CSS, cont.

October 12, Unit 4

Creating a Sidebar

- We know how to create a sidebar
  - Use the *float* property

```
div#sidebar        {
   float: left;
}

<div id = “sidebar”>
   Item1<br/>
   Item2 <br/>
   Item3 <br/>
</div>
```

CSS and Boxes

- The underlying model of CSS relies on the notion of *boxes*
  - Your screen is a rectangular box
  - A paragraph fits into a box
  - An image, etc.
- Boxes have a number of properties (we've used them)
  - Margin
  - Border
  - Padding

Boxes, cont.

- *Content*: the main content of the box
- *Padding*: space between content and border
- *Border*: the border around the content (taking into account padding)
- *Margin*: the space between this box and adjacent boxes

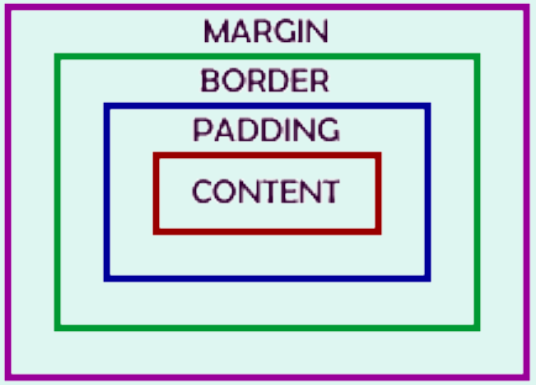

Image from: http://academ.hvcc.edu/~kantopet/old/css/index.php?page=css+box+properties&parent=css+boxes

Margins are Special

- Margins will actually change size depending on what's adjacent to the box
- If two boxes with margins are horizontally adjacent (side by side), the margin for each box stays the same
  - Two boxes both with left and right margins of 15px, the total space between the boxes would be 30px
- But, when you have one box on top of another, the margins are collapsed
  - The space between the boxes becomes the larger of the two margins
  - If the top box had a margin of 25px and the bottom box a margin of 10px, the space between the boxes would be 25px

How Floats Change Pages

- Normally when we write html, our block-level elements stack up to create the vertical space on our page
- Block-level elements are not placed side-by-side
- But we can break this using the float property
  - We remove the block from the normal flow of the page and allow content to wrap around it.

Issues with Float

- Float seems pretty simple to use
- But, let's say you have two nested tags, the inner one has the float: left applied to it

```
<div id = "wholepage">
        <div id = "sidebar">
        ……
        </div>
        …….
</div>
```

Floated Objects Outside of “Parent” Boxes

- When elements are floated, you can think of them as being pulled out of the standard html flow
- Because of this, that *sidebar* we created earlier may extend beyond the borders of parent *wholepage*
- But the purpose of having a *wholepage* identifier was to specify something for the entire page

Issues with Float

- When floating elements, it's not always a sidebar
  - We could be floating something like an image
- Remember from the 2 column example that the second column appeared *next to* the sidebar
- What would have happened if we wanted to place an element *below* the floated element?

Clear Property

- To save us from these float issues, we can use the clear property
- Specifies that you want to move below the floated content
- clear can take the value of : none, left, right, and both
- *clear: none*
  - This element can appear *next to* any floated element

Clear cont.

- *clear: left*
  - Clear all elements to the left
  - This element cannot appear to the right of any floated content
  - This element can still appear on the left side of floated content
- *clear: right*
  - Clear all elements to the right
  - This element cannot appear to the left of any floated content
  - This element can still appear on the right side of floated content

Clear: both

- *clear:both*
  - Clears elements on both the left and right side
  - This element cannot appear to the left or the right of any floated content
  - Means that this element has to appear below the floated content
- We can use this to place elements below all floated content
- We can also use this to help make parent tags contain (think borders) all of their floating content

In class Example

- 3 column layout
- Adding a footer

Using Display Property

- The display property can have the following 4 values:
  - none
  - block
  - inline
  - list-item
- What's it do?
  - Tells the browser how to display the element its applied to

display:inline

- *display:inline* is a fairly useful property
- It allows a developer to tell the browser to display a block-level tag inline
- Instead of taking up vertical space, the item now takes up horizontal space
- We can make horizontal list items

In Class Example of display:inline

display, cont.

- *display:block* displays an element as a block-level element
  - Takes up vertical space on the screen
- *display:list-item* displays an element as a list item
  - Block-level element plus a list-item marker
- *display:none* tells the browser not to display the item
- The display property can be quite useful for some applications
  - But, it causes items to be displayed in a manner they weren't intended
  - There's a good reason lists look like they do
  - Why paragraphs take up vertical space
- Do not change the display property of an element when simply using another element would achieve the same results

Pictures

- We can float columns of content
- What about making pages using pictures
- We simply use the same principles as we did for columns and content within the columns
- We can create not only single floating images, but image galleries (thumbnails arranged nicely)

In Class Example with Pictures