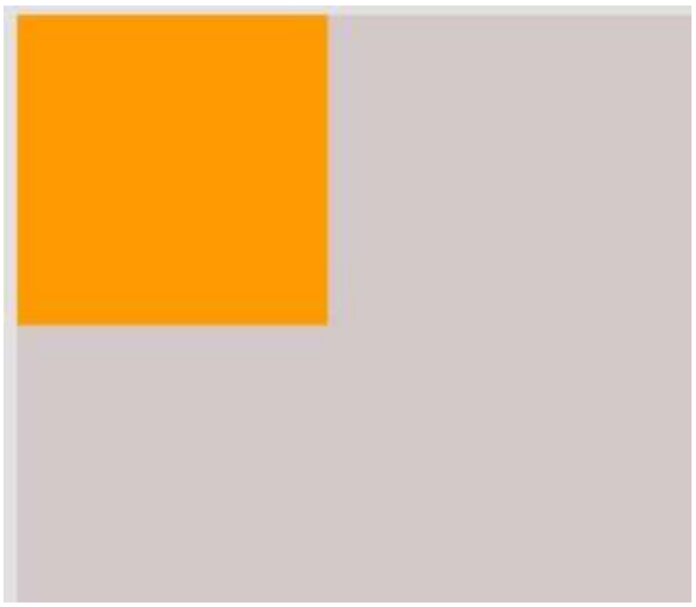


```html
<body>
    <div class="container">
        <div class="box-orange"></div>
        <div class="box-blue"></div>
    </div>
</body>
```

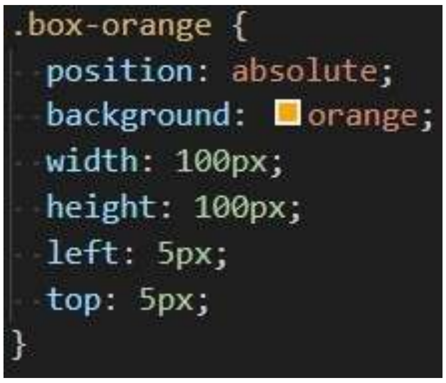


```css
.box-orange {
    position: absolute;
    background: ■orange;
    width: 100px;
    height: 100px;
}
```

Results:

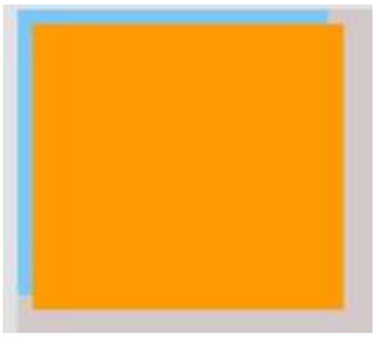


It looks like the blue box has disappeared, but it hasn't. The blue box behaves like the orange box is removed, so it shifts up to the orange box's place. So Let's move the orange box 5 pixels:



```
.box-orange {
  position: absolute;
  background: ■orange;
  width: 100px;
  height: 100px;
  left: 5px;
  top: 5px;
}
```

Results:

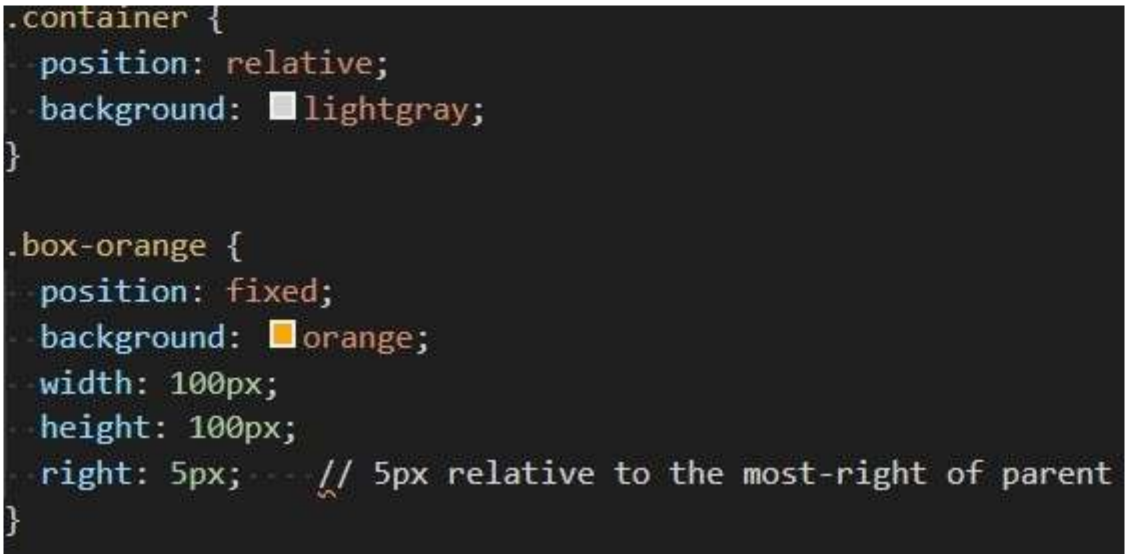


The coordinates of an absolute positioned element are relative to its parent if the parent also has a non-static position.

4.Fixed

Like position: absolute, fixed positioned elements are also removed from the normal document flow. The differences are:

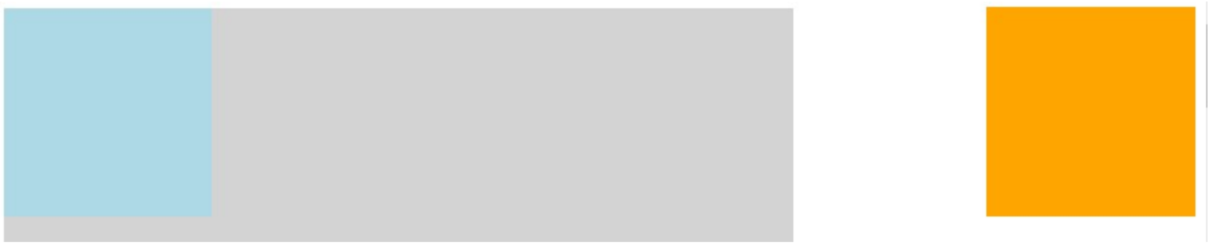
- They are only relative to the <html> document, not any other parents.
- They are not affected by scrolling.



```
.container {
  position: relative;
  background: lightgray;
}

.box-orange {
  position: fixed;
  background: orange;
  width: 100px;
  height: 100px;
  right: 5px;    // 5px relative to the most-right of parent
}
```

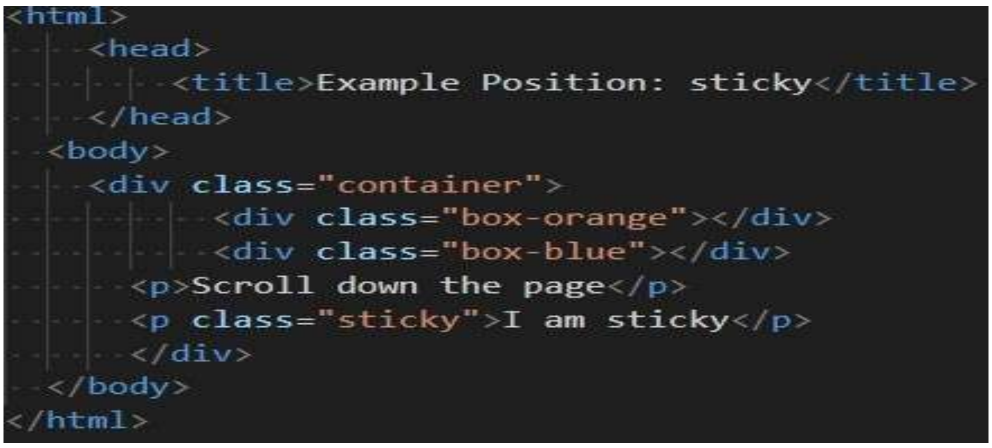
Here in the example, the orange box's position is changed to fixed, and this time it is relative 5px to the right of the <html>, not its parent (container):

Result :

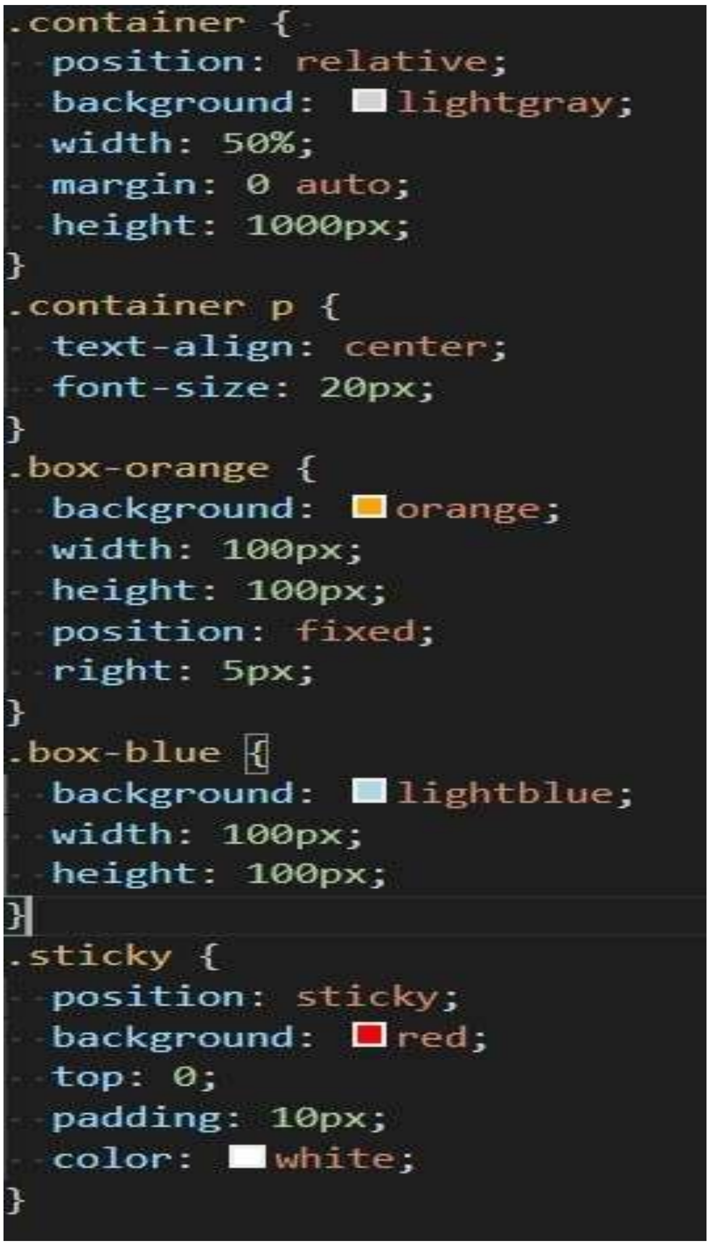


5.position: sticky

It can be explained as a mix of position: relative and position: fixed.It behaves until a declared point like position: relative, after that it changes its behaviour to position: fixed .

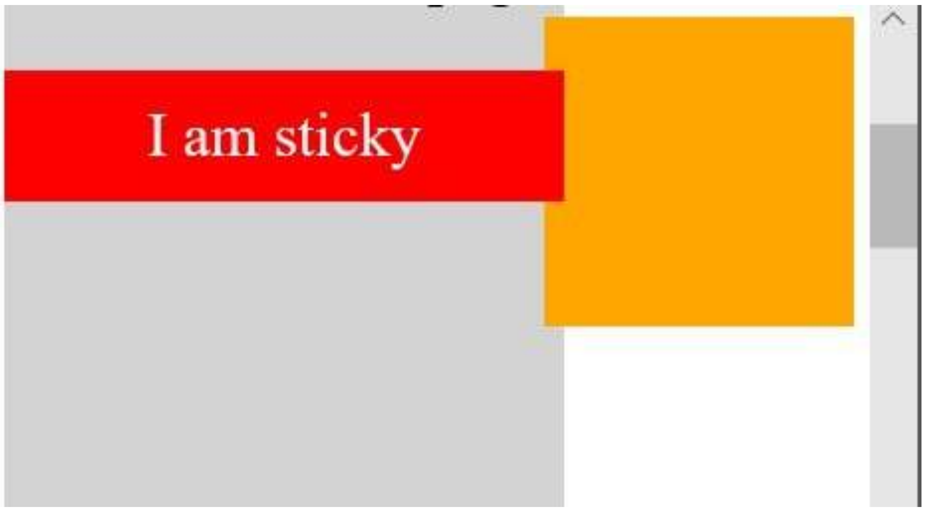


```
<html>
    <head>
        <title>Example Position: sticky</title>
    </head>
  <body>
    <div class="container">
          <div class="box-orange"></div>
          <div class="box-blue"></div>
      <p>Scroll down the page</p>
      <p class="sticky">I am sticky</p>
      </div>
  </body>
</html>
```



```
.container {
  position: relative;
  background: lightgray;
  width: 50%;
  margin: 0 auto;
  height: 1000px;
}
.container p {
  text-align: center;
  font-size: 20px;
}
.box-orange {
  background: orange;
  width: 100px;
  height: 100px;
  position: fixed;
  right: 5px;
}
.box-blue {
  background: lightblue;
  width: 100px;
  height: 100px;
}
.sticky {
  position: sticky;
  background: red;
  top: 0;
  padding: 10px;
  color: white;
}
```

Results:

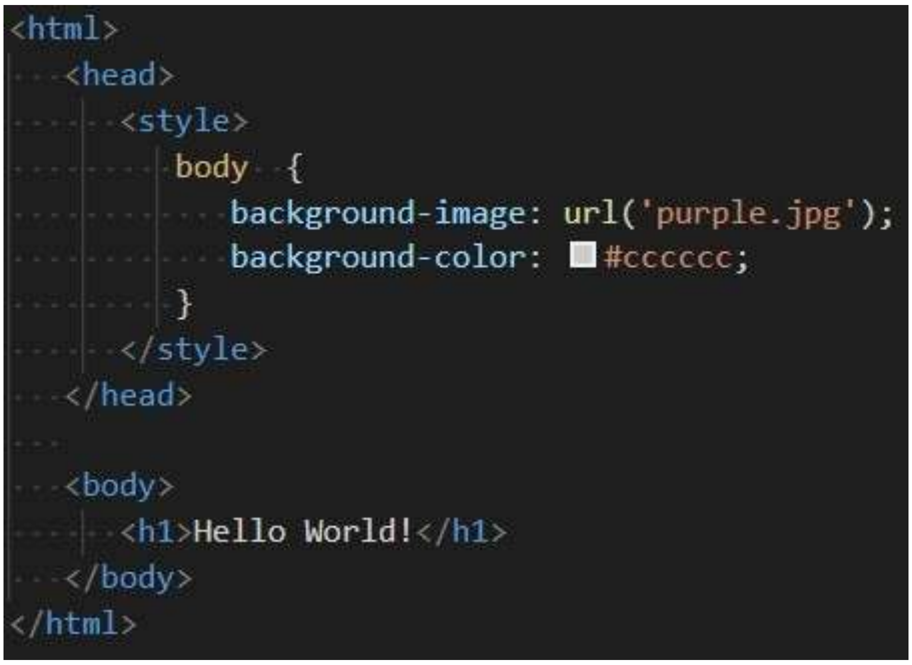


Background-image:

Background-image defines a pointer to an image resource which is to be placed in the background of an element.

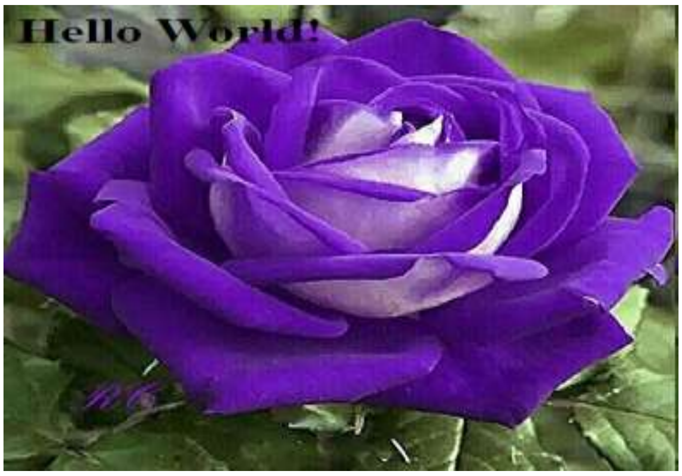
Syntax

object.style.backgroundImage = "Any value as defined above";



```
<html>
   <head>
      <style>
         body  {
            background-image: url('purple.jpg');
            background-color: ■ #cccccc;
         }
      </style>
   </head>
   
   <body>
      <h1>Hello World!</h1>
   </body>
</html>
```

Relts:


Hello World!