

**If you have not already done so,  
please download Aptana:  
<http://aptana.com>**

# GDI Cincinnati

## Intro to HTML/CSS: Class 3

Erin M. Kidwell / @erinmkidwell/ [erin@girldevelopit.com](mailto:erin@girldevelopit.com)

Heather Glenn Rock / @hglennrock / [heather@hgr.me](mailto:heather@hgr.me)

*don't be shy. develop it.*

# Agenda

## **Brief Review of Terms**

HTML: Tags, Elements, Attributes

CSS: Selectors, Properties, Values

Pseudo-classes

CSS Box Model

## **HTML Tables for Page Layout**

One way to control a page's layout is to use an HTML table.

We will walk through the HTML of a sample web page to see how a table is used to lay out the page.

## **CSS for Page Layout**

Another, arguably better way to lay out a page is with CSS.

There are a number of CSS properties that you can leverage to have more control over your web page's layout.

The properties we will learn are:

Position

Float

Clear

# Review

- Presentation from Class 2:  
<http://www.slideshare.net/emkidwell/gdi-intro-to-html-css-class-2-final>
- Code from Class 2:  
<https://gist.github.com/3147562>

# Brief review of terms: HTML

## Tag

Tags are used to denote the start of an element or the end of an element

A tag is either a start tag or an end tag. (i.e. `</p>`).

Examples of tags: `<strong>`, `<html>`, `</p>`, `</body>`

## Element

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

`<p>This is some paragraph text</p>`

## Attribute

Attributes provide additional information about HTML elements. Attributes are formatted like this: `attr="value"`

The attribute always goes in the opening tag, never in the closing tag.

In `<a href="http://www.google.com">go to google</a>`,

**href is the attribute.**

**class** and **id** are both kinds of attributes that we have used extensively in previous weeks.

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

`<div id="username"></div>`

Reference: <http://www.squidoo.com/HTML-Tutorial>

# Brief Review of Terms: HTML

## Selector

A selector is what you call the CSS that gets matched up with an HTML element or attribute.

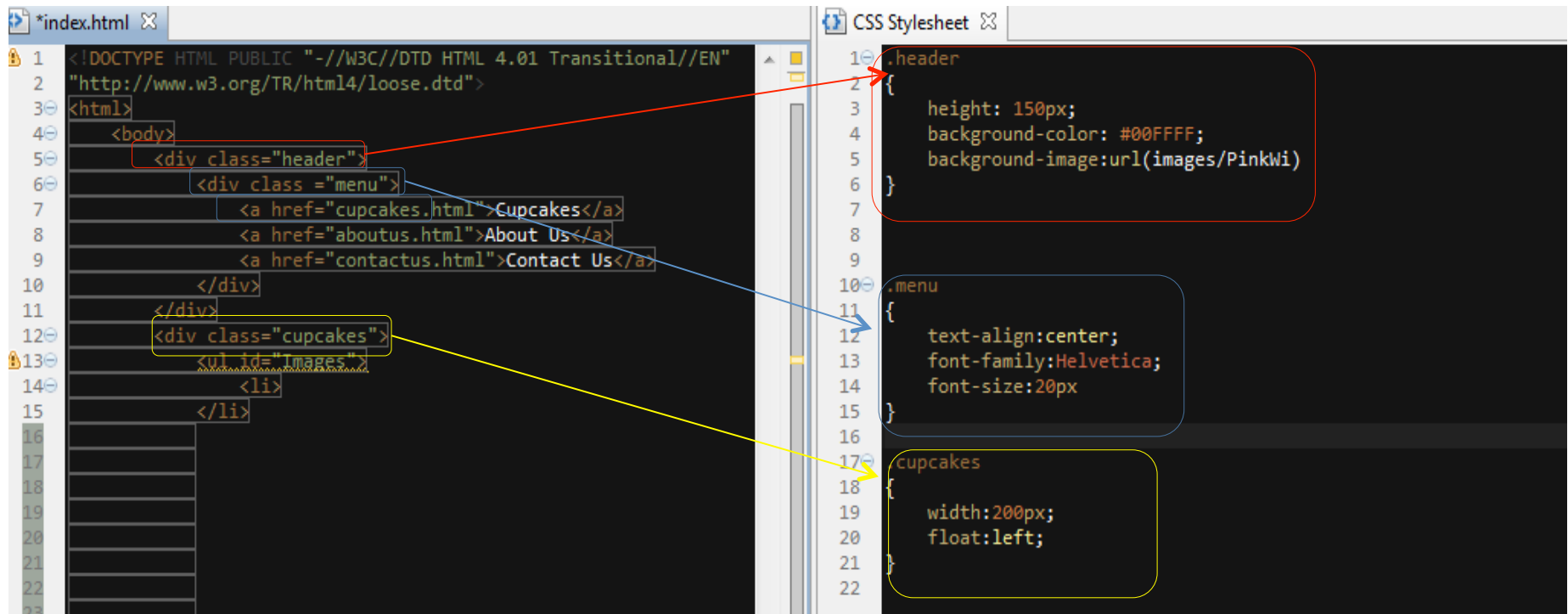
It's the thing that comes before the curly braces {} in your CSS class or inline styles in your HTML.

We have used three kinds of selectors in previous weeks:

1. **element-type** selectors (**a**, **body**, **html**)
2. **class** selectors
  - these are denoted by a "."
  - For example: **.cupcakes**, **.menu**, **.header**
  - styles you define in class selector (say, **.cupcakes**) will be applied to any HTML element with an attribute of **class="cupcake"**
3. **id** selectors
  - these are denoted by a "#"
  - For example: **#username**, **#password**
  - styles you define in id selector (say, **#pass**) will be applied to the one (id tags should always be unique on a given web page) HTML element with an attribute of **id="pass"**

# CSS Class Selectors: Example

Class Selectors being matched to HTML elements

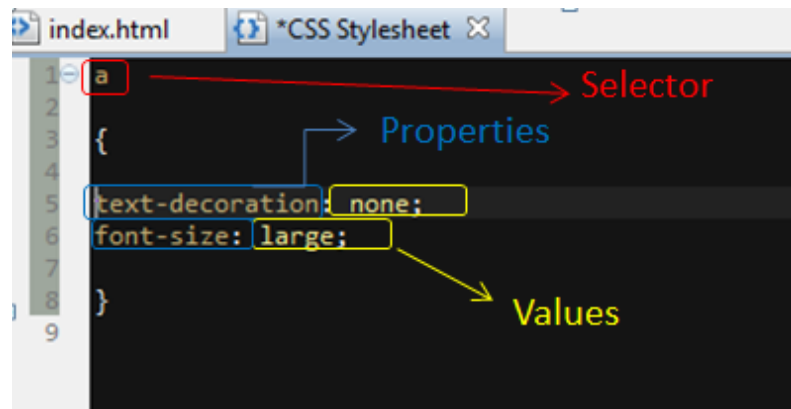


# Brief Review of Terms: CSS Properties

## Properties

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

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



For a comprehensive listing of CSS properties, see:

<http://htmldog.com/reference/cssproperties/>

<http://htmlhelp.com/reference/css/properties.html>

# Pseudo-classes: more CSS for links

Have you ever seen links on the web that change their format when you hover your mouse over them?

For example:

Links that do not have underlines, but become underlined once you hover your mouse over them?

Links whose background colors change when you hover over them?

How is this accomplished?



# Pseudo-classes: more CSS for links

Changing the format of a link when you hover over it is accomplished by using **pseudo-classes**.

CSS pseudo-classes are used to add special effects to some selectors.

Syntax:

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

Example:

```
a:link  
{  
  text-decoration: none;  
}
```

# Pseudo-classes: more CSS for links

```
a:link {color:#FF0000;} /* unvisited link */  
a:visited {color:#00FF00;} /* visited link */  
a:hover {color:#FF00FF;} /* mouse over link */  
a:active {color:#0000FF;} /* selected link */
```

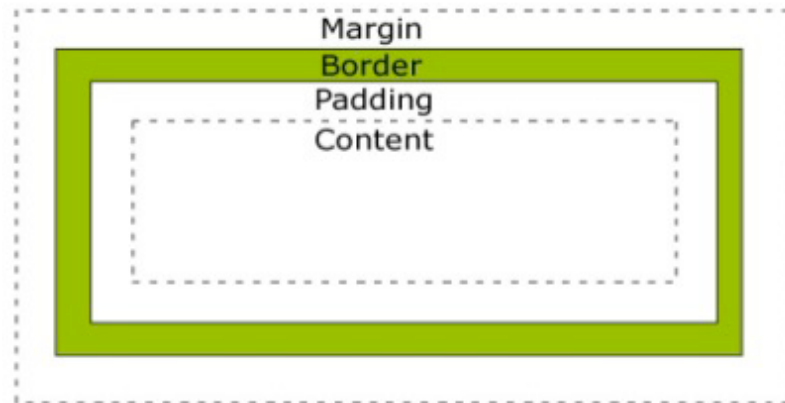
**Note:** a:hover MUST come after a:link and a:visited in the CSS definition in order to be effective!

**Note:** a:active MUST come after a:hover in the CSS definition in order to be effective!

# The 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. The image below illustrates the box model:



Read more at: [http://w3schools.com/CSS/css\\_boxmodel.asp](http://w3schools.com/CSS/css_boxmodel.asp)

# The CSS Box Model: Border

You can define your margins like this: `border: [size] [border type] [border color];`

example: `border: 10px solid black;`

You can also define only one side with a border:

- `border-top: 10px dashed blue;`
- `border-right: 10px dotted green;`
- `border-bottom: 10px solid #FFF;`
- `border-left: 10px solid orange;`

# The CSS Box Model: Margin Shortcuts

You can define your margins like this:

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

But they're all 10px... isn't there a faster way to type this out? YES!

```
margin: 10px;
```

# The CSS Box Model: Margin Shortcuts

## Long way:

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

## Shortcut ways:

```
margin: [all];  
margin: [top] [right] [bottom] [left];  
margin: [top & bottom] [left & right];
```

# The CSS Box Model: Paddling Shortcuts

## Long way:

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

## Shortcut ways:

```
padding: [all];  
padding: [top] [right] [bottom] [left];  
padding: [top & bottom] [left & right];
```

# HTML tables for 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, a header, and a footer. There are a few approaches we could take:

- Lay out the columns, header and footer using an html **table**
- Lay out the columns, header and footer using CSS properties: **position, float, margin, padding and clear**

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



# HTML tables for Page Layouts:

## *Adding more CSS Styling*

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

Whether you use tables to lay out your page, or CSS, is really a matter of personal preference.


It's become more popular/more accepted to use CSS positioning instead of tables, but if you like tables, you should use them.

Our goal is to teach you multiple ways to reach a single goal.

# Exercise: page layout w/ HTML table

See handout:  
Creating a layout  
with HTML tables

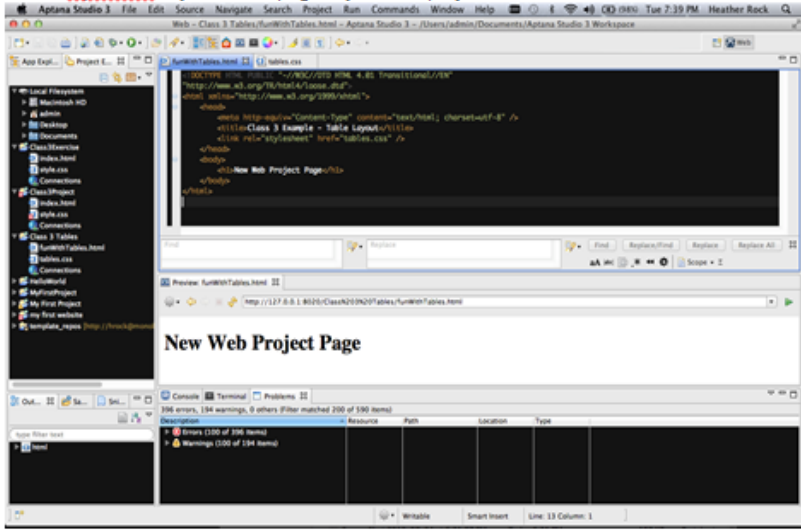
*Follow screenshot  
for live code*



## Creating a Layout with HTML Tables

Create a New Aptana Project

- Create a new Aptana Project named "Class 3 Tables". Change the name of your HTML page to funWithTables.html. Create a new stylesheet called tables.css and link the stylesheet to your HTML page in your new project.



The screenshot shows the Aptana Studio IDE. The left sidebar displays a project explorer with a new project named "Class 3 Tables". The main editor area shows the HTML code for "funWithTables.html". The code includes a DOCTYPE declaration, a meta charset declaration, a link to "tables.css", and a body element. Below the code editor is a preview window showing the rendered HTML page, which has the title "New Web Project Page". At the bottom, there is a console window showing error and warning messages.

- Inside the body, add a `<table>` element with three nested `<tr>` table row elements.

```
<table>  
<tr>  
<tr>  
<tr>
```

# CSS Properties for Page Layouts

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

Position

- Static
- Fixed
- Relative
- Absolute

Float

Clear

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

# CSS Position: 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](http://www.w3schools.com/Css/tryit.asp?filename=trycss_position_fixed)

The file **fixedPosExample.html** included in the class3.zip file

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

# CSS Position: Relative

## Relative Positioning

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

For example, **position:relative; left:-20px;** will set an element 20 pixels to the left of its normal position; it subtracts 20 pixels from its normal left position.

## Reference:

[http://www.w3schools.com/Css/tryit.asp?filename=trycss\\_position\\_relative](http://www.w3schools.com/Css/tryit.asp?filename=trycss_position_relative)

# CSS Position: 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."

"Container block"? It's 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 away from it.

# Exercise: page layout w/ CSS absolute positioning

See handout:  
Creating a layout  
with CSS:  
Absolute  
Positioning

*Follow screenshot  
for live code*

## Creating a Layout with CSS: Absolute Position

Create a New Aptana Project

- Create a new Aptana Project named "Class 3 CSS Absolute". Change the name of your HTML page to 3coolCSSAbsolute.html. Create a new stylesheet called absolutePos.css and link the stylesheet to your HTML page in your new project.  
`<link rel="stylesheet" href="absolutePos.css" />`

- Instead of using table cells for our layout, we are going to use `<div>` elements. We will go ahead and use class names as we go, so that it's obvious what we want to do with each `<div>` element.

- Add three `<div>` elements to the `<body>` of your HTML page for the header, container

# CSS Float

CSS float: an element can be pushed to the left or right, allowing 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. It also specifies which direction it should float (left, right).

Example:

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

This is most commonly used with images, in order to align them left or right so text flows around an image. It is also useful when working with layouts.

Source: [http://www.w3schools.com/Css/css\\_float.asp](http://www.w3schools.com/Css/css_float.asp)



# CSS Float

If you want to read more after class, here are some great tutorials on using floats to create a layout:

<http://css.maxdesign.com.au/floatutorial/tutorial0916.htm>

[http://www.w3schools.com/css/tryit.asp?filename=trycss\\_float6](http://www.w3schools.com/css/tryit.asp?filename=trycss_float6)

[http://articles.techrepublic.com.com/5100-10878\\_11-5160911.html](http://articles.techrepublic.com.com/5100-10878_11-5160911.html)

# CSS Clear

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

Source:

[http://www.w3schools.com/cssref/  
pr\\_class\\_clear.asp](http://www.w3schools.com/cssref/pr_class_clear.asp)

# 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 **styles**. It's a really handy way to add a certain style to a whole group of elements.

Reference: [http://w3schools.com/tags/tag\\_div.asp](http://w3schools.com/tags/tag_div.asp)

# Further Reading

## **General Web Development Tutorials:**

<http://www.webmonkey.com/tutorials/>

<http://www.webmonkey.com/cheat-sheets/>

[http://www.webmonkey.com/2010/02/color\\_charts/](http://www.webmonkey.com/2010/02/color_charts/)

<http://htmldog.com/guides/>

## **Positioning with CSS:**

The Official CSS Guide: [http://www.w3schools.com/Css/css\\_positioning.asp](http://www.w3schools.com/Css/css_positioning.asp)

CSS Positioning in 10 Steps:

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

<http://www.brainjar.com/css/positioning/>