HTML/CSS Class 3

CSS for Page Layout

Stephanie Finken
@sfinken

These slides are available at:

http://stephaniefinken.com/classes

Agenda

- Brief Review of terms
  - HTML: Tags, Elements, Attributes
  - CSS: Selectors, Properties, Values
- HTML Tables
  - Use of HTML tables for page layouts
  - Appropriate use of HTML tables (data, information)
- CSS for Page Layout
  - CSS properties: position, float, clear

Brief Review of Terms

Tags
A tag is either a start tag or an end tag. These are used to denote the start of an element (i.e.) or the end of an element (i.e.)

Element
An element is the start tag + its content + the end tag:

```
<p>This is a paragraph</p>
```

Attribute
Attributes provide additional information about HTML elements.

```
<a href="http://www.google.com">Google</a>
```

href is the attribute. *class* and *id* are also attributes

```
<div class="header"></div>
```

Brief Review of Terms: CSS

Selector
A selector is what you call the CSS that gets matched up with an HTML element or attribute.
It's the thing that comes before the curly braces {} in your CSS class.

3 kinds of selectors

- **element-type** selectors (a, body, html)
- **class** selectors - denoted with a "." (.header)
  - styles you define in class selector .header will be applied to any HTML element with an attribute of class="header"
- **id** selectors - denoted with a "#" (#username)
  - styles you define in id selector #username) will be applied to the **one** HTML element with an attribute of id="username"

Review: CSS Properties

Properties
CSS properties are the actual styles you give to your HTML elements.

Examples:
font-family, text-decoration, margin, color, background-color

Resources
http://htmldog.com/reference/cssproperties/
http://www.w3schools.com/css/

CSS for links: pseudo-classes

Some links change styles when you hover your mouse over them. (Add an underline, change text color)

This is accomplished by using **pseudo-classes**

Syntax and Example:

```
selector:pseudo-class{
  property:value;
}

a:hover{
  color:#000000;
}
```

Using pseudo-classes with links

```
a:link {color:red;}    /* unvisited link */
a:visited {color:green;} /* visited link */
a:hover {color:#444;} /* mouse over link */
a:active {color:yellow;} /* selected link */
```

A few things to note: **Order matters**

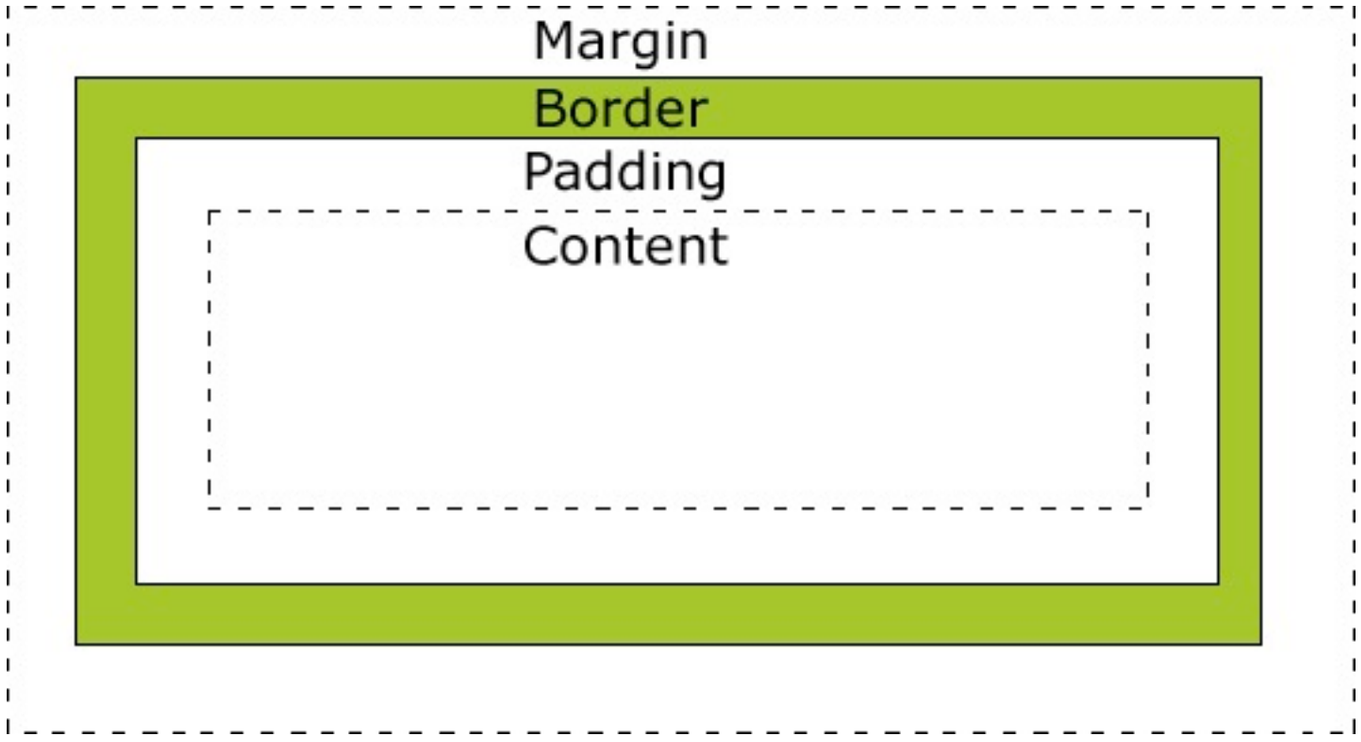
`a:hover` must come after `a:link` and `a:visited` in the CSS definition in order to be effective

`a:active` must come after `a:hover` in the CSS definition in order to be effective

CSS Box Model

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.



http://www.w3schools.com/css/css_boxmodel.asp

CSS Box Model: Borders

You can define borders in your CSS as
border: [size][border type][border color];

example: border:2px solid #000;

-will apply to all four sides

You can also define only one side:

```
border-top:1px dashed #333;
border-right: 2px dotted green;
border-bottom: 10px solid #999;
border-left: 4px solid #000;
```

CSS Box Model: Margins

You can define margins in your CSS as

```
margin-top:10px;
margin-right: 10px;
margin-bottom: 10px;
margin-left: 10px;
```

In this example they are all the same. Isn't there a faster way? Yes!

```
margin:10px;
```

Other shortcuts:
margin:[all sides];
margin:[top][right][bottom][left]
margin:[top and bottom] [left and right]
margin:[top][left and right][bottom]

CSS Box Model: Padding

Padding is done in the same way

```
padding-top:10px;
padding-right:10px;
padding-bottom:10px;
padding-left:10px;
```

```
padding:10px;
```

Same shortcuts:

padding:[all sides];
padding:[top][right][bottom][left]
padding:[top and bottom] [left and right]
padding:[top][left and right][bottom]

Page Layouts

There is more than one way to structurally layout a web page.

Let's say we wanted to build a page with 3 main columns, (left, center, right) a header and a footer. There are a few approaches:

- Using an **HTML table**
- Using **CSS properties**: position, float, margin, padding and clear

First, let's try the approach using an **HTML table**. We will still use some CSS to style the font, links, colors, etc.

The files we will be reviewing are all in the zip file you can download at: http://stephaniefinken.com/classes

HTML Tables (for layouts)

First, let's review how to build a basic table

Open up two of the files from the class3.zip:

- **funWithTables.html**
- **tables.css**

Some notes about these files:

- The header and footer are both table rows. They stretch across the entire table because each has a single td with the attribute **colspan set to 3** (td colspan="3")

- We force the middle column to be a certain height by using the css height property in the middleCell class selector:
  .middleCell{ height:400px;}

HTML Tables: Adding some more CSS styling

Next, let's review a similar page with just a little more content and a few more styles.

This is an opportunity to practice using some of the CSS properties we learned about last week.

Open up two of the files from the class3.zip

- **moreCompleteTableLayout.html**
- **tableLayout.css**

Notes:

- The a tags are styled using pseudo-classes.
  http://www.w3schools.com/css/css_pseudo_classes.asp

- The links in the left and right columns are inside an **unordered list (ul tag)**. We've used CSS to space them out (with the padding property) and to remove the bullets with **list-style-type:none;**

HTML Tables: Not the only way

We can do everything we just did with tables using CSS properties instead of these tables.

There are advantages to using only CSS properties:

- Lighter Code
- Easier to make changes later
- More Accessible
- Semantics

We will learn the advantages in more depth after creating a page with a CSS layout.

CSS Properties for Page Layouts

We are going to use the following CSS properties to make a more flexible layout:

- Position
- Float
- Properties from the CSS Box Model
  - Margin
  - Padding
  - Border

Before we put it into practice, we are going to review what each of these properties does, how they work, and have a chance to play with each.

CSS Positioning: Static & Fixed

Static Positioning

HTML elements are positioned static **by default**; there is no need to set them to static. Static positioned elements are positioned according to the normal flow of a page. They ignore anything specified by top, bottom, right or left properties.

Fixed Positioning

An element with fixed position is positioned relative to the browser window. It will not move even if the window is scrolled, it will always stay in the same, fixed location on the screen.

See this in action:

http://www.w3schools.com/css/tryit.asp?filename=trycss_position_fixed

fixedPosExample.html included in the class3.zip file

CSS Positioning: Relative

A relative positioned element is positioned relative to its normal position. You use the properties **top**, **right**, **bottom**, and **left** to position the element.

For example, `position:relative; left: -20px;` will set an element 20 pixels left of its normal position. It subtracts 20 pixels from its normal left position.

See this in action:
http://www.w3schools.com/css/tryit.asp?filename=trycss_position_relative

CSS Positioning: Absolute

Absolute Positioning
The position of an absolutely positioned element is determined by its offset values in the properties: top, right, bottom and left.

But unlike relative positioning, where the offsets are measured relative to its normal position, an absolutely positioned element is offset from its container block.

A container block is the first parent element that has a position other than static. If no such element is found, the containing block is <html>.

Absolutely positioned elements can overlap other elements.

Unlike a Fixed element, an absolute element will move as you scroll in the browser.

CSS Float

Using the float property an element can be pushed to the left or right and allow other elements to wrap around it. When an element is set to float, text and other content will flow around the floated element.

The float property specifies whether or not an element should float and which direction it should float (left, right).

```
.alignLeft{ float:left;}
```

This is used with images to align them left or right and with layouts.

Examples
http://www.w3schools.com/css/tryit.asp?filename=trycss_float
http://www.w3schools.com/css/tryit.asp?filename=trycss_float_elements

CSS Clear

Source: http://www.w3schools.com/css/css_float.asp

The clear property specifies which sides of an element other floating elements are not allowed.

Options are left, right, both, none and inherit.

Example:
http://www.w3schools.com/css/tryit.asp?filename=trycss_float_clear

Refresher: the div tag

The `<div>` tag defines a division or a section in an HTML document. It is often used to group elements to format them with CSS styles. It's a great way to add styles to a whole group of elements.

For more on div tags:
http://www.w3schools.com/tags/tag_div.asp

Lets practice adding divs to make sure we understand the concept.

Re-building our 3-Column Layout with CSS

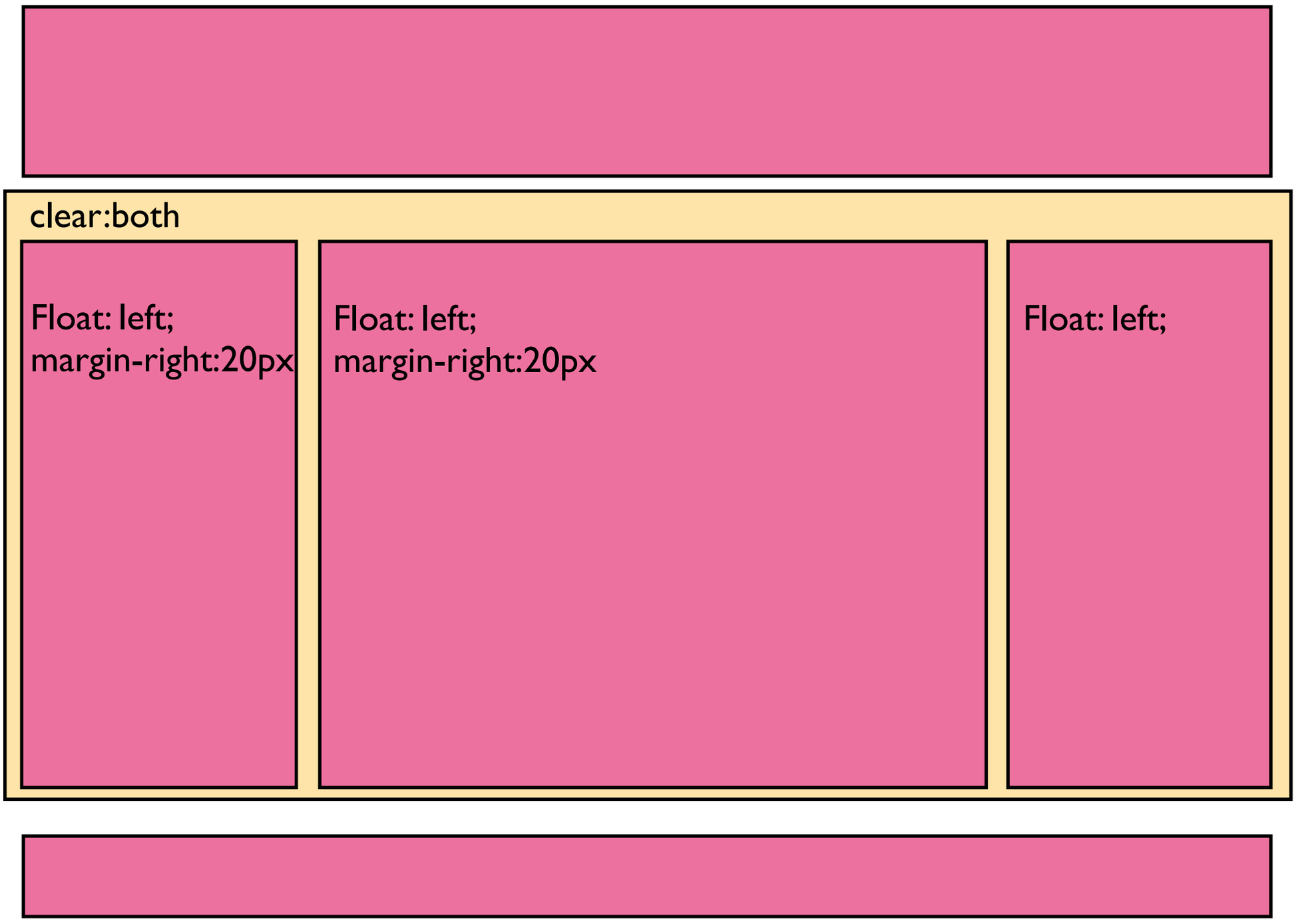
Let's take the basic 3-column layout we created earlier using an HTML table and re-create it using these CSS properties we just learned.

Open up these two files

- float-layout.html
- floatLayout.css

Let's take a look in Aptana...

CSS float example



Advantages to CSS Layouts: Lighter Code

Tables bring a lot of additional markup along for the ride. Things like `table, tr, td`.

A CSS layout means it is structured with div tags and uses `class` and `id` attributes.

Lighter code means less HTML markup, less HTML markup means the page will load faster and be easier to edit and read through later.

Advantages to CSS Layouts: Flexible Changes

Flexible Presentation
With CSS you can make changes to your layout without having to change the markup.

Easy to update styles and fonts using the classes/IDs you specify.

You can make changes to every page of your site in the CSS and don't have to go page by page and make accommodations for each table.

Advantages to CSS Layouts: More Accessible

Screen readers have trouble reading content in context if everything is in a different table cell.

Using divs and other markup tags help screen readers to determine what area the content is in.

Advantages to CSS Layouts: Semantics

Semantics means using tags as what they are designed for and what they actually mean.

Table tags mean tabular data, div tags mean container.

Labeling areas (with class and id names and using appropriate tags) with what they hold or represent helps search engines find content.

Other Resources

Web Tutorials:
http://www.webmonkey.com/tutorials/
http://www.webmonkey.com/cheat-sheets/

Clearing:
http://css-tricks.com/all-about-floats/

Positioning:
http://www.w3schools.com/css/css_positioning.asp