Beyond Simple CSS

Philly on Rails – August 2006

Meeting sponsored by Chariot Solutions - www.chariotsolutions.com

Send questions/comments to Erin Mulder - emulder@chariotsolutions.com

Going to cover...

- CSS Refresher
- Beyond Basic Selectors
- Understanding the Box Model
- Display and Float
- Sizing and Positioning
- Building a Layout
- Working with Text
- Maintaining your Stylesheets

CSS Refresher


```
<!-- old way -->
<p><font face="arial"><b><i>Hi!</i></b></p>

<!-- new way -->
p.pop {
  font-family:   arial;
  font-weight:  bold;
  font-style:     italic;
}
...
<p class="pop">Hi!</p>
```

Where does it all go?

- Inline
  - Hi!
  - Use very sparingly, mostly to try out something
- In <head><style>...</style></head> of page
  - Handy for initial stylesheet construction
  - Extract it before going live
- In external stylesheet
  - Best option!

Anatomy of a selector


```
p {  font-size: 1.2em; }

.pop {
  font-family:   arial;
  font-weight:  bold;
  font-style:     italic;
}

p.pop a {
  color: #ff3333;
  text-decoration: none;
}
```

Why bother?

```
<html>
<head>
  <link rel="stylesheet" href="foo.css" type="text/css" />
</head>
<body>
  <div id="header">
    <h1>Philly on Rails</h1>
    <ul>
      <li><a href="/home/">Home</a></li>
      <li><a href="/meetings/">Meetings</a></li>
      <li><a href="/contact/">Contact Us</a></li>
    </ul>
  </div id="header">
  ...
```

Beyond basic selectors


```
p { ... }
.dramatic { ... }
p.dramatic { ... }
p.dramatic a { ... }
p.dramatic a.external { ... }
.header { ... }
#header { ... }
#header h1 { ... }
#header li { ... }
p, li { ... }
p.dramatic, .forlorn a, #header li a.ecstatic { ... }
```

Pseudo-classes and elements


```
a:hover      {  color: #ff0;  text-decoration: underline; }
.tab:hover  {  background-color: #09a; }
input.focus {  border: 2px solid #00c; }
a:active      { color:#f00; }
a:link          { color:#900; }
p:first-letter { font-size: 20px, background-color: #090; }
p:first_line { font-variant: small-caps; }
<!-- not in all browsers -->
.errorfield.after {
  content: “This field contains an error.”
  color: #900;
}
```

Understanding the box model

- The key to CSS-based layouts
- Every block element has
  - Height
  - Width
  - Position (left, right, top, bottom)
  - Margins
  - Padding
  - Borders
- All of these can be individually controlled

Understanding the Box Model


```
<!-- old way -->
<table>
<tr>
  <td colspan="3"><img src="pixel.gif" height="20" width="1"></td>
</tr>
<tr>
  <td><img src="pixel.gif" height="1" width="200"></td>
  <td>Hi!</td>
  <td><img src="pixel.gif" height="1" width="200"></td></tr>
<tr>
  <td colspan="3"><img src="pixel.gif" height="20" width="1"></td>
</tr>
</table>
```

Understanding the Box Model

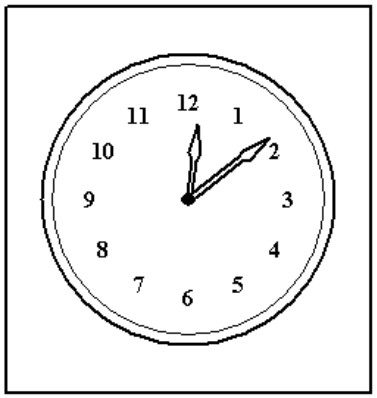

```
<!-- new way -->
<style>
  .greeting {
    padding-top: 20px;
    padding-bottom: 20px;
    padding-left: 200px;
    padding-right: 200px;
  }
</style>

<p class=”greeting”>Hi!</p>
```

Different ways to say it


```
.greeting {
  padding-top: 20px;
  padding-bottom: 20px;
  padding-left: 200px;
  padding-right: 200px;
}
.greeting {
  padding: 20px 200px 20px 200px;
}
.greeting {
  padding: 20px 200px;
}
```

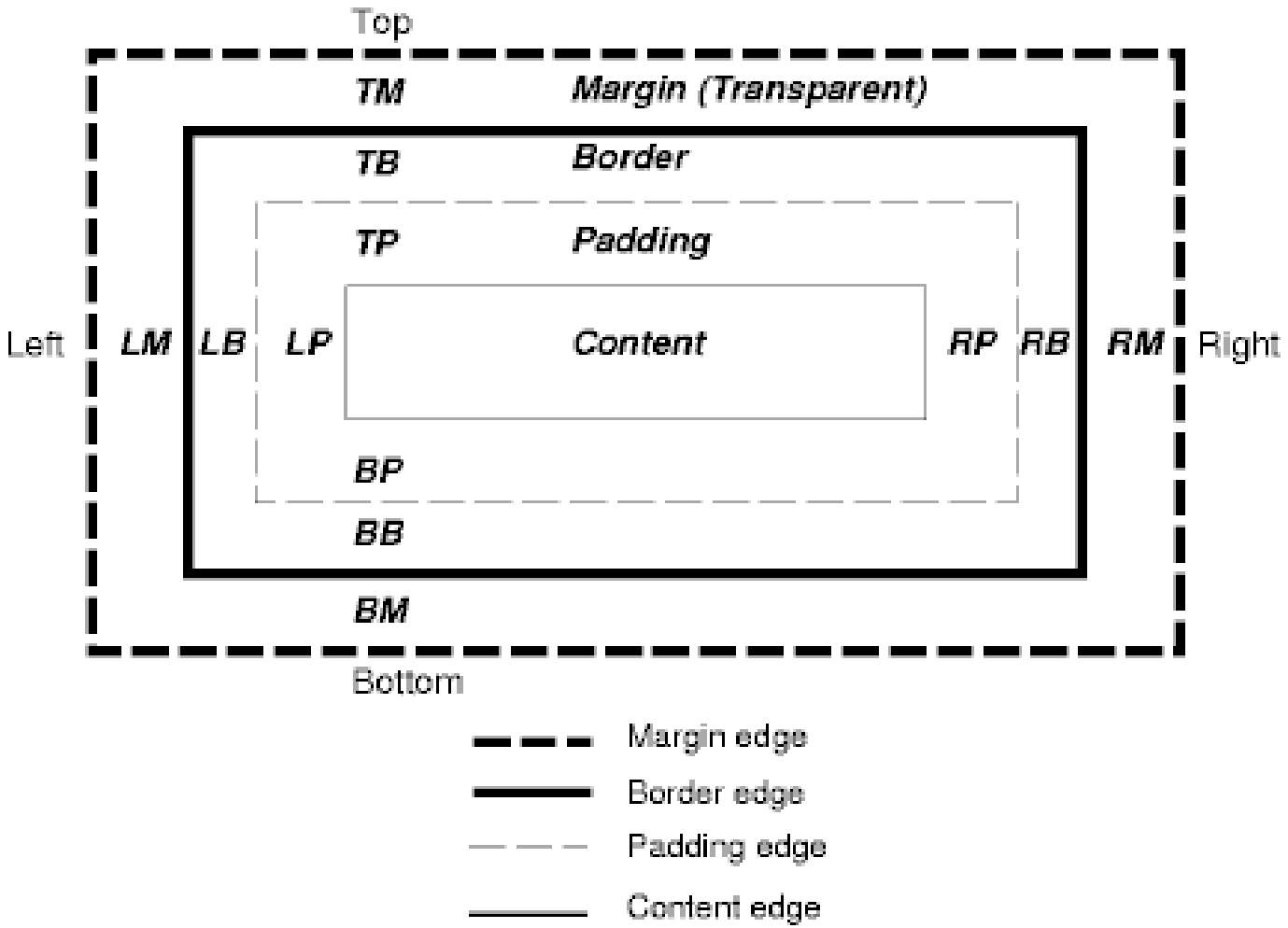
Remembering syntax

- Border, padding and margin can take 4 arguments
  - Order is TOP, RIGHT, BOTTOM, LEFT
  - Start at the top and go clockwise



- Can also take 2 arguments
  - Same as 4-argument version, except...
  - Duplicates TOP and RIGHT values for BOTTOM and LEFT

Inside the Box Model



Box Model Gotchas

- Padding is OUTSIDE the content
  - CSS width and height do NOT include padding
  - Not really intuitive
  - You'll see things overlap if you forget
- Except... in some Internet Explorer versions
  - v6.0+ can do it right in “standards” mode
  - Older versions include padding in height/width
  - Google “box model hack” for a workaround

Padding vs. Margin

- Padding
  - gets background color of content
  - never collapsed
  - useful for pixel-based designs (with box model hack)
- Margin
  - transparent (outer container shows through)
  - often collapsed
  - acts as a “minimum space” rather than a pixel-perfect padding
  - useful for aesthetics/readability

The display attribute

- Controls how an element interacts with its surroundings
  - Does it cause line breaks?
  - Does it get margins and padding?
  - Does it show up at all?
- Every HTML element has a default display value
- There are a lot of display values that aren't fully supported across browsers
  - We're not going to cover those tonight

The display attribute - inline

- display: inline
  - Some tags that default to inline:
    - Blah
    - foo
    -
  - everything flows together and wraps at end of line
  - no true box model/margins
  - inconsistent support for margins/padding/etc. across browsers
  - useful for layouts that should expand to fill a horizontal space

The display attribute - block

- display: block
  - Some tags that default to block:
    - ...
    - <table>...</table>
    - ...
  - gets line break before and after
  - gets full box model
  - use to add margin/padding/border to normally inline elements (often want float too)

The display attribute - none

- display: none
  - Handy for collapsible layouts
  - Use Javascript to set/unset it
  - document.getElementById('menu').style.display='none';
  - Element will take up no space while display is 'none'
  - Get to know defaults for standard HTML elements so you can set display back to the right value later

The visibility attribute

- Setting “visibility: hidden” hides an element
  - Still takes up space on page
  - Set “visibility: visible” to get it back
- Useful for elements that have dedicated space but aren't always visible:
  - Error icons
  - Loading icons
  - Messages
  - Hints

The float attribute

- Similar to old
- Only works if display is 'block'
- If “float:left”, other elements will flow around it to the right
- If “float:right”, they flow around on left
- Floats can stack
  - Example: put 50 same-size boxes in a row, all with float left, and they'll often line up in a resizable grid
- Escape with “clear” (e.g. “clear: both;”)

Sizing and Positioning

- Adjust size with height, width, min-height, min-width, max-height and max-width
  - Decide what should happen when content is too big and make it so with overflow
- Optionally, set a position style like 'absolute'
  - Can work in from top, right, bottom or left
  - Use z-index to manage overlaps
- Use clip to make cool shape effects
  - No, I've never used it. :)

Sizing and Overflow

- Can set max-height to constrain vertically
- Can set max-width to constrain horizontally
- If you do both, you should set an overflow style
  - **overflow: visible** – show the extra outside the box
  - **overflow: hidden** – don't show the extra
  - **overflow: scroll** – add scroll bars
  - **overflow: auto** – add scroll bars if necessary
- Be sure to test in different browsers

Positioning

- By default, things just flow (position: static)
- position: relative
  - element is still in regular flow, but gets pushed out of position based on position values (left, top, etc.)
- position: absolute
  - positions it relative to the first ancestor with non-static positioning, which is usually the page
  - lets you pin things to page borders
  - lets you rearrange content
- position: fixed

Building a layout

- Can build clean, attractive header without images
  - Style h1 for the logo area
  - Create nav tabs with block display, background colors, padding and margins applied to
- Use positioning or float for columns
- Use overflow for scrollable areas within the page
- Use positioning to hug edges
- Keep styles/positioning out of HTML

Working with text

- Lots of practical, cross-platform font choices (google for a list)
- Sans-serif fonts tend to be most readable, but that's not set in stone
- Bump up line-height for a more readable *and* attractive look
- Use ems instead of pixels for a more resilient and usable layout
- It's okay if it looks different across browsers

Maintaining stylesheets

- Minimize inline styles
- Minimize <head><style> styles
- Break things up into multiple stylesheets and link to all of them
- One stylesheet can import another with @import “style.css”
- Don't get too fancy or the precedence rules will drive you insane

Things for you to investigate!

- Rounded corners with no images, just CSS
- Box model hack
- Hiding styles from old browsers
- Cool selectors and pseudo-elements that aren't supported in IE (sibling, attribute...)
- Getting floated columns to expand vertically
- Media types (print, etc.)
- Changing the cursor
- Customizing list displays

Don't go on without...

- W3Schools.com CSS2 Reference
  - top hit if you google “css2 reference”
- “Web Developer Extension” Firefox toolbar
  - validate both local and live pages
  - resize your browser, outline divs, etc.
- Reading at least one CSS book
  - Check out Stylin' with CSS
- Visiting cool standards-based design sites
  - AListApart, StopDesign, etc.