Week Four: CSS Boxes & Positioning

The Div & the ID

Div tags: where `<div>` is short for division, are the basic tags you will use to group your content into logical components

```
<div id="logo">
    <a href="http://www.premiumdw.com/">
     <img src="images/flameLogo.gif" alt="Premium Design Works" />
    </a>
</div>
```

ID's: are the means to control each `<div>` tag of logical components via the style, selected by the **#**

```
#logo {
    width: 300px;
    height: 60px;
    margin-top: 0px;
    }
```

- only one instance of an ID can be used per page
- used to identify a unique piece of your page's markup; logo, navigation, etc.
- used to identify and enable a specific target for a JavaScript function

The Box Model – *p. 103, fig. 4.2*

- every element you create in your markup produces a "box" on the page
- by default the border of the box is not visible and the background is transparent
- by giving borders and colors to your boxes, you will see the "box model" in effect

Borders: you can set the *thickness*, *style* and *color* of your border – ***p. 104, fig. 4.3***

```
#borders {
    border-width: 2px;
    border-style: solid;
    border-color: #85898A;
    }
```

1) **Border-Width:** you can set the thickness of your border via: *thin*, *medium*, *thick* or any unit
2) **Border-Style:** you can set the style of the border via: *none*, *hidden*, *dotted*, *dashed*, *solid*, *double groove*, *ridge*, *inset* and *outset*
3) **Border-Color:** you can set the color of each border via: *RGB*, *hexadecimal* or *keyword*

Border-Shorthand: you can write the border declaration with shorthand; *width*, *style*, *color*

```
#copy img {
    border:1px solid #F20017;
    }
```

Padding: will allow you to set the space between the box's content and the border of the box – ***p. 106, fig. 4.4***

```
#padding {
    padding-top: 10px;
    padding-right: auto;
    padding-bottom: 10px;
    padding-left: auto;
    }
```

1) **Padding -Top:** allows you to set the distance from the ***top*** of your “box” to adjacent elements in units
2) **Padding -Right:** allows you to set the distance from the ***right*** of your “box” to adjacent elements in units
3) **Padding -Bottom:** allows you to set the distance from the ***bottom*** of your “box” to adjacent elements in units
4) **Padding -Left:** allows you to set the distance from the ***left*** of your “box” to adjacent elements in units

Margins: will allow you to set the distance between this “box” and the adjacent elements – ***p. 107, fig. 4.6***

```
#margins {
    margin-top: 10px;
    margin-right: auto;
    margin-bottom: 10px;
    margin-left: auto;
    }
```

1) **Margin-Top:** allows you to set the distance from the ***top*** of your “box” to adjacent elements in numerical and percentage units
2) **Margin-Right:** allows you to set the distance from the ***right*** of your “box” to adjacent elements in numerical and percentage units or auto units
3) **Margin-Bottom:** allows you to set the distance from the ***bottom*** of your “box” to adjacent elements in numerical and percentage units
4) **Margin-Left:** allows you to set the distance from the ***left*** of your “box” to adjacent elements in numerical and percentage units or auto units

Margins & Padding Defaults: each browser has a default margin and padding to the page and its elements, so it is a good idea to set your own to zero

```
* {
    margin: 0;
    padding: 0;
    }
```

- by using the * selector, we are specifying “all” elements

Margins & Padding Shorthand: by using the shorthand method you can combine all margin and padding properties into one declaration

```
#shorthand {
    margin: 5px 10px 5px 10px;
    padding: 10px 5px 10px 5px;
    }
```

- when using shorthand, you will write your values clockwise; *top*, *right*, *bottom*, *left*

Collapsing Margins: vertical margins will “collapse” when there are both top and bottom margins horizontal margins do not

- when top and bottom margins meet, they overlap until one of the margins touches the border of the other element
- the larger of the two will override – ***p. 108, fig. 4.8***

Block & Inline Properties

Block: elements, such as `<p>` , `<h1>` , and `<div>` sit one above another when displayed in the browser window

Inline: elements such as `<span>` , `<img>` sit side by side when displayed in the browser window

Positioning Elements

Static Positioning: with *static* positioning, each element is simply laid out on the page one-by-one as it is written in the code – ***p. 112, fig. 4.13***

- *static* positioning is the default type of positioning
- to override *static* positioning, you will need to specify either *relative*, absolute or *fixed* positioning

Float & Clear Properties

Float: the *float* property is primarily used to flow text around images – ***p.117, fig. 4.18***

```
img {
 float:left;
 margin:0 4px 4px 0;
 }
```

- in order for the float property to work properly, you must write, in you code, the element to be floated before the element that wraps around it

```
<img src="../ch1_xhtml/stylin_logo1.gif" width="150" height="80" />
<p>Here is a paragraph of text. It wraps around the image because the image's float
property is set to left. </p>
```

Clear: the *clear* property is used to override the *float* property – ***p.119, fig. 4.20*** & ***p.121, fig. 4.21***

```
p {
   margin:0 0 10px 0;
   }

img {
   float:left;
   margin:0 4px 4px 0;
   }

.clearthefloats {
   clear:both;
   }
```

Absolute Layouts

Absolute Positioning: will allow you to put your boxes anywhere on the page by means of a *top* and *left* coordinate

```
#logo {
     width: 500px;
     position: absolute;
     top: 25px;
     left: 50px;
     }
```

Position: by giving the *position* property a value of *absolute*, you are specifying that the rule will adhere to a position that you specify on the page

Top: the *top* property will say how many pixels down from the *top* of the page that your box will sit

Left: the *left* property will say how many pixels across from the *left* of the page that your box will sit