**The General Sibling Selector**

The general sibling selector matches an element type that is a sibling of another, although it does not have to be the directly preceding element. So, if you had two <p> elements that are siblings of an <h1> element, they would both use the rules of this selector.

h1~p {}

The general sibling selector is part of CSS3; IE7 was the first version of Internet Explorer to support the general sibling selector and Firefox 2 was the first version of Firefox to support it.

**Using Child and Sibling Selectors to Reduce Dependence on Classes in Markup:**

The child and adjacent sibling selectors are both very important because they can reduce the number of class attributes you need to add into an HTML document.

It is very easy to add classes for all kinds of eventualities. For example, if you wanted the first paragraph after an <h1> element to be shown in bold, you might have been tempted to add a class attribute to the first <p> element after every <h1> element. While this would work, your markup can be littered with all kinds of classes that are only there to make it easier to control the presentation of the pages.

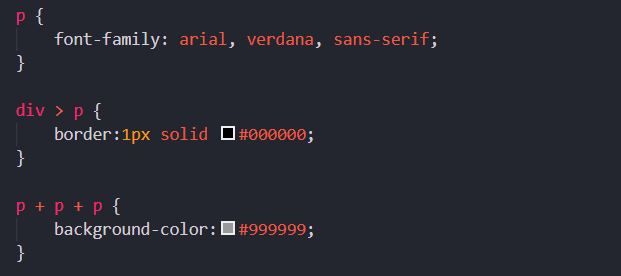
If you then decided you wanted the first two <p> elements after every <h1> element to be bold, you might have to go back and add in new class attributes for the second <p> elements after every <h1> element. So, the child and adjacent sibling selectors add a lot of flexibility to how you style documents and can make for much cleaner markup.

Take a look at the following HTML content:

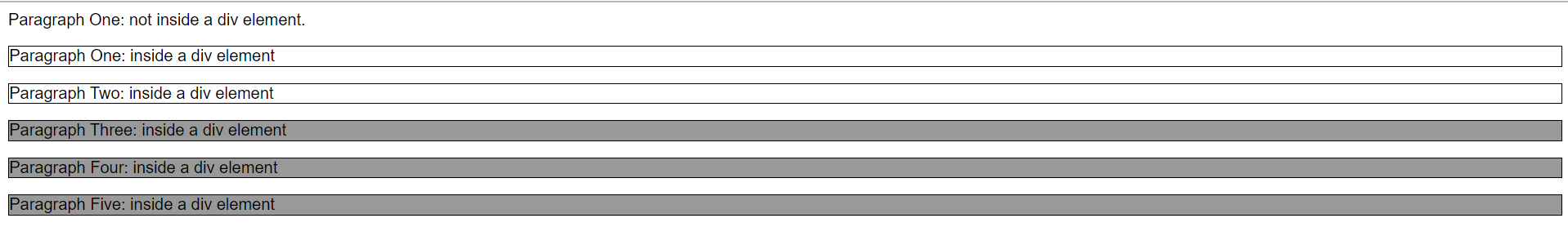
Example: (html file)



css file:



Output: (Using the adjacent and adjacent sibling and child selectors only, you are going to create a page that looks like the one shown in Figure)



The three different paragraph styles are as follows:

* The first paragraph has no border or background color.
* The paragraphs inside the <div> element all have borders.
* The last three paragraphs have a gray background as well as their border.

I have not used three different classes to specify different paragraph styles; rather, I have one rule that controls the font used for all paragraphs:

p { font-family:arial, verdana, sans-serif; }

The following is the second rule for any paragraph that is a child of a < div > element. (Because the first paragraph is not inside a <div> element, the rule does not apply to the first paragraph.)

div > p { border:1px solid #000000; }

The third rule matches any paragraph that is also a third consecutive <p> element. (Because the fourth and fifth <p> elements have two previous <p> elements, this rule applies to them, as well as the third <p> element inside the <div> .)

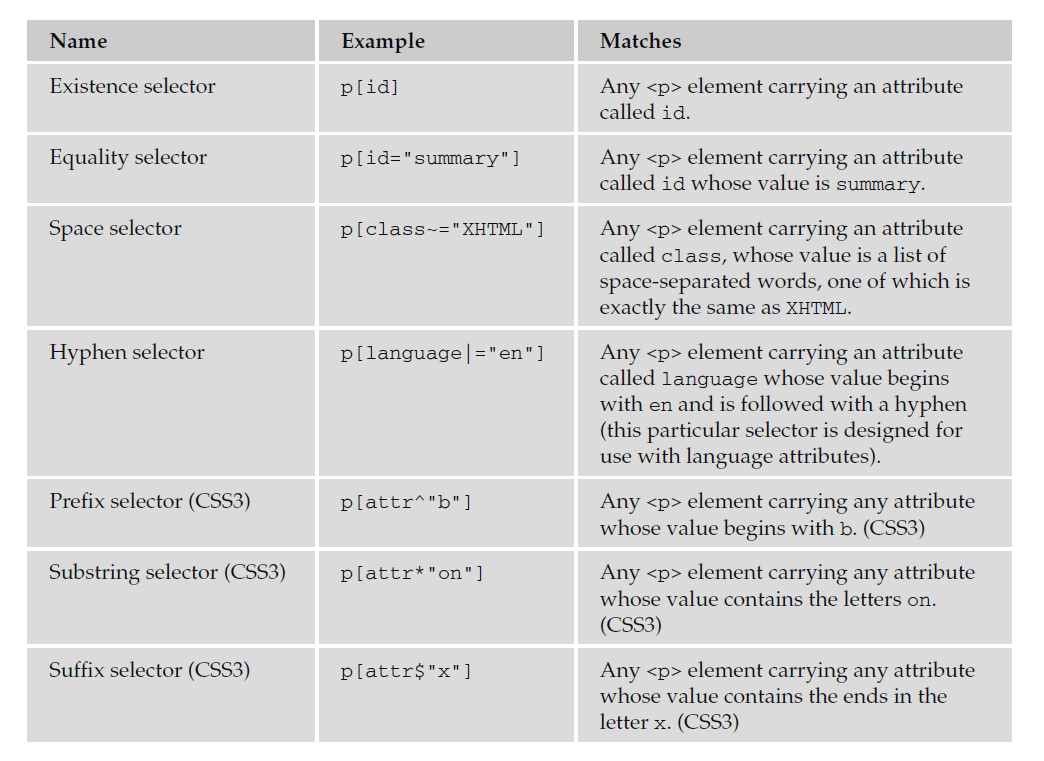
p+p+p { background-color:#999999; }

Remember that this example will not work in IE6 or earlier versions of Internet Explorer, as these selectors were first introduced in IE7.

**Attribute Selectors:**

Attribute selectors enable you to use the attributes that an element carries, and their values, in the selector. There are several types of attribute selector and they allow far more complex ways of selecting elements in a document.

The use of attribute selectors is fairly limited because they have only been supported in the latest versions of browsers. Some of the attribute selectors in the following table are from CSS3, which has not yet been completed.

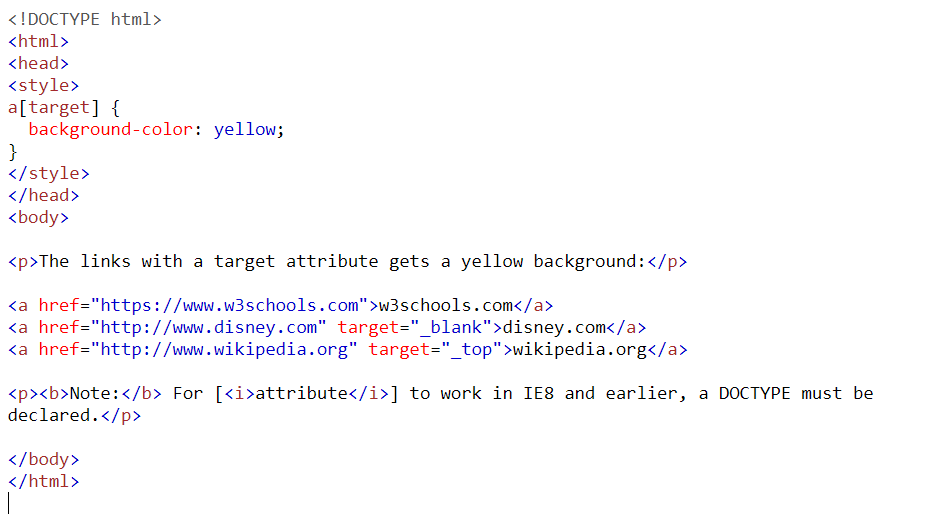


Internet Explorer implemented these attribute selectors in IE7, and in order for them to work,

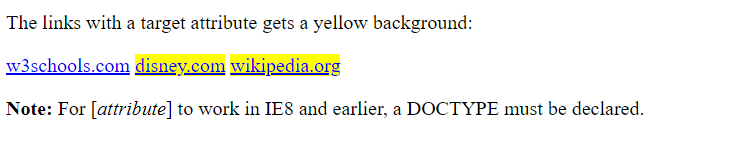
the XHTML document must have the strict !DOCTYPE declaration. Firefox started to support them in Firefox 2.

1. **Existence Selector:**

Example:

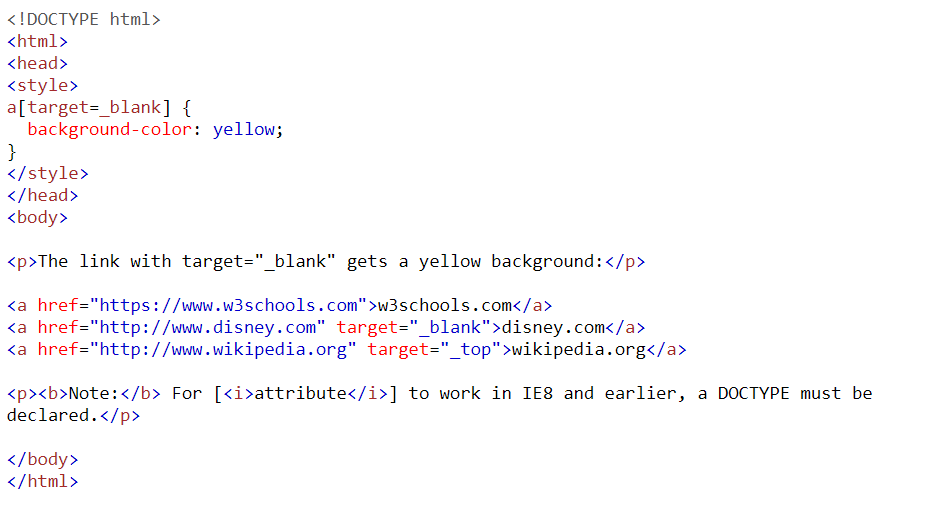


Output:

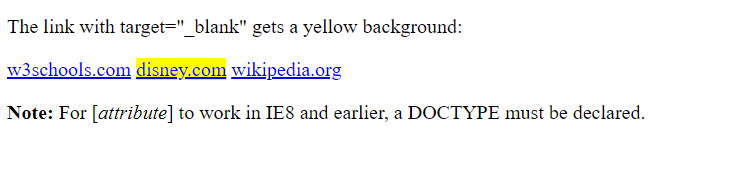


1. **Equality Selector:**

Example:

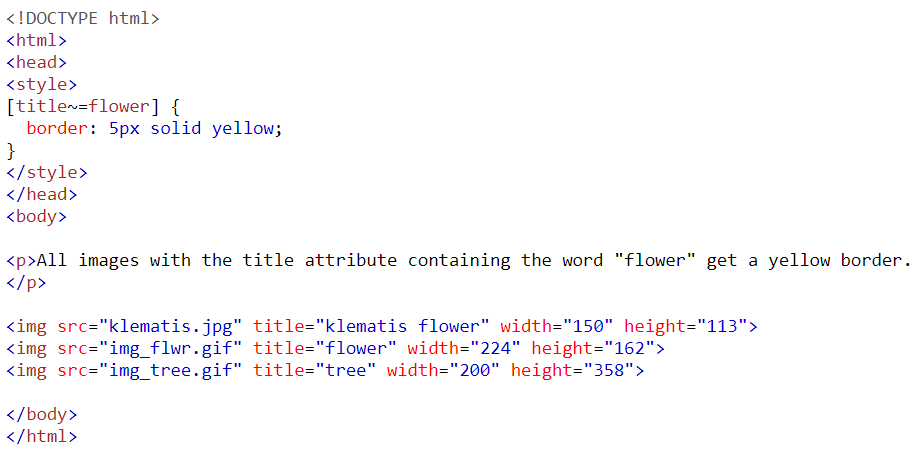


Output:

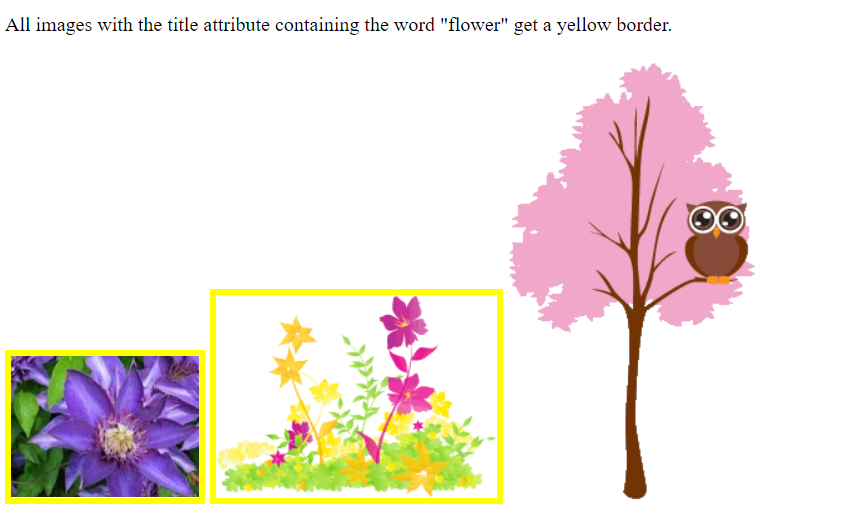


1. **Space Selector:**

Example:

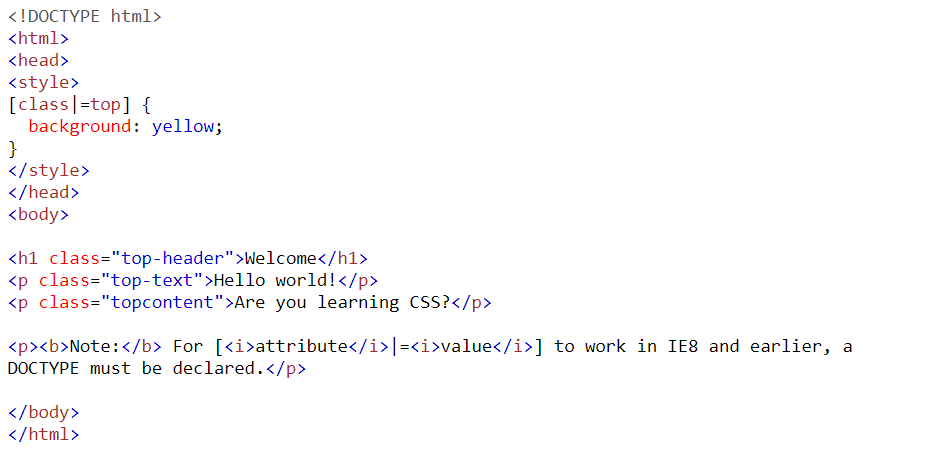


Output:

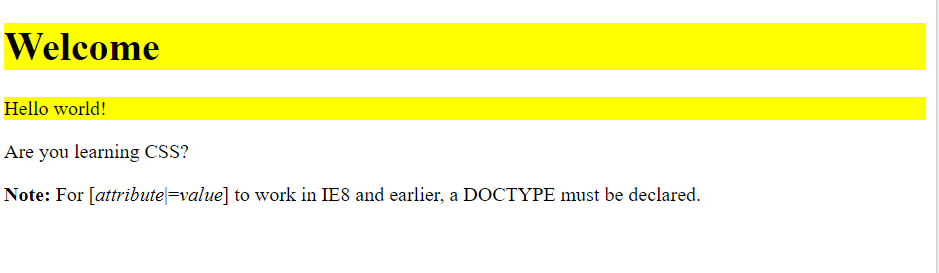


1. **Hyphen Selector:**

Example:

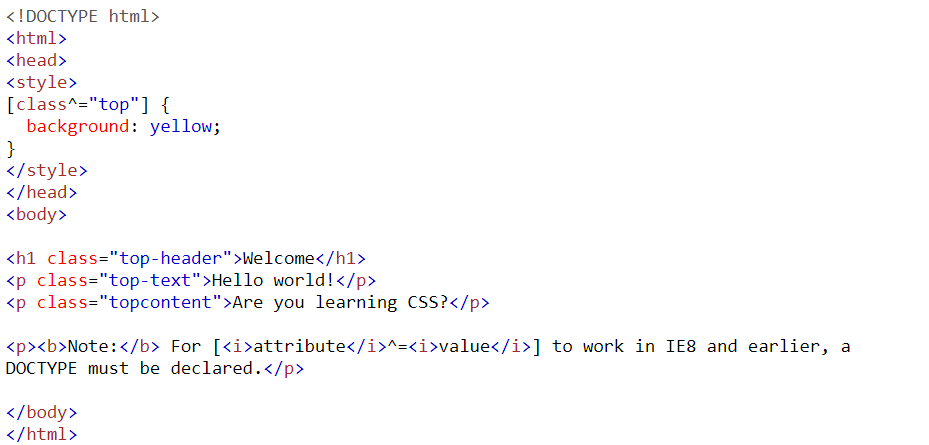


Output:

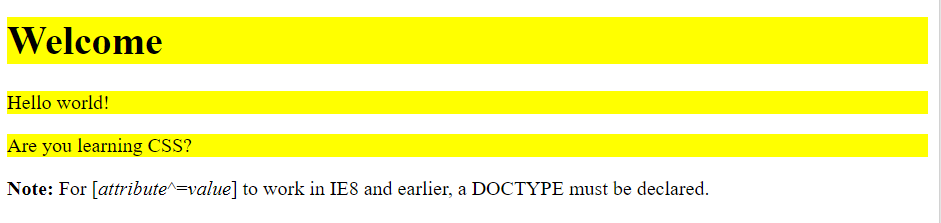


1. **Prefix Selector:**

Example:

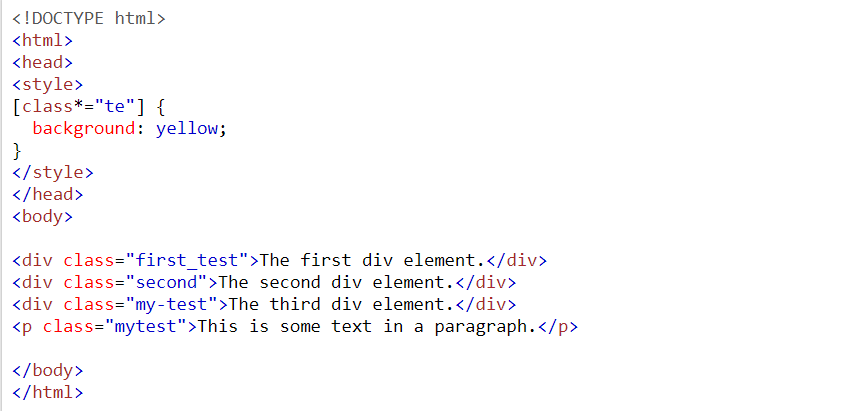


Output:



1. **Substring Selector:**

Example:

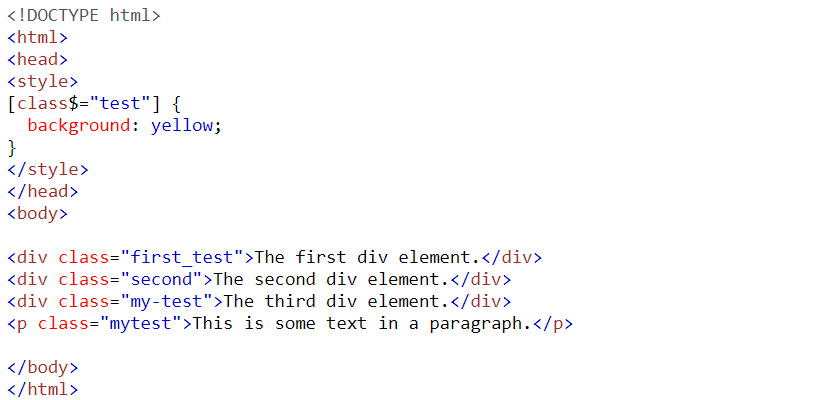


Output:



1. **Suffix Selector:**

Example:



Output:

