

CSS Tutorial Part 2:

Lists:

The default style for an unordered (bulleted) list is the bullet, or dot. You can change the style to either a square or a circle as follows:

```
ul {  
  list-style-type: circle;  
}  
or
```

```
ul {  
  list-style-type: square;  
}
```

Equally, you can change the bullet to an image as follows:

```
ul {  
  list-style-image: url("smallstar.gif");  
}
```

Make sure that either the image is in the same folder as the html page or that you include a path to the image, in the same manner in which we included images and links to other pages before.

With lists, people often center the elements, and then want to center the bullets with the elements as well. This is done as follows:

```
ul {  
  list-style-position: inside;  
}
```

(The default list bullet position is outside. In other words, if there was a border, the bullets would be in the margin. But sometimes people want them in the padding section. That's what inside does).

Links

We might want to style our links as well. We can style everything about a link that we can style about everything else – border, background-color, padding, font color, etc. We can style the:

- link (before we've visited the page the link links to)
- visited (the style of the link after we've visited the page)
- hover (what the link looks like when we run our mouse over it)

Step 1: Open your css stylesheet.

Step 2: Add a style for the basic (unvisited) link (note: no space between a:link):

```
a:link {
```

```
    color: #AA1300;
    font-size: 110%;
}
```

Step 3: Now add a different style for links you've visited)

```
a:visited {
    color: #649900;
    font-size: 110%;
}
```

Step 4: Now (and this is cool) add a style for links when you hover the mouse over them):

```
a:hover {
    color: #FF3300;
    font-style: italic;
    font-size: 120%;
}
```

Step 5: Save the css file and reload the html file. Run your mouse over a link. See what happens. Click on one, then go back to the web page and see what it looks like. Feel free to play with the links a bit. Add a background color or image to the "hover" style. Couldn't you have fun with this?

Div versus Span, Class versus Id:

Div:

So far you've added style to preexisting html tags, such as paragraphs (<p></p>), headers (<h1></h1>), etc. You can also create your own elements to be styled.

Specifically, what if you want to set a style for all the elements within one section of a web page (e.g., the header, the paragraphs, the lists, the links, etc. in a side panel on a web page). You can do that using "div" tags.

Inside the Div tag, we give it an identifier. The identifier can either be a class or an id. For now we'll use an id

NOTE: every id MUST BE UNIQUE on your web page. A class can exist many times throughout the web page.

Div tags allow you to section off parts of your html. You place div tags around the area in your html page you want to style. So first, you must edit the html page.

In your html file, add a div tag around a section you want to style:

```
<div id = "sidebar">
  <h2>Find out more about giraffes!
</h2>
  <p> this is where I'd put a lot of info about giraffes </p>
  <h2> Relevant links </h2>
  <p><a href = "http://www.sandiegozoo.org/animalbytes/t-giraffe.html">
    San Diego Zoo: Giraffe</a><br>
</p>
</div>
```

Save your html file. Now you can give the "sidebar" div style in your css stylesheet.

In your.css file, add a style for the sidebar div. To do this, you should add the following:

```
#sidebar{
    background-color:#FFFF66;
    width: 295px;
    color: #5E2605;
}
```

Save the CSS file and reload the html file in the browser to see what you've got so far. To be safe, validate your css stylesheet by going to:

<http://jigsaw.w3.org/css-validator/>

and validating the stylesheet. If you don't get error messages, your page is valid.

Styling an element just within an id element

What if you want the paragraphs on your web page to look one way, but the paragraphs within the div to have a different look than those on the rest of the page? You can set styles for specific tags within a div by first specifying the div's id, and then specifying the element, e.g.,

```
#sidebar p {
    color: #5E2605;
    line-height: 170%;
    padding-top: 15px;
    padding-bottom: 5px;
    margin: 0px;
    width: 300px;
}
```

Span:

A div tag goes around other tags, e.g.,:

```
<div>
    <h1> animals </h1>
    <p> A paragraph about animals goes here </p>
    <h2> list of my favorite animals </h2>
    <ol><li> wombats</li>
        <li> echidnas </li>
        <li> blue footed boobies </li>
    </ol>
</div>
```

On the other hand, span tags go inside other tags, e.g.,:

```
<p> I might put a span tag around a <span> word </span> I want to create a special style for </p>
```

In other words, the span tag goes inside of other html tags so you can style just a word or a few words if you want to. A div tag goes outside and around a bunch of other tags, so you can style a bunch of tags with one style.

Id versus class

Now to style these tags, I need to either give them an id or a class. I can give both div tags and span tags ids or I can give either a class name. Equally, I can give any tag an id or make it belong to a class.

The difference between the two: each id must be unique on a web page, whereas more than one tag can belong to the same class.

So, for instance, I could have:

```
<div id = "firsttag">
```

```
...
```

```
</div>
```

Or even

```
<p id = "firstparatag"> ... </p>
```

And then to style the id, I'd include the following in my css:

```
#firsttag {  
    font-family: arial;  
}
```

```
#firstparatag{  
    font-size: small;  
}
```

I can also make different tags belong to a class:

```
<div class = "emphclass">
```

```
...
```

```
</div>
```

Or

```
<p> I might put a span tag around a <span class = "emphclass"> word </span> I want to create a special style for </p>
```

Or

```
<h2 class = "emphclass"> Stuff I Want Emphasized </h2>
```

To style a class, in my css I'd do the following:

```
.emphclass {  
    color: #3278F4;  
}
```

So to summarize: Div and Span tags are tags we add to the html code. Div tags go outside and around other tags, whereas span tags go inside of other tags.

Ids and classes are ways of identifying the style we want to give to a particular tag or set of tags. Ids must be unique on your web page, so we reserve ids for styles you'd only need to use once on your web page. More than one tag can have the same class. So the style you create for a class will affect every tag that belongs to that class.

To style anything with an id, you'd start with the #.

To style any class, you'd start with a . .

ID to pre-existing elements

If you want to just make one particular element have its own unique style, you can do that by giving a unique id to a preexisting element.

In your html file, find a paragraph or header (or something else) you want to give a unique style to.

```
<p >
  D. Yarrington, 1313 Mockingbird Lane, Newark, DE 19716
</p>
```

Give this tag its own unique id as follows:

```
<p id = "address">
  D. Yarrington, 1313 Mockingbird Lane, Newark, DE 19716
</p>
```

Now in your css stylesheet, create a style for the address element: Add:

```
#address {
  background-color: #003322;
  color: #FF2200;
  text-align: center;
  margin: 0px;
  padding: 3px;
}
```

Only the paragraph with the id of address should have this new style.

Positioning:

You can position elements on your web page using both html techniques and css techniques. One of the easiest ways to position elements is to use an html **table** (which you saw and worked with in the html tutorial).

You can also use CSS to position elements. You can use float, or you can position absolutely or relatively. Let's start with float:

Float

Often you want images, or certain text, or possibly a sidebar to move to one side of your web page, with all the text and images under it to float up next to it. Again, think of an image on your web page. You don't want an image sitting there by itself, with the text below it. You want the text to flow around the image. To do that you'd create a style for the img tag and then include in it float: right (or left). Then all the text below the image will flow up and around the image.

NOTE: When you float an element, the text and images *below* the element will flow up into the blank spaces surrounding the element. Only those elements below your floated element flow up. Anything above will not flow down. ***This means on occasion you may need to rearrange your html code!***

First, if there isn't an image in your web page, add an image. Make sure there are paragraphs or some other content below the image. If you save the file and look at it now, it should have the image on its own line, with the other content starting below the image. Often you'd want the text below the image to flow up beside the image, so it looks like it's in the flow of the document. To do that, in your css stylesheet create a style for the img tag.

Add the float to the img style:

```
img { padding-left: 10px;
      padding-top: 0px;
      padding-bottom: 10px;
      padding-right: 10px;
      text-align: center;
      margin-left: 25px;
      float: right;
}
```

Save your css stylesheet. Reload your html page in the browser. See how everything below the img in your html page now flows up to the left of the img?

Note: if you want one image to float up to the right, and then another image to float to the left, give each img an id (or give a set of images a class). So in your html, you'd have something like:

```
<img src = "giraffe.jpg" width = "200" height = "300" alt = "a giraffe" id = "giraffeimage >
```

Then in your css you'd style it as follows:

```
#giraffeimage {
  float: left;
  margin-right: 20px;
  margin-bottom: 10px;
  margin-top: 5px;
  margin-left: 0px;
}
```

Clear

Clear forces a the element being styled to go all the way to the edge, either on the right or the left, of the element it is in. We use this tag for something that we absolutely want to flow BELOW something that has been floated.

For example, let's say you have an image of a giraffe and some text on giraffes. Then the next section is on lions. You may want the text about giraffes to flow up and around the giraffe image, but then the h2 tag on lions to be below the image. You could put a tag (either an empty tag, or a paragraph, or something like that) right below the paragraph of text on giraffes. And then add a clear style to that tag:

```
#bottomofgiraffesection {
  clear: right;
}
```

Now on your web page, you know that no matter how wide you make the browser, the lion section will start below the giraffe image.

SIDEBARS:

NOTE: on web pages with sidebars, you often put a div tag around everything you'd want in the sidebar and give it an id (e.g., `<div id = "sidebar">... </div>`). Then you set the width of the sidebar style and float it to either the right or the left. Everything below it will float up to the side of the sidebar, giving you the look you want).

```
#sidebar{
    background-color: #FFFF66;
    width: 295px;
    float: right;
}
```

Now, right above the bottom of the sidebar div section, add another tag (maybe call it `bottomofsidebar`). SO you'd have something like:

```
<div id = "sidebar">
    <h2> Sidebar section </h2>
    <h3> list of links </h3>
    <ol>
        <li> <a href = "pandas.html"> Link to pandas </a></li>
        <li><a href = "koalas.html"> Link to koalas </a></li>
    </ol>
    <div id = "bottomofsidebar"></div>
</div>
```

And then in css:

```
#bottomofsidebar {
    clear: left;
}
```

Using Float for a Navigation Bar

You can use a table to create a horizontal navigation bar (or to align images horizontally next to each other), but today it is more common to use float to create a horizontal bar.

As an example, let's assume you have 4 links in your navigation bar, and you want them to run horizontally across your web page, with

```
<a href="home.html" >Home</a>
<a href="AboutUs.html">About Us</a>
<a href="Schedule.html">Schedule</a>
<a href="Contact.html">Contacts</a>
<div class = "clear"> </div>
```

And then the css styling would be:

```
a {
    box-sizing: border-box;
    background-color: black;
    color: white;
    width: 23%;
    height: 40px;
    text-align: center;
    padding: 10px;
    margin-right: 5px;
    float: left;
}

a:hover {
```

```
background-color: #FF0000;
}
div.clear {clear: both;}
```

This creates a navigation bar for 4 items (thus the width of each is at 23%, with 5px margin between each element). The hover will turn the background color red when the user runs their mouse over a particular link. The clear div makes sure that the item below the navigation bar does not appear to the right of the navigation bar.

Adding Absolute Positioning:

Absolute positioning is different from float in that when you position something absolutely, the thing you've positioned in essence is placed on top of the other elements on your web page, and those other elements would flow under the thing you've positioned absolutely. For elements positioned absolutely, we specify the pixels down from the top and over from the left. The element is then placed there in relation to the edge of your web page. Everything else on the page will flow under the element (not around it). You might use absolutely positioned elements to add something to grab the user's attention. To get a feel for what I mean, let's add an element and then position it absolutely:

First add another <div> element to the html code. The <div> tag will surround the html code we want to position absolutely. So open your html file for editing. In the html file, scroll to the bottom. Right above the </body> tag add a new div element (we're adding the link as well):

```
<div id = "absolutetag">
    <h4> Check out the <a href = "http://www.sandiegozoo.org/"> zoo! </h4>
</div>
```

Save the html file and bring it up in your browser and look at where the absolutetag element (the zoo link) is located. Change the size of the browser. Does the link move at all?

Now add some style to the absolutetag element: Open your css stylesheet. Add (nothing we haven't seen before so far):

```
#absolutetag {
border-style: solid;
background-color: #778877;
border-width: 8px;
border-color: #996633;
color: #5E2605;
padding: 0px;
padding-left: 15px;
padding-right: 15px;
position: absolute;
top: 0px;
left: 70px;
}
```

Save your css file and reload the html file in the browser. Note that the element is now positioned at the top.

Resize your browser a few times. Notice how the element stays exactly 0 pixels down and 70 pixels from the left, while other elements slide around. Notice also how other elements slide UNDER the absolutely positioned element.

Relative Positioning:

Relative positioning is like absolute positioning, only instead of the position being positioned from the top and left of the web page's edge, it's positioned from where the element would be normally in the web page. Think of it this way. You create a web page. The element is at a certain place in the web page. If you want it to be slightly over or down from that position, you can use relative positioning.

For example, say you had the following html:

```
<img src = "bird.jpg" width ="25" height = "30" alt = "a bird" id = "bird1">
<img src = "dog.jpg" width = "30" height = "20" alt = "dog" id = "dog1">
```

As it stands now, the dog image would be to the right of and lined up with the bird image. To reposition the dog to be over and down from the bird, you'd include the following style:

```
#dog1 { position: relative;
        top: 30px;
        left: 10px;
    }
```

Now the dog will be 30 px down and 10 px to the right of the bird image. It should remain in the document, so other elements should not flow under it.

Fixed Positioning

Fixed positioning is kind of like absolute positioning, only it is positioned based on the edge of your browser. So if you scroll up and down, the element with the fixed position won't move, while the rest of your web page will scroll up and down. You might use this tag for personal information you always want to be available on the screen.

The css style would look something like this:

```
#contactinfo {
    position: fixed;
    top: 0px;
    left: 20px;
}
```

z-index

Finally, when positioning things, sometimes they may overlap. If you want to control which is on the top and which is on the bottom, you can set the z-index. Think of the z-index as coming straight out at you from the web page. The z-index is a number, and the greater the number the higher up the position. So if you had the following html:

```
<p id = "first"> First </p>
<p id = "second">Second </p>
<p id = "third">Third </p>
```

Then the CSS could be:

```
#first { width: 140px;
        height: 50px;
        position: absolute;
        background-color: #FF1143;
        top: 100px;
        left: 80px;
        z-index: 1;
        text-align: left;
        padding: 10px;
        padding-top: 20px;
        color: white;
    }
#second{ width: 250px;
        height: 150px;
        position: absolute;
        background-color: #21CC00;
        top: 10px;
```

```
    left: 10px;
    z-index: -1;
    text-align: center;
    padding: 20px;
    color: white;
}
#third { width: 120px;
         height: 215px;
         position: absolute;
         background-color: #4477FF;
         top: 65px;
         left: 180px;
         z-index: 2;
         text-align: center;
         padding: 20px;
         color: white;
}
```

And you'd get something that looks like this:

