**CSS: An Overview (Cascading Style Sheets)**

There are two ways to put CSS in one place. This first is to put your CSS between <style></style> tags, right in the same file as your HTML. These <style> tags go inside the <head></head> of your webpage; or by putting a <link> tag between the <head>...</head> tags of your HTML page. Your <link> tag needs three attributes:

1. A type attribute that should always be equal to "text/css"
2. A rel attribute that should always be equal to "stylesheet"
3. A href attribute that should point to the web address of your CSS file

<link type="text/css" rel="stylesheet" href="stylesheet.css"/>

**The importance of semicolons**

As you start adding more and more property-value pairs for each CSS selector, it's important to remember to put a semicolon (;) at the end of each line.

The semicolon tells CSS that one property-value pair is over and it's time to move on to the next one. Without semicolons, it'll become confused and your page won't look right.

Also, don't forget: all property-value pairs for a selector are surrounded by curly braces ({}).

Comments

It's a good idea to write **comments** as you go along. Good comments will help remind you why you did something a certain way (or will help someone else out if they're reading your code without you there to explain it).

CSS comments look like this: /\*I'm a comment!\*/

**Hexadecimal Colours**

There are a lot of tools available on the Internet for looking up hexadecimal (or simply **hex**) color values.

Search for "hex color palette" or "hex color picker" with your favorite web browser to find a bunch of options!

Hex values always start with a pound sign (#), are up to six "digits" long, and are **case-insensitive**: that is, they don't care about capitalization. #FFC125 and #ffc125 are the same color.

**Background color, height, and width**

<!DOCTYPE html>

<html>

<head>

<link type="text/css" rel="stylesheet" href="stylesheet.css"/>

<title>Result</title>

</head>

<body>

<div style="background-color:#cc0000;height:100px;width:100px" </div> /\*produces a red 100px2 block\*/

</body>

</html>

**Multiple Selectors**

If you want to grab <p>s that are inside two <div>s, and not all <p>s?

You select those in the CSS tab like this:

div div div h3

{

color:red; /\*h3 header 4th line down\*/

}

There's also a very special selector you can use to apply CSS styling to every element on the page: the \* selector. For example, if you type

\* {

border: 2px solid black;

}

/\*You'll create a two-pixel wide solid black border around every element on the HTML page.\*/

**Classes and ID’s**

<!DOCTYPE html>

<html>

<head>

<link type="text/css" rel="stylesheet" href="stylesheet.css"/>

<title>Result</title>

</head>

<body>

<h3 class="red">I'm an h3 header in the red class!</h3>

<h3>I'm just a regular old h3 header.</h3>

<p class="red">I'm a paragraph in the red class!</p>

<p>I'm just a regular old paragraph.</p>

<p id="rogue">I'm a rogue paragraph! I do what I want!</p>

</body>

</html>

**Links**

There are a number of useful pseudo-class selectors for links, including:

a:link: An unvisited link.  
a:visited: A visited link.  
a:hover: A link you're hovering your mouse over.

**Nth child**

You can actually select any child of an element after the first child with the pseudo-class selector nth-child; you just add the child's number in parentheses after the pseudo-class selector. For example,

p:nth-child(2) {

color: red;

}

Example exercise.

a:hover{

text-decoration:none;

}

a:first-child

{

color:#CDBE70;

}

a:nth-child(3) {

color:#FFC125;

}

**Class and ID selectors to visually separate different classes**

<!DOCTYPE html>

<html>

<head>

<link type="text/css" rel="stylesheet" href="stylesheet.css"/>

<title>My Social Network</title>

</head>

<body>

<div class="friend" id="best\_friend"><p>Jurgen</p></div>

<div class="defender"><p>Nathaniel</p></div>

<div class="defender"><p>Dejan</p></div>

<div class="defender"><p>Kolo</p></div>

<div class="defender"><p>Alberto</p></div>

<div class="midfielder"><p>James</p></div>

<div class="midfielder"><p>Emre</p></div>

<div class="midfielder"><p>Adam</p></div>

<div class="midfielder"><p>Phillipe</p></div>

<div class="forward"><p>Daniel</p></div>

<div class="forward"><p>Roberto</p></div>

<div class="friend"><p>Michael</p></div>

<div class="family"><p>Eric</p></div>

<div class="enemy" id="archnemesis"><p>Fergie</p></div>

</body>

</html>

**The Box Model**



Image courtesy of [www.w3.org](http://www.w3.org)!

**TM**, **TB**, and **TP** in the diagram. These stand for "top margin," "top border," and "top padding."

Each HTML element gets its own box to live in.

As you saw, the outermost box of each element went all the way across the page. This is why until now, your HTML elements have been sitting on top of one another: by default, they take up the full width of the page.

**block**: This makes the element a block box. It won't let anything sit next to it on the page! It takes up the full width.

*div {*

*display: block;*

**inline-block**: This makes the element a block box, but will allow other elements to sit next to it on the same line.

*div {*

*display: inline-block;*

**inline**: This makes the element sit on the same line as another element, but without formatting it like a block. It only takes up as much width as it needs (not the whole line).

**none**: This makes the element and its content disappear from the page entirely!

**Margin top, right, bottom, left**

margin: 20px 50px 10px 5px; /\* Always in this order. Use zeros if needed\*/

Will set a top margin of 20px, a right margin of 50px, a bottom margin of 10px, and a left margin of 5px.

**Padding**

Good! Let's adjust the padding. Remember, the padding is the space between your border and your innermost layer: the actual content

div {

height: 50px;

width: 100px;

border: 4px solid #FF0000;

border-radius: 5px;

background-color: #308014;

margin:20px 50px 10px 5px;

padding:40px;

}

**Floats**

To determine where your elements go on the page we can use floats. This tells the webpage where to position the element but within the flow of other elements.

div {

height: 300px;

width: 100px;

border: 2px solid black;

border-radius: 5px;

background-color: #308014;

float:left;

}

We can use floated elements to naturally divide our pages into different sections.

div {

height: 300px;

width: 300px;

border: 2px solid black;

border-radius: 5px;

background-color: #308014;

float:right;

}

p {

font-family: Verdana, sans-serif;

font-size: 20px;

width: 280px;

float:left;

}

**Clearing elements**

If you tell an element to clear: left, it will immediately move below any floating elements on the left side of the page; it can also clear elements on the right. If you tell it to clear: both, it will get out of the way of elements floating on the left *and* right!

The syntax is what you're used to:

element {

clear: /\*right, left, or both\*/

}

Example:

**HTML code**

<!DOCTYPE html>

<html>

<head>

<link type="text/css" rel="stylesheet" href="stylesheet.css"/>

<title>Result</title>

</head>

<body>

<div id="header"></div>

<div class="left"></div>

<div class="right"></div>

<div id="footer"></div>

</body>

</html>

**CSS Code**

div {

border-radius: 5px;

}

#header {

height: 50px;

background-color: #F38630;

margin-bottom: 10px;

}

.left {

height: 300px;

width: 150px;

background-color: #A7DBD8;

float: left;

margin-bottom: 10px;

}

.right {

height: 300px;

width: 450px;

background-color: #E0E4CC;

float: right;

margin-bottom: 10px;

}

#footer {

height: 50px;

background-color: #69D2E7;

clear:both;

}

**Positioning**

**Absolute positioning**

The first type of positioning is absolute positioning. When an element is set to position: absolute, it's then positioned in relation to the first parent element it has that doesn't have position: static. If there's no such element, the element with position: absolute gets positioned relative to <html>

**Relative positioning**

**Relative** positioning is more straightforward: it tells the element to move relative to where it would have landed if it just had the default static positioning.

If you give an element relative positioning and tell it to have a margin-top of 10px, it doesn't move down ten pixels from any particular thing—it moves down ten pixels from where it *otherwise would have been*.

**Fixed positioning anchors an element to the browser window—you can think of it as gluing the element to the screen. If you scroll up and down, the fixed element stays put even as other elements scroll past.**

Declare the correct CSS property on the #inner and #container divs so that the #inner div is positioned relative to the parent element.

**div#container {**

**border: 1px solid black;**

**height: 150px;**

**margin: 60px;**

**width: 100px;**

**position: relative;**

**}**

**div#inner {**

**background-color: black;**

**height: 30px;**

**left: 20px;**

**top: 40px;**

**width: 30px;**

**position: absolute;**

**}**

Declare the correct CSS property on the #inner div so that it is positioned relative to the browser window and not relative to the parent element.

**div#container {**

**border: 1px solid black;**

**height: 150px;**

**margin: 60px;**

**width: 100px;**

**}**

**div#inner {**

**background-color: black;**

**height: 30px;**

**left: 20px;**

**top: 40px;**

**width: 30px;**

**position: absolute;**

**}**