

# CSS Intermediate Guide

html5dog.com

Like the HTML Intermediate Guide, this CSS Intermediate Guide should not be that difficult, but rather build on the basics of the CSS Beginner's Guide.

## Class and ID Selectors

For the CSS Beginner's Guide we looked solely at HTML selectors - those that represent an HTML tag.

You can also define your own selectors in the form of **Class** and **ID** selectors.

The benefit of this is that you can have the same HTML element, but present it differently depending on its class or ID.

In the CSS, a class selector is a name preceded by a **full stop** (`.`) and an ID selector is a name preceded by a **hash character** (`#`).

So the CSS might look something like:

```
#top {background-color: #ccc;padding: 1em}.intro {color: red;font-weight: bold;}
```

The HTML refers to the CSS by using the attributes **id** and **class**. It could look something like this:

```
<div id="top"><h1>Chocolate curry</h1><p class="intro">This is my recipe for making curry purely with chocolate</p><p class="intro">Mmm mm mmmmm</p></div>
```

The difference between an ID and a class is that an ID can be used to identify one element, whereas a class can be used to identify more than one.

You can also apply a selector to a specific HTML element by simply stating the HTML selector first, so `p.jam { whatever }` will only be applied to *paragraph* elements that have the class 'jam'.

## Grouping and Nesting

### Grouping

You can give the same properties to a number of selectors without having to repeat them by separating the selectors by **commas**.

For example, if you have something like:

```
h2 {
  color: red;
}
.thisOtherClass {
  color: red;
}
.yetAnotherClass {
  color: red;
}
```

You could make it:

```
h2, .thisOtherClass, .yetAnotherClass {
  color: red;
}
```

## Nesting

If the CSS is structured well, there shouldn't be a need to use many class or ID selectors. This is because you can specify properties to selectors *within* other selectors.

For example:

```
#top {
  background-color: #ccc;
  padding: 1em
}
#top h1 {
  color: #ff0;
}
#top p {
  color: red;
  font-weight: bold;
}
```

Removes the need for classes or ID's if it is applied to HTML that looks something like this:

```
<div id="top">
<h1>Chocolate curry</h1>
<p>This is my recipe for making curry purely with chocolate</p>
<p>Mmm mm mmmmm</p>
</div>
```

This is because, by separating selectors with **spaces**, we are saying '**h1** inside ID **top** is colour **#ff0**' and '**p** inside ID **top** is **red** and **bold**'.

This can get quite complicated (because it can go for more than two levels, such as this inside this inside this inside this etc.) and may take a bit of practice.

## Pseudo Classes

**Pseudo classes** are bolted on to selectors to specify a state or relation to the selector. They take the form of **selector:pseudo class { property: value; }**, simply with a **colon** in between the selector and the pseudo class.

Many CSS proposals are not supported by all browsers, but there are four pseudo classes that can be used safely when applied to links.

- \_ **link** is for an unvisited link.
- \_ **visited** is for a link to a page that has already been visited.
- \_ **active** is for a link when it gains focus (for example, when it is clicked on).
- \_ **hover** is for a link when the cursor is held over it.

```
a.snowman:link {
    color: blue;
}
a.snowman:visited {
    color: purple;
}
a.snowman:active {
    color: red;
}
a.snowman:hover {
    text-decoration: none;
    color: blue;back
    ground-color: yellow;
}
```

### Note

Although CSS gives you control to bypass it, maintaining different colours for visited links is good practice. As pseudo classes (other than **hover**) are not often used, this is a feature that is not as common as it once was. Because of this, it is less important than it used to be, but if you are looking for the optimum user response, then you *should* use it.

Traditionally, text links were **blue** if not visited and **purple** if visited, and there is still reason to believe that these are the most effective colours to use, although, again, with the increasingly widespread use of CSS, this is becoming less commonplace and the average user no longer assumes that links must be blue or purple.

## Note

You should also be able to use the **hover** pseudo class with elements other than links. Unfortunately, Internet Explorer doesn't support this method. This is a bloody irritation because there are lots of nice little tricks you can do that look delightful in other browsers.

### Shorthand Properties

Some CSS properties allow a string of values, replacing the need for a number of properties. These are represented by values separated by **spaces**.

**margin**, **padding** and **border-width** allow you to amalgamate **margin-top-width**, **margin-right-width**, **margin-bottom-width** etc. in the form of **property: top right bottom left;**

So:

```
p {  
  border-top-width: 1px;  
  border-right-width: 5px;  
  border-bottom-width: 10px;  
  border-left-width: 20px;  
}
```

Can be summed up as:

```
p {  
  border-width: 1px 5px 10px 20px;  
}
```

**border-width**, **border-color** and **border-style** can also be summed up as, for example:

```
p {  
  border: 1px red solid;  
}
```

(This can also be applied to **border-top**, **border-right** etc.)

By stating just two values (such as **margin: 1em 10em;**), the first value will be the top and bottom and the second value will be the right and left.

Font-related properties can also be gathered together with the **font** property:

```
p {  
  font: italic bold 1em/1.5 courier;  
}
```

(Where the '/1.5' is the line-height)

So, to put it all together, try this code:

```
p {  
  font: 1em/1.5 "Times New Roman", times, serif;  
  padding: 3em 1em;  
  border: 1px black solid;  
  border-width: 1px 5px 5px 1px;  
  border-color: red green blue yellow;  
  margin: 1em 5em;  
}
```

Lovely.

## ***Background Images***

There are a number of properties involved in the manipulation of **background images**.

Luckily, the property **background** can deal with them all.

```
body {  
  background: white  
  url(http://www.html5dog.com/images/bg.gif)  
  no-repeat top right;  
}
```

This amalgamates these properties:

- \_ **background-color**, which we have come across before.
- \_ **background-image**, which is the location of the image itself.
- \_ **background-repeat**, which is how the image repeats itself. This can be **repeat** (equivalent to a 'tile' effect across the whole background), **repeat-y** (repeating on the 'y-axis', above and below), **repeat-x** (repeating on the 'x-axis', side-by-side) or **no-repeat** (which shows just one instance of the image).
- \_ **background-position**, which can be **top**, **center**, **bottom**, **left**, **right** or any sensible combination, such as above.

Background-images can be used in most HTML elements - not just for the whole page (body) and can be used for simple but effective results, such as shaped corners.

## **Note**

It is easy to get carried away with background images and plaster them all over your web pages. Some visually hyperactive people might believe it looks good to have a full-on brightly coloured photograph tiled across the background of a page, giving the user a Mensa-esque challenge in deciphering the foreground text. This is an extreme example, but the fact is that the most user-friendly, readable text is black on a plain white background or white on a plain black background (there is also a suggestion that a slightly off-white or off-black background is better as this reduces glare).

So, the best use of background images is either to use them where there will be no content over the top or making the background image very light, which would also reduce the file size of the image, because there should be less colours involved (supposing you are using an indexed-colour format, such as GIF).