**Inline CSS**

* **Inline CSS** requires the style attribute placed **inside an HTML element**.
* To add inline CSS, you use a style attribute and place it inside the opening tag of an HTML element. Here's the syntax:

<element style="CSS property: value">

* **It will override any other CSS targeting the same elements**. For this reason, inline CSS is effective for targeting a single element with unique style properties — but it should be avoided when it's possible to use internal or external CSS.
* Let’s say you want to change the color of a key term in a paragraph to a bright orange and leave the other text as is.

|  |  |
| --- | --- |
| <p>Since the default text color for the page is black, the paragraph is black. However, thanks to <span style="color: #FF7A59">inline CSS</span>, I can change the color of a few words within a paragraph to orange to make it really stand out.</p> |  |

**Internal CSS**

* Instead of being placed inside a style attribute, it is placed inside brackets and defined by a CSS selector. This rule set is then wrapped in <style></style> tags and found in the head section of the HTML file.

<!DOCTYPE html>

<html>

<head>

<style>

selector {

CSS property: value;

}

</style>

</head>

* Using **internal CSS** is considered a better practice than using inline CSS.
* **Internal CSS** allows you to **style groups of elements at once** — rather than having to add the same style attributes to elements over and over again.
* If you have a multi-page site and would like to make changes across your site, you would still have to open up each HTML file representing those pages and add or change the internal CSS in each head section. (Or you can use external CSS).
* Let’s say you want to change the text color of every paragraph element on a web page to a navy blue color.

<!DOCTYPE html>

<html>

<head>

<style>

p {

color: #33475B;

}

</style>

</head>

<body>

<h2>Internal CSS Example</h2>

<p>The default text color for the page is black. However I can change the color of every paragraph element on the page using internal CSS.</p>

<p>Using internal CSS, I only need to write one rule set to change the color of every paragraph element.</p>

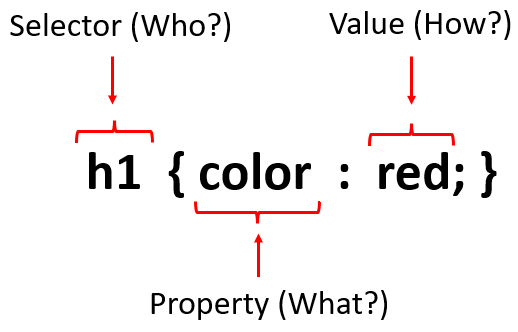
<p>With inline CSS, I'd have to add a style attribute to every single paragraph in my HTML.</p>

</body>

</html>

**Selector**

* In CSS, selectors are patterns used to select the element(s) you want to style.
* See below the anatomy of CSS syntax used to style.



Some of the most used selectors are:

|  |  |  |
| --- | --- | --- |
| **Selector** | **Example** | **Example description** |
| [*.*class](https://www.w3schools.com/cssref/sel_class.asp) | *.intro* | *Selects all elements with class="intro"* |
| .class1.class2 | *.name1.name2* | *Selects all elements with both*name1*and*name2*set within its class attribute* |
| *.class1 .class2* | *.name1 .name2* | *Selects all elements with*name2*that is a descendant of an element with*name1 |
| [*\**](https://www.w3schools.com/cssref/sel_all.asp) | *\** | *Selects all elements* |
| [element](https://www.w3schools.com/cssref/sel_element.asp) | *p* | *Selects all <p> elements* |
| [element,element](https://www.w3schools.com/cssref/sel_element_comma.asp) | *div, p* | *Selects all <div> elements and all <p> elements* |
| [elementelement](https://www.w3schools.com/cssref/sel_element_element.asp) | *div p* | *Selects all <p> elements inside <div> elements* |
| [*:hover*](https://www.w3schools.com/cssref/sel_hover.asp) | *a:hover* | *Selects links on mouse over* |
| [*:nth-of-type(*n*)*](https://www.w3schools.com/cssref/sel_nth-of-type.asp) | *p:nth-of-type(2)* | *Selects every <p> element that is the second <p> element of its parent* |

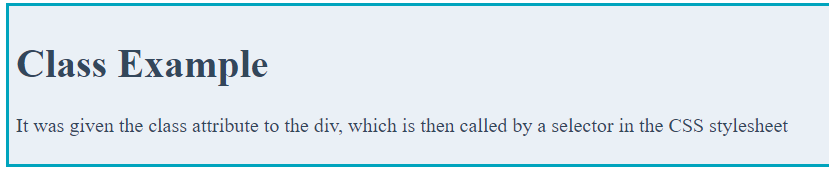
**External CSS**

* **External** **CSS** is formatted like internal CSS, but it isn’t wrapped in <style> tags or placed in the head section of your HTML file. Instead, it’s placed in an external file with the extension “.css.” In the head section, you’ll just have to add a link to this external stylesheet that looks something like:

<link rel="stylesheet" type="text/css" rel="noopener" target="\_blank" href="mystyles.css">

* Using external CSS is considered the **best practice** for a few reasons. It’s also the fastest and most **SEO-friendly**. Storing CSS in another file makes your HTML file **easier** **to** **read** for search engines. It also enables a visitor’s browser to cache the CSS file to load your website faster for their next visit.

|  |  |
| --- | --- |
| **HTML File** | **CSS File** |
| <!DOCTYPE html>  <html>  <head>  <link rel="stylesheet" type="text/css" rel="noopener" target="\_blank" href="mystyle.css">  </head>  <body>  <div class=container>  <h1>Class Example</h1>  <p>It was given the class attribute to the div, which is then called by a selector in the CSS stylesheet <p>  </div>  </body>  </html> | div {  background-color: #EAF0F6;  color: #33475B;  border: 3px solid #00A4BD;  padding: 5px;  } |



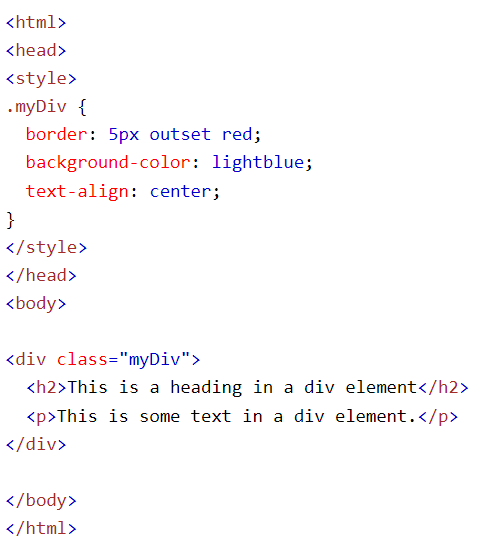
**Classes**

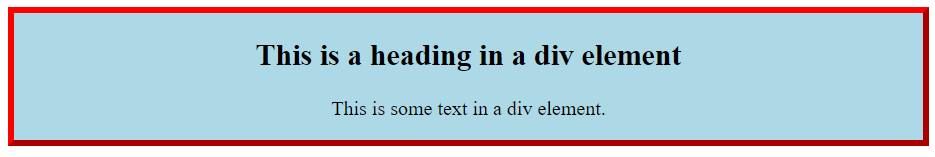
* The **class** **selector** selects elements with a specific class **attribute**. It matches all the HTML elements based on the contents of their class attribute. The **.** symbol, along with the class name, is used to select the desired class.

|  |  |
| --- | --- |
| **HTML File** | **CSS File** |
| <!DOCTYPE html>  <html>  <head>  <link rel="stylesheet" type="text/css" rel="noopener" target="\_blank" href="mystyle.css">  </head>  <body>  <div class=container>  <h1>Class Exampleh1>  <p>It was given the class attribute to the div, which is then called by a selector in the CSS stylesheet p>  </div>  </body>  </html> | .container {  background-color: #EAF0F6;  color: #33475B;  border: 3px solid #00A4BD;  padding: 5px;  } |

**DIV**

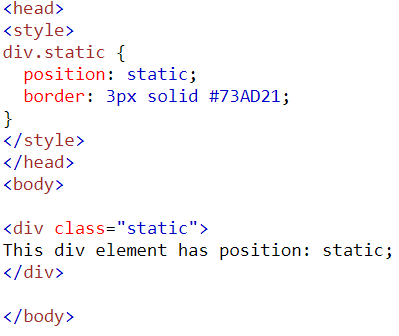
* The <div> tag defines a division or a section in an HTML document.
* The <div> tag is used as a container for HTML elements - which is then styled with CSS or manipulated with JavaScript.
* The <div> tag is easily styled by using the class or id attribute.
* Any sort of content can be put inside the <div> tag!

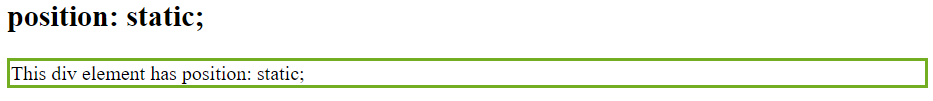




**CSS Positioning – Static**

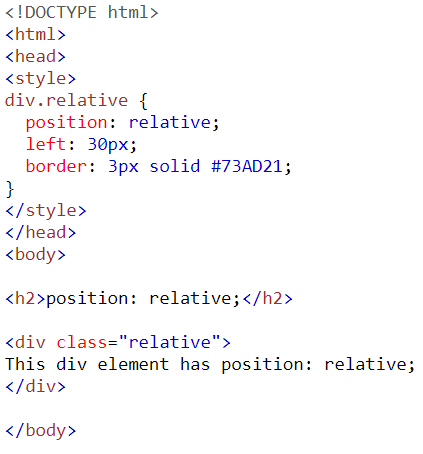
* **HTML** elements are positioned static by default.
* An element with position: static; is not positioned in any special way; it is always positioned according to the normal flow of the page.
* Static positioned elements are not affected by the top, bottom, left, and right properties.

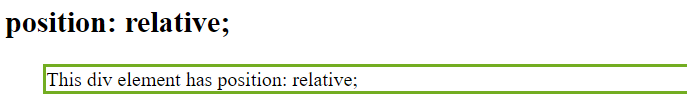




**CSS Positioning – Relative**

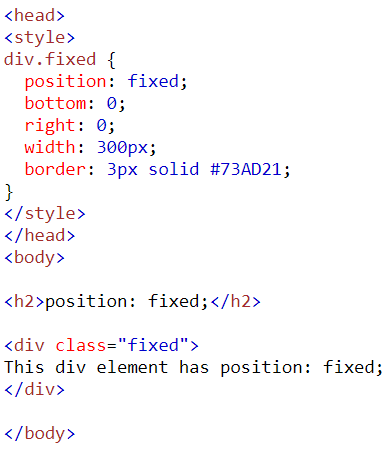
* An element with **position: relative**; is positioned relative to its normal position.
* Setting the top, right, bottom, and left properties of a relatively-positioned element will cause it to be adjusted away from its normal position. Other content will not be adjusted to fit into any gap left by the element.





**CSS Positioning – Fixed**

* An element with **position: fixed**; is positioned relative to the viewport, which means it always stays in the same place even if the page is scrolled.
* The top, right, bottom, and left properties are used to position the element.
* A fixed element does not leave a gap in the page where it would normally have been located.
* Notice the fixed element in the lower-right corner of the page. Here is the CSS that is used:



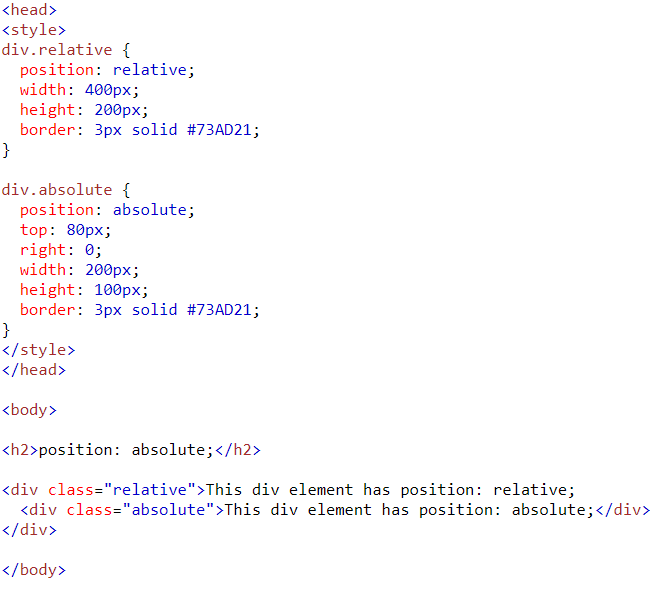


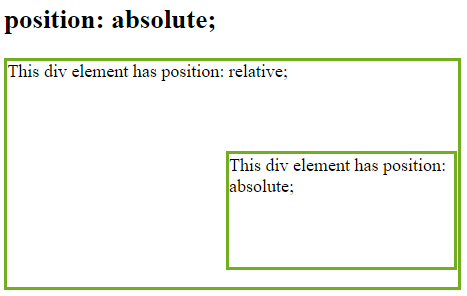
**CSS Positioning – Absolute**

* An element with **position: absolute**; is positioned relative to the nearest positioned ancestor (instead of positioned relative to the viewport, like fixed).
* However; if an absolute positioned element has no positioned ancestors, it uses the document body, and moves along with page scrolling.

Note: Absolute positioned elements are removed from the normal flow, and can overlap elements.

Here is a simple example:





**Font-Sizing**

* The *font-size* property sets the size of the text.
* Being able to manage the text size is important in web design. However, you should not use font size adjustments to make paragraphs look like headings, or headings look like paragraphs.
* Always use the proper HTML tags, like <h1> - <h6> for headings and <p> for paragraphs.
* The font-size value can be an absolute, or relative size.

Absolute size:

* Sets the text to a specified size
* Does not allow a user to change the text size in all browsers (bad for accessibility reasons)
* Absolute size is useful when the physical size of the output is known

Relative size:

* Sets the size relative to surrounding elements
* Allows a user to change the text size in browsers
* **Set Font Size With Pixels**: Setting the text size with pixels gives you full control over the text size:

|  |  |
| --- | --- |
|  |  |

* **Set Font Size With *em*:** To allow users to resize the text (in the browser menu), many developers use *em* instead of pixels. 1*em* is equal to the current font size. The default text size in browsers is 16px. So, the default size of 1em is 16px.

|  |  |
| --- | --- |
|  |  |

* **Use a Combination of Percent and *em***: Specifying the font-size in percent and *em* displays the same size in all major browsers, and allows all browsers to resize the text!

|  |  |
| --- | --- |
|  |  |

* **Responsive Font Size**: The text size can be set with a *vw* unit, which means the "viewport width". That way the text size will follow the size of the browser window:

