ADV3550
Instructor: Genevieve Hitchings

The CSS Box Model
Cascading Style Sheets – continued
CSS is an extension to basic HTML that allows you to "style" how the content within your web pages will look. Each browser has a set of parameters for how it reads paragraphs, headings, tables, etc., and then renders it on your screen. Each of the tags can be defined so that you can tell the browser how to display it. With this in mind, you can think of your CSS file as a sort of dictionary with a clear definition of how each item is displayed on the screen.

With CSS you can set up rules (or definitions) to tell specific HTML tags how to display content, or you can create generic rules and apply them to tags when you need them. There are the three types of rules:

HTML Selector
All HTML tags have default properties. With CSS you can define any HTML tag to change the defaults.

```
->      p {line-height: 14px;}
```

You can define a style for multiple tags at one time by separating each html tag name with a comma.

```
->      h1, h2, h3 {color: #c00000;}
```

Class Selector
A class is a rule that can be applied to an HTML tag. You can give a class any name you want with the exception of a few reserved words.

```
->      .myclass {font: bold 14px Verdana;}
```

Classes have an advantage in that you can create a number of different classes and apply them to different paragraphs, headings, divs, etc.

ID Selector
ID rules are similar to a class, but they can only be applied once on a page to a particular HTML tag. IDs are commonly used in CSS positioned layouts since items like a header or footer are only going to appear once on a page. An ID rule looks like the following:

```
->      #myid {
         background-color: #ffffff;
         height: 40px;
         padding: 10px 20px;}
```

Writing a Rule
Each of the rule examples above (html selector, class, and ID), consist of three distinct parts.
1. the selector whether it be an html selector, a class, or an ID.
2. a space and then a beginning curly brace.
3. then the property, followed by a colon, like this:

```
->      selector {
         property: value;
         property: value; }
```

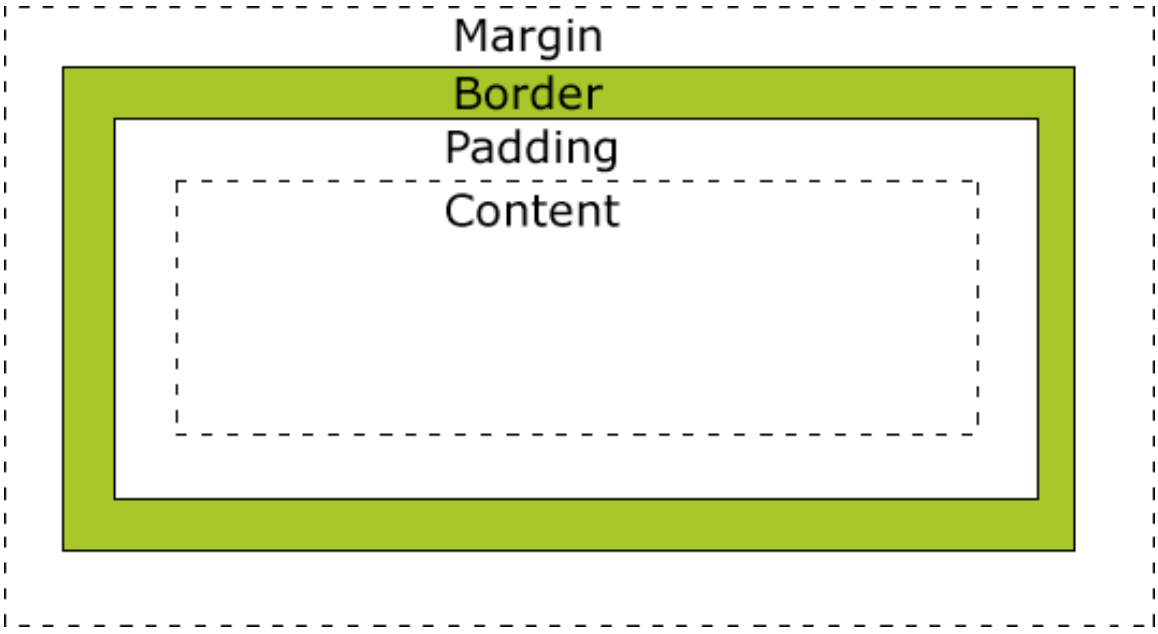
The CSS Box Model
Many HTML tags are considered block elements, meaning that they are a type of box with the same four sides. In CSS, the term "box model" is used when talking about design and layout.

Each block level tag has four sides that can be specified in order going clockwise beginning at the top. Every box element can have a width and height defined, as well as a margin, border, and padding to all four sides.

Width and height properties accept a length value, a percentage value, or auto (which leaves the dimensions up to the browser.) An example is as follows:

```
.myarea {
  width: 300px;
  height: 100px; }
```

The CSS box model is essentially a box that wraps around HTML elements, and it consists of: margins, borders, padding, and the actual content. The box model allows us to place a border around elements and space elements in relation to other elements.



Explanation of the different parts:

- **Margin** - Clears an area around the border. The margin does not have a background color, and it is completely transparent
- **Border** - A border that lies around the padding and content. The border is affected by the background color of the box
- **Padding** - Clears an area around the content. The padding is affected by the background color of the box
- **Content** - The content of the box, where text and images appear

The **total width** of an element should always be calculated like this:
Total element width = width + left padding + right padding + left border + right border + left margin + right margin

The **total height** of an element should always be calculated like this:
Total element height = height + top padding + bottom padding + top border + bottom border + top margin + bottom margin

---

Block-level elements vs. Inline elements

Block-level elements
A block-level element may contain other block-level elements, or it may contain inline elements. But inline elements cannot contain block-level elements, only data or other inline elements. So a block-level element is "bigger" or "higher" in the hierarchy. It defines a larger section of of the page than an inline element does.

Inline elements
Block-level elements generally begin on new lines. Inline elements do not start on a new line. If you type out a paragraph, and decide to make one of the words italic, this would be an example of an inline tag. So is bold, strong, em, a href, etc. None of these tags causes the browser to start a new line, so these are all inline elements.
Example:
<a>

Div tag:
The tag defines a division or a section in an HTML document. It is often used to group block-elements to format them with styles.

Span tag
The tag simply tells the browser to apply the style rules to whatever is within the

Below is a list of common **block-level** HTML tag pairs:
<blockquote> ...</blockquote>
<body> ... </body>
<button> ... </button>
 ...
<dl> ... </dl>
<fieldset> ... </fieldset>
<form> ... </form>
<h1> ... </h1>
<h2> ... </h2>
<h3> ... </h3>
<h4> ... </h4>
<h5> ... </h5>
<h6> ... </h6>
<head> ... </head>
<html> ... </html>
<iframe> ... </iframe>
<img />

<layer> ... </layer>
<legend> ... </legend>
 ...
 ...
<select> ... </select>
<table> ... </table>
 ...

So here's an example of how it all works (don't worry if you don't understand it all at this point).

Lets say we wanted to create a page 900pixels in width. In the example below, this simple, typical web page layout consists of three blocks. The first is the 'wrapper' that holds the whole page together; the second (inside the wrapper) is the sidebar; the third , also inside the wrapper, is the 'content' which displays the left hand sidebar of page.

Here is the CSS:

```
#wrapper {
  width: 900px;
  text-align: left;
  margin: 0 auto;}

#sidebar {
  font-size: 85%;
  float: left;
  width: 270px;
  padding: 10px 30px 0 0;}

#content {
  float: right;
  width: 600px;
  line-height: 1.5em;
  padding: 10px 0 0 0;}
```

The sidebar has a width of 270 pixels and the content has a width of 600 pixels. You will see that this does not add up to 900 pixels! It's only 870 pixels.

Now look at the padding values and you will see there are pixel values. Remember that values are listed clockwise starting at the top. All you are concerned about are values affecting the width.

The sidebar has a 30 pixel value added to the right side. Add this 30 pixel value to the sidebar's width and you have 300 pixels. The content has no values added for the left and right sides, so its total width is still 600 pixels.

Add the two together and you come up with 900 pixels that exactly matches the 900 pixel width of the wrapper. It's a perfect fit.

To change both the sidebar and content div width, make adjustments so that all width values still equal 900 pixels.

The float property can be applied to divs and accepts the value of right, left, or none. To have your content and sidebar areas sit side by side

Here is what the basic HTML page would looks like:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head> <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
<title>Untitled Document</title>
<link href="style-BoxModel.css" rel="stylesheet" type="text/css" /> </head>

<body>
        <div id="wrapper">
                <div id="content"> </div>
                <div id="sidebar"> </div>
        </div>
</body>
</html>
```