CSS Positioning

Putting Things Where We Want Them

CSS and HTML

- These two languages need to work together. We've seen some of this already, but to work with them we need to know a little more about how CSS "sees" HTML.

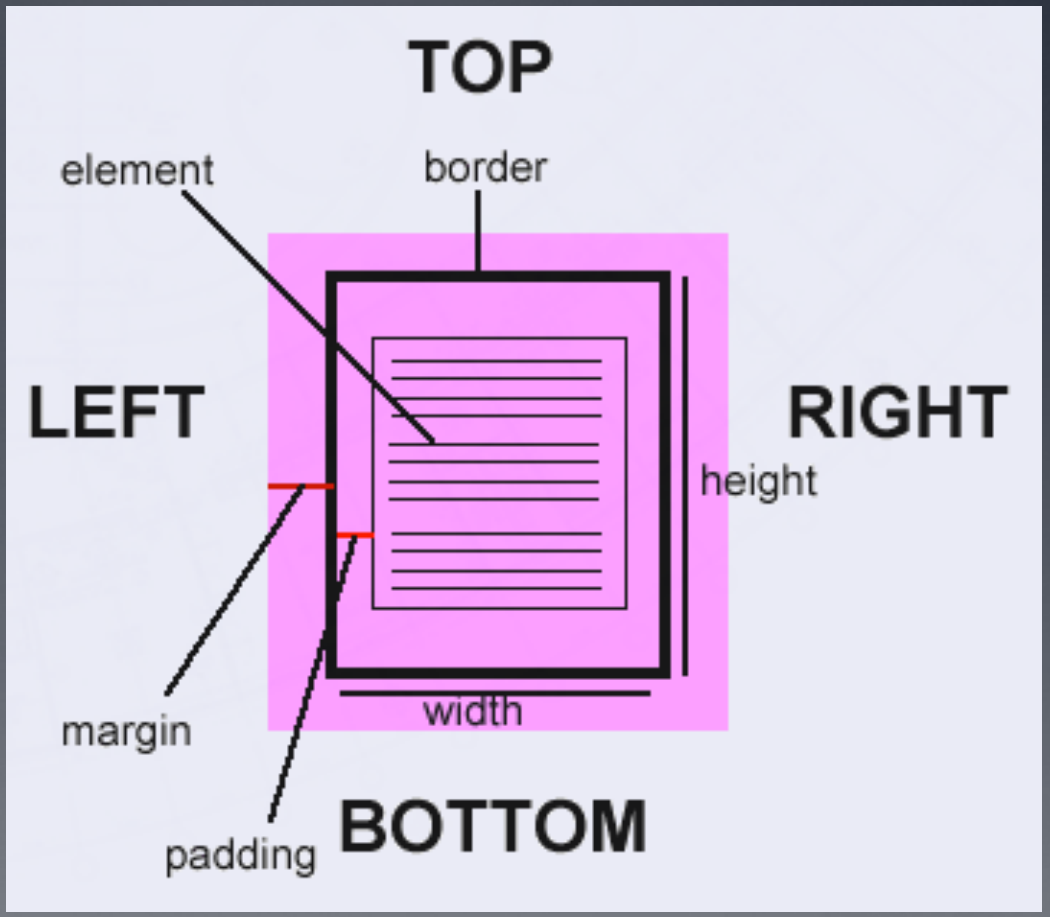
- Everything in HTML/CSS is a "box"

The Box Model

- CSS treats your web page as if every element (<p>, , …) is enclosed in an invisible box.
- CSS allows you to change a large number of the properties of this box, such as its border, color, position, flow, etc. to achieve the look and feel you are striving for.

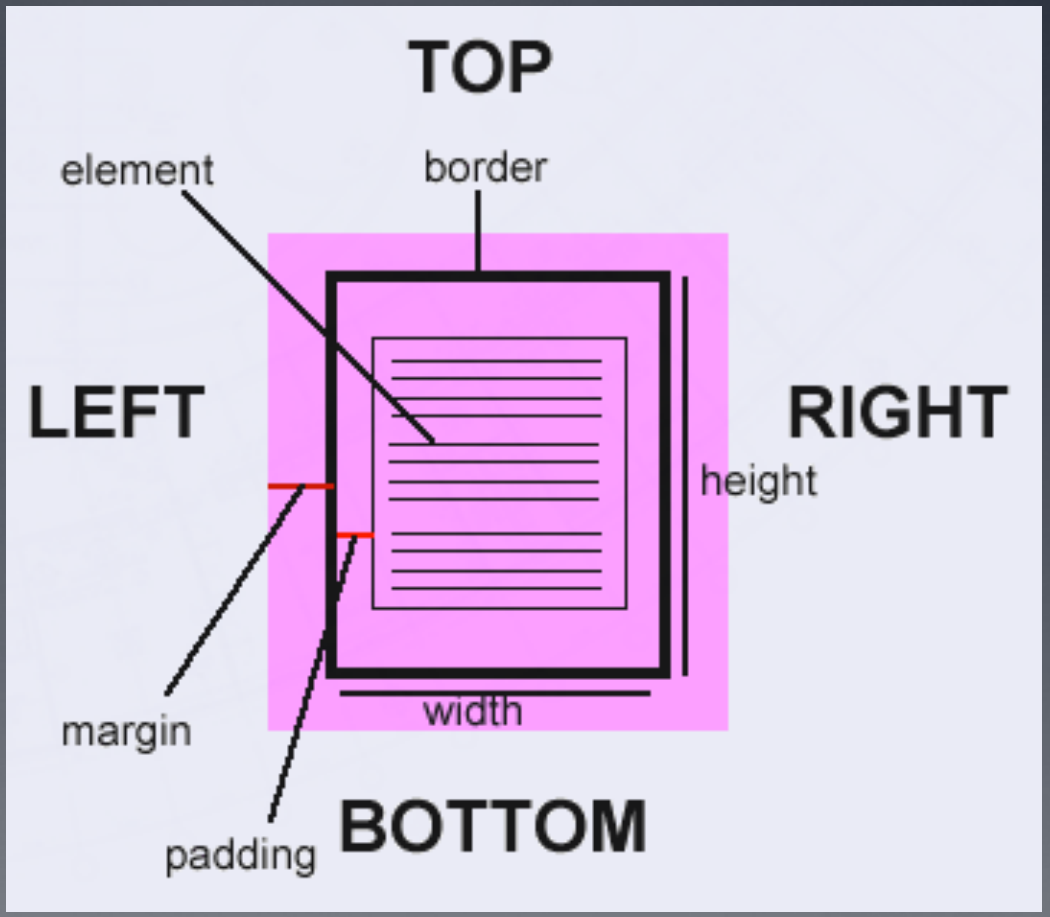
Properties of the "Box" we can set

- background-color
- background-image
- border
- Width
- height
- margin
- padding
- position
  - and related properties
    - top:
    - right:
    - bottom:
    - left:



Margin vs. Padding

- *Margin* adds space "outside of the box"
- *Padding* adds space "inside of the box"



CSS Positioning

- HTML Tables for positioning
  - Using a grid
  - Lots and lots of tags
  - Not flexible in terms of responsive design
  - Don’t use tables for this!
- CSS: Anything is possible

Five Types of Positioning

- float
- static
- relative
- absolute
- fixed

Note: Positioning only works on **block**-level elements (like), not **inline** elements like

Confusing?

- Float is like an island, not a boat
- Static items move
- Relative is relative to static
- Absolute is relative
- Fixed is absolute
- Got that?

Float

- Other items "flow" down the right side of a left-floated box and down the left side of a right-floated box.
- Used for many of the new standards for dynamic “liquid layouts,” but we don’t get precision control.
- Complete layout control involves a combination of CSS-P and floating elements; the decision of when to use what is up to the designer.

Static

- `{ position: static; }`
- "Static" means that the element will show up where it would have had you not positioned it. That is, inline with the other elements on the page.
- By default all elements are static - "static" is the way you're used to HTML behaving.

Relative

- `{ position: relative; top: 10px; left: 100px; }`
- "Relative" positioned items are positioned *relative to where they would have been* had they been positioned statically (or not positioned at all).

Absolute

- `{ position: absolute; top: 20px; left: 60px; }`
- Absolute positioned items are positioned *relative to the item enclosing them* (the parent container).
- Absolute positioned items *act as though they have no size*. Static and relative positioned items behave as though the absolute positioned items are not there.
- A good rule of thumb is to always place absolute positioned items inside a relative or static positioned item.

Fixed

- `{ position: fixed; top: 100px; left: 200px; }`
- Fixed positioning is much like absolute positioning, except that fixed items are always positioned relative to the browser window.
- An important feature of fixed items is that they *do not scroll* with the rest of the page.
- Fixed items are good for things like navigation where you don't want the navigation to scroll off the page.

Positioning Rules of Thumb

- If you're going to position some elements on the page, position them all.
- Don’t position with the intention of targeting a specific screen size; consider how it will look at various resolutions
  - More on this when we get to responsive design

z-index

- ```
  { position: absolute; top: 425px;
    left: 330px; z-index: 4; }
  ```
- CSS allows you to work in three dimensions with the z-index property. This lets you position items on top of or below other items. You can use this, for example, to create shadows. Or put text on top of images or other text.
- The higher the z-index integer, the "closer" to the user the item is. (Highest number is at the top of the stack.)


  ```

Scrolling in a box

- `{ overflow: auto; }`
- Allows a box to scroll to view its own content. Especially useful with fixed-positioned items, or making a "content" area in your content that scrolls while other content doesn't.

Another Property… Visibility

- `{ visibility: hidden; }`
- Not really positioning, but often used together with it, the visibility property lets you hide elements (and later show them using scripting).

CSS-P Demo

Let's try it!

This demo is also a nice one:

http://www.barelyfitz.com/screencast/html-training/css/positioning/