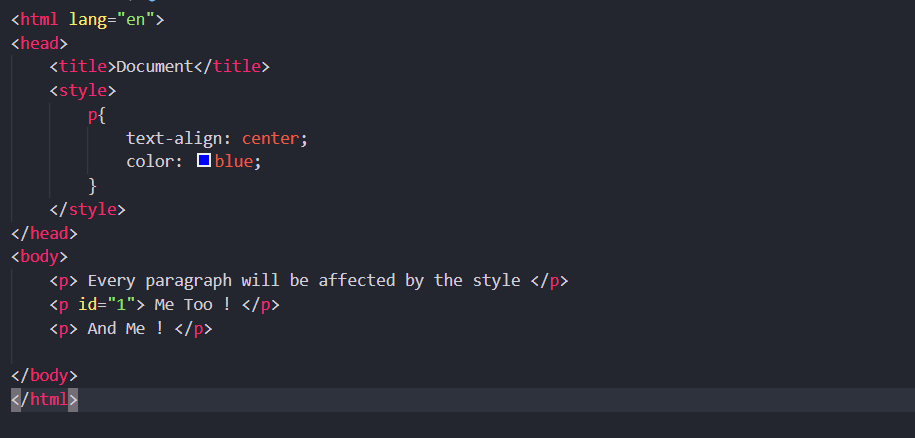
**SELECTORS**

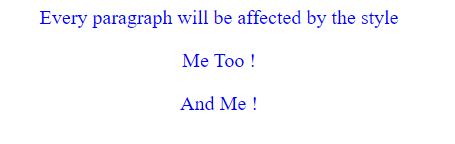
So far, you have seen lots of examples of properties that affect the presentation of text, and you have seen how these properties are applied to elements using style rules. Before we continue to look at more properties, we need to look at some more fundamental issues. We will start by looking at how you can use different types of *selectors* to specify which elements a style sheet rule will apply to before going on to look at the different units of measurements used in CSS.

You can specify which elements a style sheet rule applies to in several ways, not just by using the names of the elements as you have seen so far in this chapter (which is, incidentally, known as a *simple selector*), or using the value of the class attribute or id attribute. Let’ s take a look at the different types of selectors that are available to specify which element(s) a CSS rule applies to.

Basic Example (element selector):



Output:



**Note: that an id and class name cannot start with a number!**

**Universal Selector:**

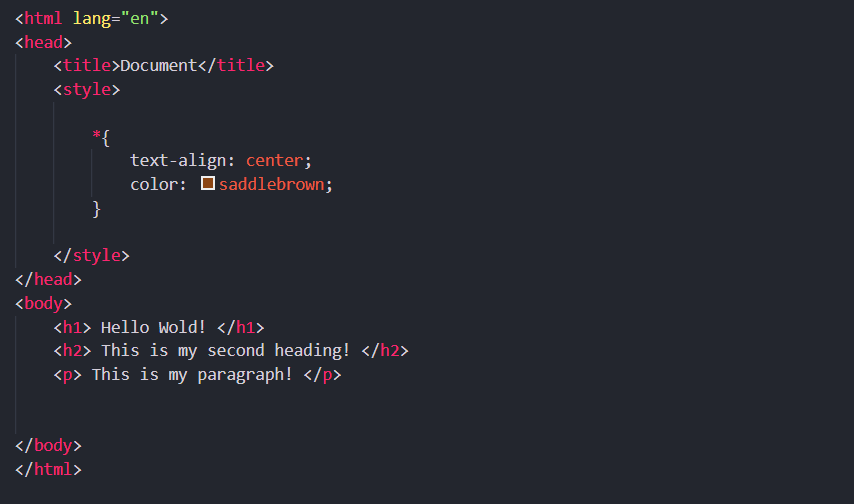
The *universal selector* is an asterisk; it is like a wildcard and matches all element types in the document.

\*{}

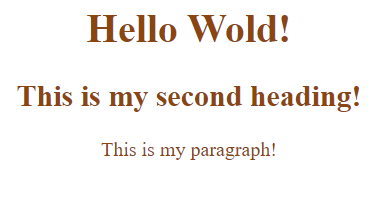
If you want a rule to apply to all elements, you can use this selector. Sometimes it is used for default values that will apply to the whole of the document (such as a font - family and font - size) unless another more specific selector indicates that an element should use different values for these same properties.

It is slightly different from applying default styles to the <body> element, as the universal selector applies to every element and does not rely on the property being inherited from the rules that apply to the <body> element.

Example:



Output:

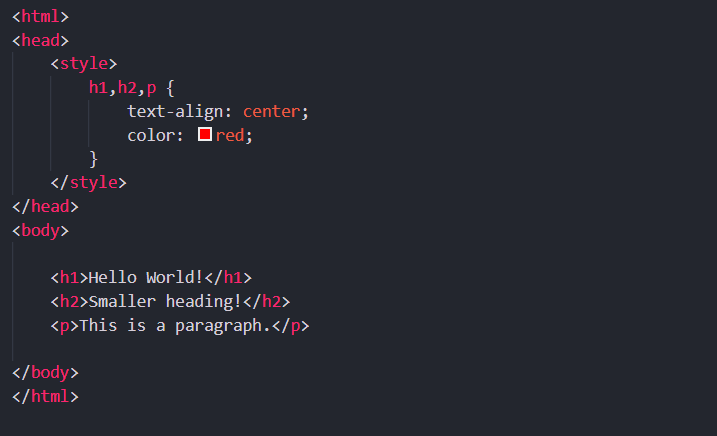


**The Type Selector (or Group Selector):**

The *type selector* matches all of the elements specified in the comma - delimited list. It allows you to apply the same rules to several elements. For example, if you wanted to apply the same rules to different sized heading elements, the following would match all h1, h2, and h3 elements:

h1, h2, h3 {}

Example:



Output:



**The Class Selector:**

The *class selector* allows you to match a rule with an element (or elements) carrying a class attribute whose value matches the one you specify in the class selector. For example, imagine you had a <p> element with a class attribute whose value was BackgroundNote , like so:

<p class = ”BackgroundNote”> This paragraph contains an aside. </p>

You can use a class selector in one of two ways here. One way is to simply assign a rule that applies to any element that has a class attribute whose value is BackgroundNote , like so, preceding the value of the class attribute with a period or full stop:

.BackgroundNote {}

Or you can create a selector that selects only the <p> elements that carry a class attribute with a value of BackgroundNote (not other elements) like so:

p.BackgroundNote {}

If you have several elements that can all carry a class attribute with the same value (for example a <p> element and a <div> element could both use the class attribute with the same value) *and* you want the content of these elements to be displayed in the same manner, you will want to use the former notation. If the styles you are defining are specific to just the <p> element whose class attribute has a value of BackgroundNote, then you should use the latter notation.

A class attribute can also contain several values separated by a space — for example:

<p class = ”important-code”>

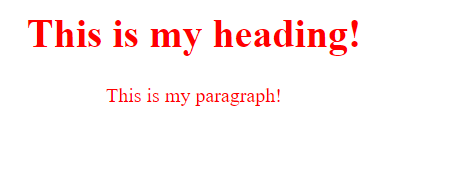
You can use the following syntax to indicate an element that has a class attribute whose value contains both important *and* code (although IE7 was the first version of Internet Explorer to support this syntax).

p.important.code {}

Example:



Output:



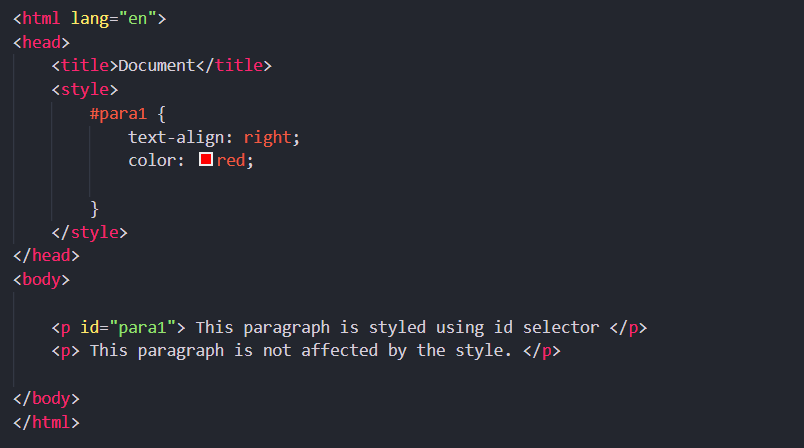
**The ID Selector:**

The *id selector* works just like a class selector, but works on the value of id attributes. Rather than using a period or full stop before the value of the id attribute, you use a hash or pound sign ( # ). So, an element with an id attributes whose value is abstract can be identified with this selector.

#abstract

Because the value of an id attribute should be unique within a document, this selector should apply only to the content of one element (and you should not have to specify the element name).

Example:



Output:



**The Child Selector:**

The *child selector* matches an element that is a **direct child of another**. In this case it matches any <b> elements that are direct children of <td> elements. The names of the two elements are separated by a greater - than symbol to indicate that b is a child of td (>) which is referred to as a *combinator*:

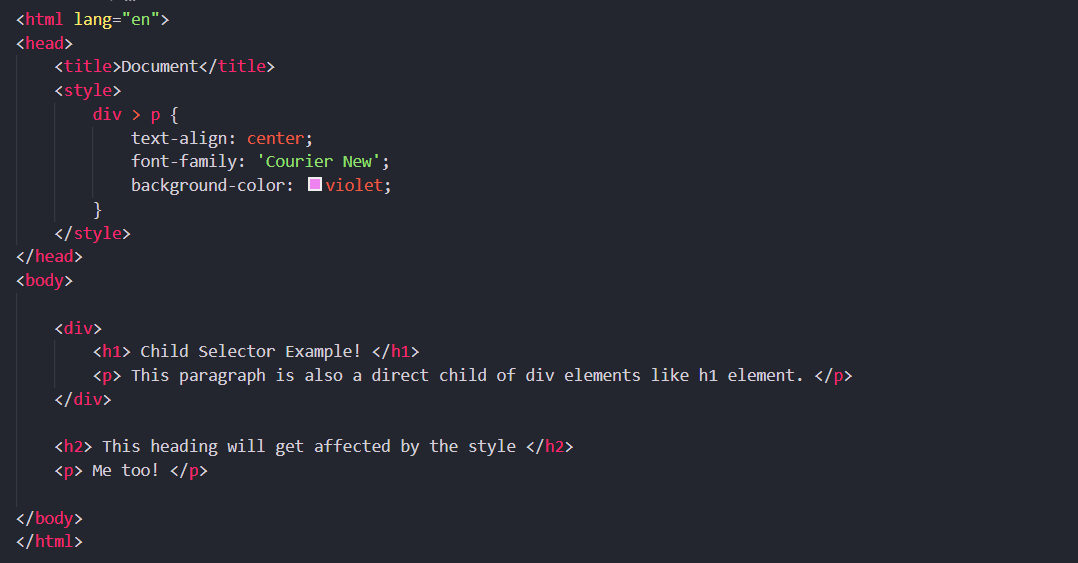
td > b {}

This would enable you to specify a different style for <b> elements that are direct children of the <td> element compared with <b> elements that appear elsewhere in the document.

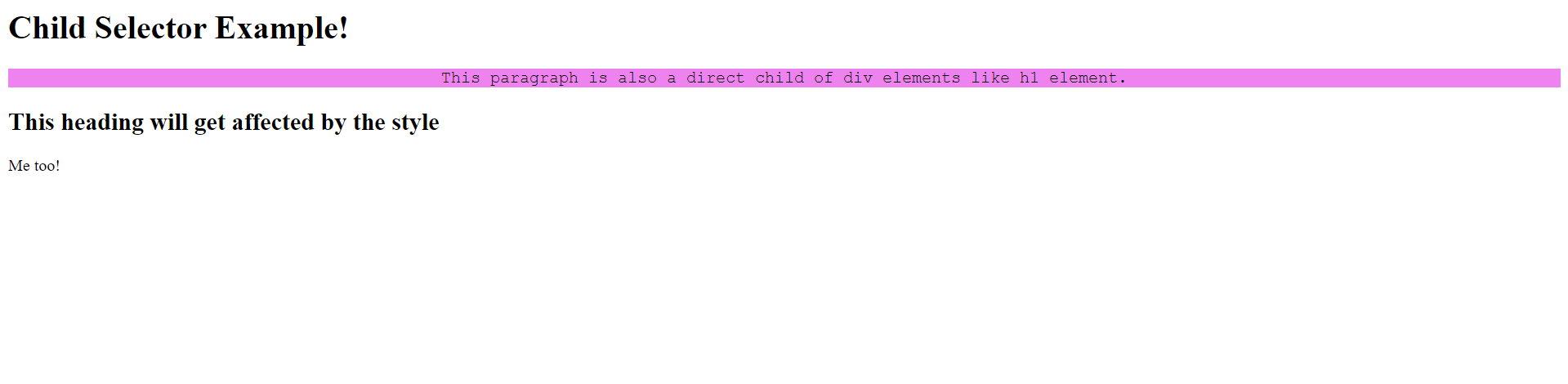
As a direct child of the <td> element, no other tags would sit between the opening <td> tag and the <b> element. For example, the following selector does not make sense because the <b> element should not be a direct child of a <table> element (instead, a <tr> element is more likely to be the direct child of a <table> element):

table > b {}

Example:



Output:



**The Descendant Selector:**

The *descendant selector* matches an element type that is a descendant of another specified element (or nested inside another specified element), **not just a direct child**. While the greater - than symbol is the combinator for the child selector, for the descendent selector the combinator is the space. Take a look at this example:

table b {}

In this case, the selector matches any <b> element that is a child of the <table> element, which means it would apply to <b> elements both in <td> and <th> elements.

This is a contrast to the child selector because it applies to all of the children of the <table> element, rather than just the direct children.

Example:

Output:

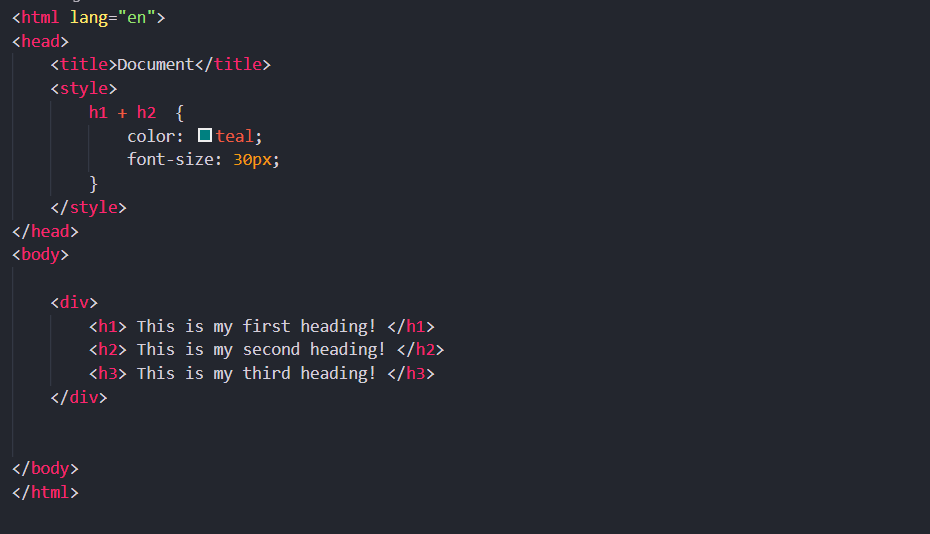


**The Adjacent Sibling Selector:**

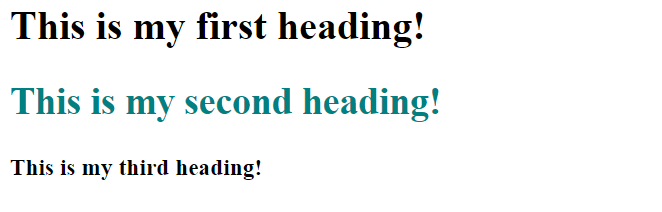
An *adjacent sibling selector* matches an element type that is the next sibling of another. (+) separates two selectors and matches the second element only if it immediately follows the first element, and both are children of the same parent [element](https://developer.mozilla.org/en-US/docs/Web/API/Element). For example, if you want to make the first paragraph after any level 1 heading a different style from other <p> elements you can use the adjacent sibling selector like so to specify rules for just the first <p> element to come after any < h1 > element:

h1+p {}

Example:



Output:



**Note:**

1. Simple Selectors are:
   1. Element selector
   2. Id selector
   3. Class selector
   4. Universal selector
   5. Grouping or type selector
2. Combinator Selectors are:
   1. Child selector
   2. Descendant selector
   3. Adjacent sibling selector
   4. General sibling selector
3. Other Selectors are:
   1. Attribute selectors
   2. Pseudo-class selectors
   3. Pseudo-elements selectors