**DIGM-243: Web Authoring II**

**Week 04**

**Design Concepts Critique**

**GOLDEN ORDER of Wordpress site modifications:**

1. **Dashboard (easy, point and click)**
2. **CSS (medium, style sheet edit)**
3. **PHP (changelling, php code modifications)**

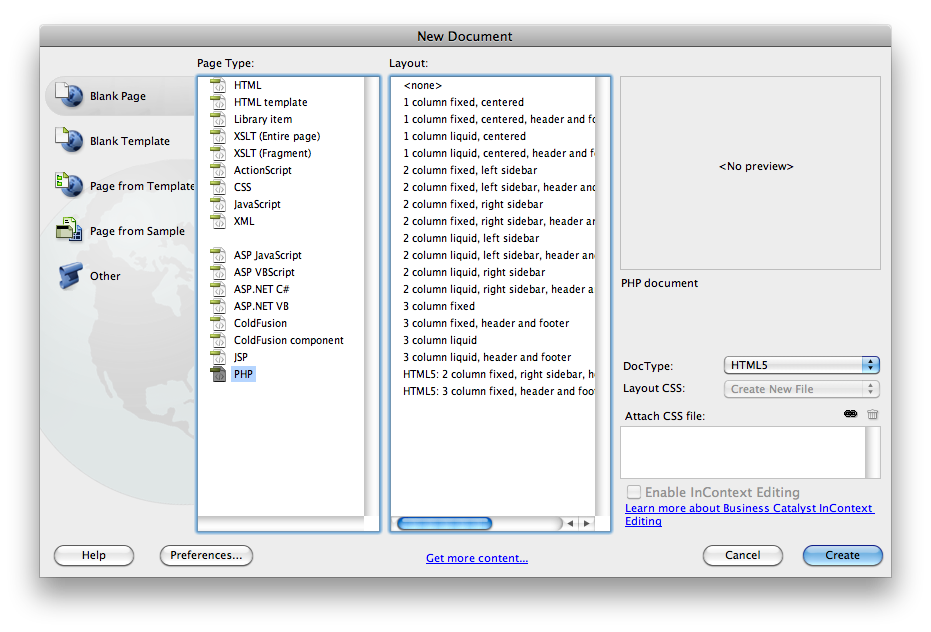
**Intro to PHP**

This is a very basic overview of PHP. A more detailed overview can be found at<http://www.w3schools.com/php/>

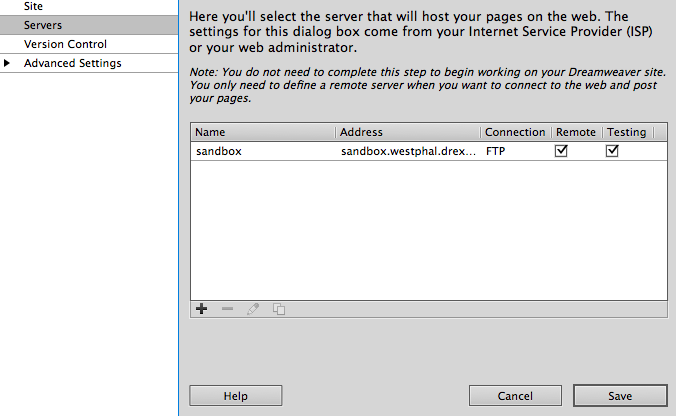
If you are interested in studying more about PHP you should look into INFO152: Web Systems and Services II

**Server Set Up**

Using your Code Editor and FTP client (i.e. DreamWeaver) create a new folder in your local folder and name it "phpIntro" inside this folder create a new PHP file (File > New) and save as basicCoding.php into the phpIntro folder.



**Note: you may need to go to Site > Manage Site and check the Testing box in the server section.**



**PHP (PHP: Hypertext Processor)**

PHP is a server-side scripting language. This means that everything PHP does happens on the web server rather than in your browser.

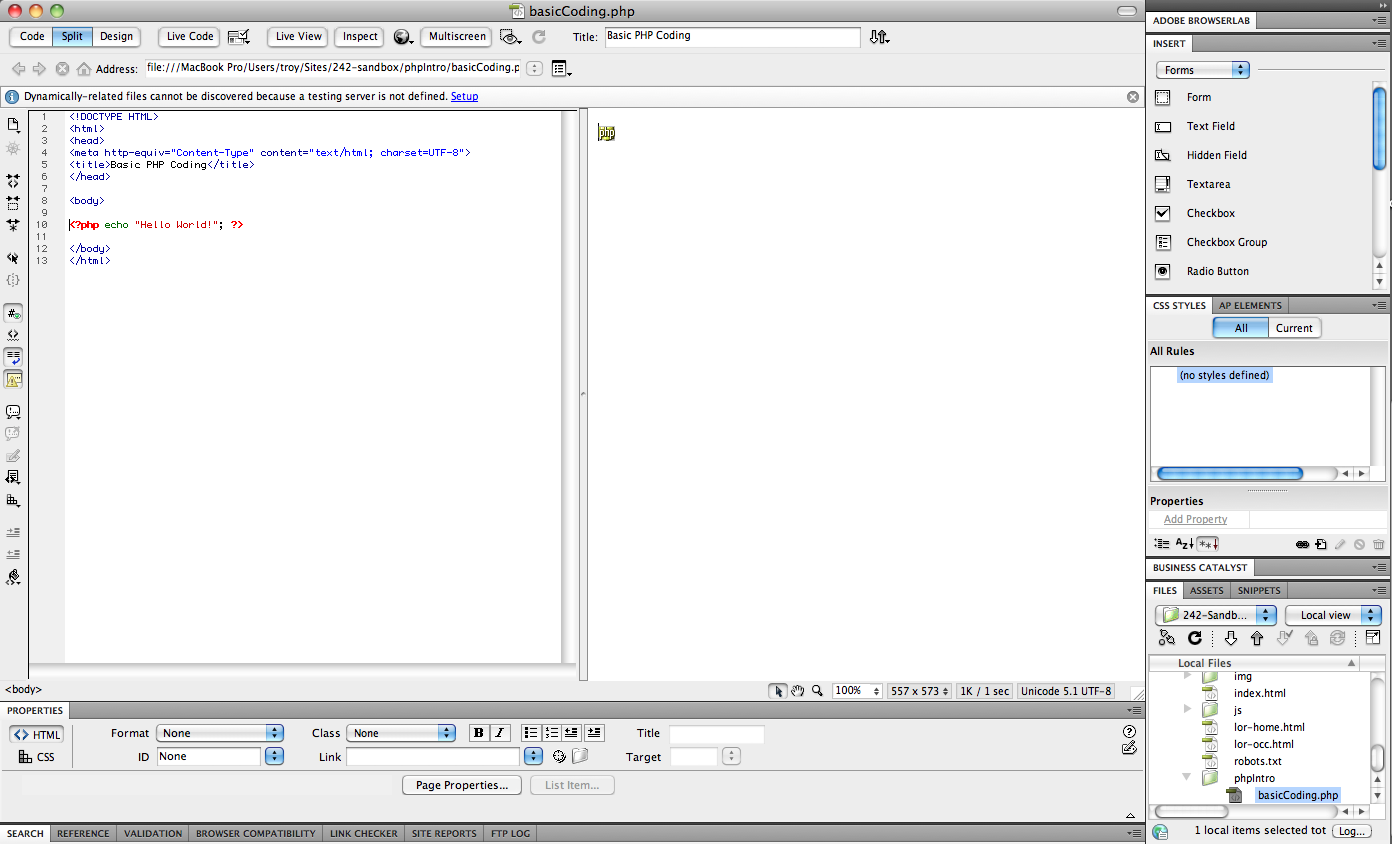
Within a .php file, you can include HTML, CSS, and Javascript like you would normally in a .html file. Valid .php files require your standard HTML tags such as <html>, <head> and <body>.

What separates a .php file from the a .html file is the ability to include php code anywhere within it.

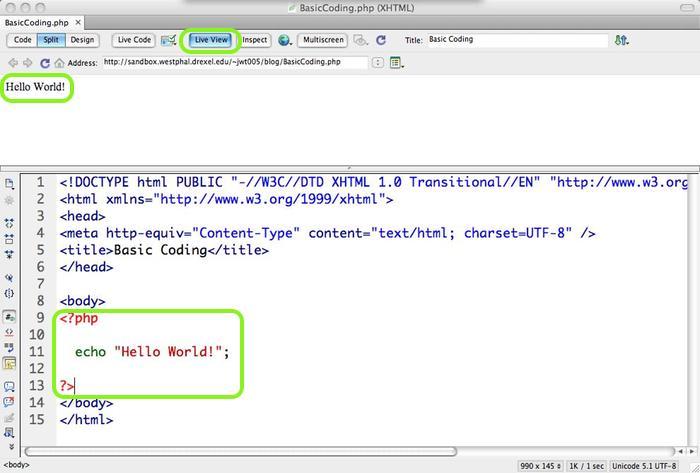
This is the basic syntax:

**<?php** echo "Hello World!"; **?>**

STEP: Put the "Hello World" php code within the <body> tag.



STEP: Click on "Live View" to see the results in Split view.



**<?php** is the opening tag to tell the server you want it to process the code following.

**?>** closes off your php code, telling the server to stop interpreting the code.

This examples also shows the most common way for php to output information back into the .php file that can then be used within html elements, the **echo** statement. The echo statement outputs anything that follows it, such as the value of variables or a string, such as in this example.

If your file is named with the .php extension, the web server will strip out all your php code run it through the processor, and return any requested data. You should never be able to see any php code in a served page via 'view source code' in your browser.

STEP: Wrap your php code with HTML tags:

<h1><?php echo "Hello World"; ?></h1>

Once the file finishes loading, viewing the source code would show you this :

<h1>Hello World</h1>

Every Line of PHP code within the opening and closing tags must end with a "**;**" just as in many programming languages.

**Comments**

Comments within the php tags are handled the same as in CSS

<h1><?php

/\* This is a comment \*/

echo "Hello World";

?></h1>

**Operators**

The operators in PHP are essentially the same as most programming languages. you can check out most of them here:

<http://www.w3schools.com/php/php_operators.asp>

One of note is the === and !== operators which represent "identical to" and "not identical too", respectively. These are different from the "equal to" and "not equal to" operators in that the values have to be of the same type (string, number, etc...)

**Variables**

PHP is very similar to other languages in that it has a standard set of code to help you perform all sorts of actions. These include **variables, functions, arrays, if-else statments, loops (for,while,etc.) and comments.**

variables are used to store information such as numbers, strings of text or arrays. Variables can store value information and be used over and over in your script.

The Syntax is

$var\_name = value;

$quantity = 4; -- declaring a number

$firstName = "Peter"; -- declaring a string value

its important to remember that every variable in PHP **must** start with the $ sign or the variable will not work. There are also a number of rules you must adhere to in order for the variable to work correctly:

* A variable name must start with a letter or "\_"
* A varaible name can only contain alpha-number characters and underscores  
   (a-z, A-Z, 0-9 and \_ )
* A variable name cannot contain spaces. If more than one word it should be separated by an underscore ($var\_name) or by using capitalization ($varName)

String variables are used to store and manipulate text. Strings are encoded with quotation marks, either double or single quotations.

$myString = "Hello World";

$myString = 'Hellow World';

You cannot do this: $myString = "Hello World';

Making sure you use the same type of quotation mark for the beginning and end of your string can save many headaches when scouring php scripts for errors. There is, however, a difference in using single quotes over double. PHP will interpret variables and function within a string that is enclosed in single quotes.

Heres an example:

<?php

$name = "Pete";

$greeting = 'Welcome to my site, $name';

echo $greeting;

?>

This code will output will look like this: **Welcome to my site, Pete**

Strings can be creating through just text or can be a combination of strings and variables combined. To do this, you use the concatenation operator. Commonly, programming languages use "+" to combine information into one string. However, PHP uses a period (.). so combining strings could look something like this

<?php

$name = "Roger";

$greeting = "Welcome to my site,";

echo $greeting . " " . $name;

?>

This would output : **Welcome to my site, Roger**

**If Statement**

PHP's if-else statements are much like other programming languages you are probably used to such as JavaScript or ActionScript. They allow you to check certain conditions within the code and allow PHP what to do based on the information.

a simple if-else statement would look like this

<?php

if ($totalPrice < 10) {

$tax = 0;

} else {

$tax = $totalPrice \* .06;

}

?>

One notable difference is how if-elseif-else statements are handled, in that "elseif" is one word as opposed to two.

<?php

if (*condition*) {

} elseif (*condition2*) {

} else {

}

?>

**Arrays**

Arrays are essentially a variable, which is able to have many of the same operations performed on it as any other variables. However, arrays are able to store a set of information that can be either used together or accessed individually through index numbers.

Typically, when using variables you need to create a unique name for every piece of information like this:

$car1 = "BMW";

$car2 = "Acura";

$car3 = "Mazda";

With arrays, however, you can do this as simply as this:

$cars = array("BMW","Acura","Mazda");

or can be indexed manually similar to assigning seperate variables.

$cars[0] = "BMW";

$cars[1] = "Acura";

$cars[2] = "Mazda";

This information can then be accessed using the index number. the first item of an array has an index of 0, and then continues in numerical order.

echo $cars[1];

outputs: **Acura**

Another type of array is the **associative array** which uses words rather than numbers to index the information, allowing you to store value information as the name of another value. These can be written the same as numeric arrays.

$ages['Peter'] = "32";

$ages['Quagmire'] = "30";

$ages['Joe'] = "34";

**or**

$ages = array('Peter'=>32,'Quagmire'=>30,'Joe'=>34);

and these values can be used like this:

echo $ages['Peter'];

outputs: **32**

As you might have guessed, arrays can also store other arrays. These are called **multidimensional arrays** and can be create like this:

$families = array

(

"Griffin"=>array

(

"Peter",

"Lois",

"Megan"

),

"Quagmire"=>array

(

"Glenn"

),

"Brown"=>array

(

"Cleveland",

"Loretta",

"Junior"

)

);

As you can see, these can easily get hard to read, so it is best not to have two many nested arrays. The information within these arrays, luckily, is easy to retrieve.

If we wanted to retrieve "Peter", the array index would look like this:

echo $families["Griffin"][0];

outputting: **Peter**

**Loops**

There are a handful of different types of loops, but they all essentially do the same thing. Loops allow you to write one piece of code and have it execute repeated until a certain condition is met.

**while -** loops through a block of code while a specified condition is true

while (condition) {

//code to be executed;

}

**do...while -** loops through a block of code once, and then repeats the loop as long as a specified condition is true

do {

//code to be executed;

}

while (condition);

example:

<?php

$i = 0;

do {

echo "<p>" . $i . "</p>";

$i++;

} while ($i < 10);

?>

**for -** loops through a block of code a specified number of times

for (init; condition; increment) {

//code to be executed;

}

commonly looks like this

for ($i = 0 ; $i < $var; $i++) {

//code to be executed;

}

**foreach -** loops through a block of code for each element in an array

foreach ($array as $value) {

//code to be executed;

}

you may not have seen a foreach loop before. These loops are specifically for arrays.

the foreach takes the name of an array and a variable name which it will store each value of the array as it loops through it.

**Functions**

Functions are blocks of code written together than can be called at anytime in a script and then executed, using taking parameters that it uses to calculate and return a new value from.

written like this:

function functionName ($parameter) {

$newValue = $parameter \* 2;

return $newValue;

}

This would be a simple function that would take a value and return a value twice as large as the parameter. "return" is a command that ends the function wherever its called and returns the specified value. anything written after a return command will not be executed (the exception is if the return is nested in an if statement that if not executed).

you could then call this function and store its results in a variable or output the infromation directly.

$value = 3;

$doubleValue = functionName($value);

echo functionName($doubleValue);

this would output : **12**

functions can have any number of parameters as long as its called with the same number of values

function multiply\_these($value1, $value2, $value3) {

return $value1 \* $value2 \* $value3;

}

$coolNumber = multiply\_these(34,21,74);

echo $coolNumber;

outputs : **52836**

**Includes**

One of the nicest features of PHP is its ability to pull other php files into the flow of another file and execute the code written inside of the seperate files. this is used largely for writing code that is repeated many times only once so that any changes made to the code only needs to be changed in one location. Writing the header and footer of a website is usually a good common practice since these are elements that are repeated across an entire site.

There are two ways to do this, with either the include() or require() function. Both are identical except that the require function will cause a fatal error if the file specified cant be found whereas include() only throws a warning, allowing the script to continue executing.

The functions take the file's path as a parameter

<?php include("header.php"); ?>

This can be included into anywhere within your PHP file, and this can be inserted into the flow of every webpage. A page could be as simple as this

<?php include("header.php"); ?>

<div id="content">

<!--content goes here-->

</div>

<?php include("footer.php"); ?>

This allows you to make changes to a header, footer, or any repeatable part of a site very quickly without having to go into each page to make a simple or complicated change. This becomes useful in even the smallest websites.

**MySQL**

MySQL is a relational database (Structured Query Language), a system of tabular information that is then retrieved through PHP and formatted and displayed as HTML to the user.

Tables are made of columns and rows that store simple bits of information. Tables often use FKs (foreign keys), to connect a cell of one table to a whole row of data in another. the designing of interaction between tables is a profession within itself.

A good intro can be found at<http://www.w3schools.com/php/php_mysql_intro.asp>

For detailed study on this you should look into INFO-210: Database Management Systems.

Although frameworks such as WordPress handle the creation of databases for the user and streamline the process of connecting to the server, its good to know how it works.

**Connecting to a MySQL database with PHP.**

In PHP, you need to establish a connection to the database, and then you can perform actions such as retrieving, storing, or changing information in the database. A connection is usually stored within a variable and then the variable is checked to see if a connection was successfully established. You need the HOSTNAME, USERNAME, and PASSWORD of the database in order to access it.

<?php

$con = mysql\_connect("localhost","username","passw0rD");

if (!$con)

{

die('Could not connect: ' . mysql\_error());

}

// Code

//then you must close the connection

mysql\_close($con);

?>

Although you can use PHP to create tables in your database, typically an graphical interface like PHPmyAdmin or SQL code is used to create and store the initial table information/structure. PHP is used largely to retrieve information stored and then change or add information to an already existing table.

**HTML5**

**(x)HTML is Dead. Long Live HTML(5)!**

This seems to be the rally cry among web designers. While the W3C has officially stated that HTML5 is NOT a standard yet and will not be completed until 2014 ( [bit.ly/f057Ed](http://bit.ly/f057Ed) ), there are a lot of proponents arguing we should be using HTML5 today( [bit.ly/l2s0ph](http://bit.ly/l2s0ph) ). That is not to say that you should forget your XHTML skills. XHTML is still the compliance standard, and the vast majority of all the XHTML code you learned is still used in HTML5.

Today we will cover the basics of HTML5 and CSS3. For further reading on this I highly recommend 2 books, *HTML5 for Designers* and *CSS3 for Designers*. Both are published by *A Book Apart*, and you can buy both as an eBook Bundle for $15 at http://www.abookapart.com/

An excellent free online source about HTML5 is http://diveintohtml5.org

**What is HTML5?**

HTML5 will be the new standard. We just don't know when yet. HTML5 is a work in progress. Currently, no browser has 100% HTML5 support., but the major browsers (Chrome, Firefox, Internet Explorer, Opera, and Safari) are continually adding new HTML5 features with each subsequent release. And all the major aspects of the language are already supported.

When people talk about HTML5 they are talking about more than just the HTML5 elements. The term HTML5 is being used as an umbrella term for HTML5, CSS3 & JavaScript. The lines between these are be blurring. We used to preach that (X)HTML was for markup *only*, CSS for display *only*, and JavaScript was the dynamic layer. But now you can create animations with CSS3 rules, or even invoke browser behaviors with HTML5 tags that used to require specific plugins.

**What's New in HTML5?**

The five most important/impressive upgrades to html are:

* **The Canvas Element**The canvas element is used for rendering graphs, graphics, games, or other visual images in the browser window without the need to rely on plugins.
* **The Video Element**Embedding videos without plugins? That's crazy! But not impossible. While it still doe not work exactly as intended, it is better than previous methods. But we covered this in DIGM-240 and nothing has changed yet.
* **Geolocation**  
  Finding you users location is nothing new to the web, but previously it was not reliable. Today, with HTML5, you can pinpoint a users location from WiFi towers and GPS.
* **Offline Web Applications**You can now create web applications that users can interact with, even without an internet connection. One typical scenario would be to allow your user to work with your app offline, then update with the cloud when a connection is re-established.
* **Form Elements & Input Type Attribute Values**HTML5 offers more form elements, with more functionality. Also, the input element's type attribute has many new values, for better input control before sending it to the server.

But we will not be learning how to use any of these this term. This term we will focus on the front end of HTML5, the new **elements** (tags) that you can use when coding a web page.

**Do I need to learn ALL of this?**

In a word, no. HTML5 is not one complete system. It is a collection of many features. You cannot detect HTML5 support in your browser, only specific HTML5 feature support.

Do not forget what you learned in XHTML all of that code still works in HTML5.

---

**Lets get started**

If you want to upgrade your XHTML document to HTML5 all you need to do is change the doctype to

<!DOCTYPE html>

All your code will still display the same and still validate. What this does is it now allows you to use the new HTML5 tags, and still validate.

We can also update your meta tag that defines your character set:

<meta charset="UTF=8">

We can also lose the type attribute from our javascript and css embeds. For all intents and purposes, javascript is the only scripting language we use on the web, and CSS is the only style sheet language.

<script src="file.js"></script>

<link rel="stylesheet" href="style.css">

**What else did we lose?**

Remember those XHTML rules regarding lowercase tags, quoted attributes, and closing tags? They are gone too. With HTML5 (nearly) anything goes. The W3C Validator for XHTML enforced a specific coding style, now it is up to you.

While the rules are casual, I would recommend that you keep a consistency in your documents. If you are going to use UPPERCASE tags then be consistent and do it everywhere.

frame, frameset, and noframes are obsolete. (who used them anyway?)

acronym is obsolete. (too many arguments about the difference between abbreviations and acronyms). We know the difference but remember: all acronyms are abbreviations, but not all abbreviations are acronyms.

font, big, center, strike… still obsolete.

For tables,

bicolor, cellspacing, cellpadding, and valign, these are gone too.

big is gone, but they kept small.

Note: <small> no longer means render at a smaller size, it means this is small print (mousetype or legalese) to be used for items like copyrights or terms and conditions.

**Upgrades**

<cite>

The cite tag has been redefined to mean "the title of a work" instead of a reference to other sources.

<a>

Previously the anchor tag was an inline element that wrapped inline content that was within a block element. but now you can use the a tag to wrap multiple block elements

<a href="index.html">

<h1>Site Name</h1>

<h2>Tag Line</h2>

</a>

The only rule is that you cannot nest an anchor tag inside another anchor tag.

**New Elements**

Most of these new elements will sound like they serve the same function as the div or other elements. But what it comes down to is semantics. In 2005 Google did some research ( http://code.google.com/webstats/ ) and compiled a list of most common classes, page names, attributes, etc. were. In HTML5 these were taken into consideration so that instead of using <div id="name"> or <div class="name"> we could just use <name>. Of course like XHTML these elements are predefined, we cannot go about using any element name we make up (that is XML). Here are some of the new ones that you can start using today…

<section>

The section element is used for grouping together thematically similar content. Typically with a header, possibly with a footer. Examples include chapters in a book, the various tabbed pages in a tabbed dialog box, or the numbered sections of a thesis. A web site's home page could be split into sections for an introduction, news items, contact information.

<article>

The article element is used for grouping together self-contained related content. This could be a forum post, a magazine or newspaper article, a Web log entry, a user-submitted comment, or any other independent item of content.

**The difference between section & article**

If you think of the WordPress standard home page where in xhtml you would probably use <div id="mainContent"> to display the list of all current blog entries, instead of the div, here you would use <section>. Then each blog entry that holds a title, meta data, and an excerpt, these would be organized using <article>.

<header>

Represents the "header" of a document or section of a document (not to be confused with <head>). The header element is typically used to group a set of h1–h6 elements to mark up a page's title with its subtitle or tagline. header elements may, however, contain more than just the section's headings and subheadings — e.g., version history information or publication date.

The W3C defines the header as a container for "a group of introductory or navigational aids." While each page should only have one masthead that contains the logo, tag line, and/or nav, you can/should use the header element multiple times on the page. Inside each article would be a header that contains the title and maybe the metadata.

<hgroup>

The hgroup element is typically used to group a set of one or more h1-h6 elements — to group, for example, a section title and an accompanying subtitle. It can only contain a group of <h1>–<h6> element(s), and it should be used for subtitles, alternative titles, and tag lines.

<time>

The time element can be used for dates, times, and combinations of both. This should also be accompanied by the datetime attribute. Represents a precise date and/or time in the proleptic Gregorian calendar. The time element encodes modern dates and times in a machine-readable way, so that, for example, user agents could offer to add an event to the user's calendar.

<nav>

Represents the navigation for a document. The nav element is a section containing links to other documents or to parts within the current document.

Not all groups of links on a page need to be in a nav element — only groups of primary navigation links. In particular, it is common for footers to have a list of links to various key parts of a site, but the footer element is more appropriate in such cases.

<aside>

Represents a section of a page consisting of content that is subsidiary to the content around the aside element, and which could be considered separate from that content. In WordPress you would use this for your sidebar which would contain category and tag links, widgets, and possibly advertising.

<footer>

Represents the "footer" of a document or section of a document. The footer element typically contains metadata about its enclosing section, such as who wrote it, links to related documents, copyright data, etc.

<mark>

Previously if you wanted to highlight a specific group of words you would use the span tag. But span is semantically meaningless. Enter <mark>.

Represents a run of text in one document marked or highlighted because of its relevance in another context.

When used in a quotation or other block of text referenced in a document, it indicates a highlight that was not present in the original document — e.g., a portion of text in an academic publication that has recently come under additional scrutiny

There are many more elements and attributes, but those are for you to research and learn on your own.

**HTML5 Enabling Script**

Believe it or not, there are still some people using browsers that do not support HTML5 (WTF, right?). While we cannot force a browser upgrade for them, we can force their browser to recognize these new HTML5 elements. We can use JavaScript to create elements for older browsers (ahem, IE) with the names of the new HTML5 elements. Then these "browsers" will magically recognize the elements and style them with your CSS.

Remy Sharp has created such a script. The script has gone through more than a dozen revisions since its inception, but this is the basic idea:

<!--[if lt IE 9]>

<script>

var e = ("abbr,article,aside,audio,canvas,datalist,details," +

"figure,footer,header,hgroup,mark,menu,meter,nav,output,"+

"progress,section,time,video").split(',');

for (var i = 0; i < e.length; i++) {

document.createElement(e[i]);

}

</script>

<![endif]-->

The <!--[if lt IE 9]> and <![endif]--> bits are conditional comments. Internet Explorer interprets them like an if statement: “if the current browser is a version of Internet Explorer less than version 9, then execute this block.” Every other browser will treat the entire block as an HTML comment. The net result is that Internet Explorer (up to and including version 8) will execute this script, but other browsers will ignore the script altogether. This makes your page load faster in browsers that don’t need this hack.

Remy Sharp has “minified” this script and hosted it on Google Project Hosting. ( http://code.google.com/p/html5shiv/ In case you were wondering, the script itself is open source and MIT-licensed, so you can use it in any project.) If you like, you can even “hotlink” the script by pointing directly to the hosted version, like this:

<!--[if lt IE 9]>

<script src="http://html5shiv.googlecode.com/svn/trunk/html5.js">

</script>

<![endif]-->

Now we’re ready to start using the new semantic elements in HTML5.

---

**CSS3 -- Review from WBDV-240 (no need to cover in class)**

CSS3 is completely backwards compatible, so you will not have to change existing designs. Browsers will always support CSS2.

Like HTML5, the CSS3 specification is still under development by W3C. CSS3 is also to be thought of as modules that are designed to be implemented independently of each other. The W3C explains the module approach:

*"Rather than attempting to shove dozens of updates into a single monolithic specification, it will be much easier and more efficient to be able to update individual pieces of the specification. Modules will enable CSS to be updated in a more timely and precise fashion, thus allowing for a more flexible and timely evolution of the specification as a whole."*

However, many of the new CSS3 properties have been implemented in modern browsers. The trick is to know *when* and *how* to use these new features.

When considering the visual experience of the site, you should think in two terms: *critical* and *non-critical*. Using cutting edge CSS techniques that are not supported in all browsers when you are dealing with critical branding elements would be risky.

Elements that have to do with Branding, Usability, Accessibility, and Layout are considered critical and you should make sure they are rendered properly in all browsers.

But when it comes to interaction, visual rewards, feedback and movement, it is far less critical that every browser render them identically, if at all.

**CSS3 Properties You Can Use Today**

**border-radius**

This rounds the corners of an element with a specific radius.

article {

border-radius: 10px;

}

**text-shadow**

adds a shadow to hyper text with options for direction, amount of blur, and color of shadow.

h1 {

text-shadow: 2px 2px 4px #666;

}

**box-shadow**

adds a shadow to an element. This uses the same syntax as text-shadow

article {

text-shadow: 2px 2px 4px #666;

}

**background**

I know, this isn't new but what you can do with it is. You can now add multiple backgrounds to a single element.

body {

background: url(image1.png) no-repeat top left,

url(image2.png) no-repeat top right;

url(image3.png) no-repeat bottom left;

url(image4.png) no-repeat bottom right;

}

To see the true power of this read:

http://designfestival.com/the-cicada-principle-and-why-it-matters-to-web-designers/

http://designfestival.com/cicada/

**opacity**

You can also define how opaque an element is by using numbers between 0 and 1, 0 being completely transparent, and 1 being completely opaque;

article {

opacity: 0.5; /\* 50% transparent \*/

}

**border-image**

You can now use an image to create a border

article {

border-image:url(border.png) 30 30 round;

}

**@font-face**

With the new @font-face css rule you can now server your own fonts and not have to render everything as image files. Like the video element, every browser requires a different font format so you will need .eot, .wott, .ttf, and .sag versions of each face in order to make sure that it renders on every browser on every platform. FontSquirrel.com has an @font-face generator so that all you need is a single format and they will generate all the other formats and the css for you. You do need to make sure that you have the rights to embed the fonts. make sure you read the EULA (End-User License Agreement) for the fonts that you are using before using them in the @font-face rule otherwise you may be held liable for copyright infringement.

More info about this rule at http://www.w3schools.com/css3/css3\_fonts.asp

**Vendor-Specific Prefixes**

Since CSS3 is a series of modules as these roll out in browsers they are considered experimental and therefore require what we call a Vendor-Specific Prefix. These codes are identical to the standard css rules, except the start with dash-vendorprefix-dash.

-webkit-border-radius: 10px;

Each browser has their own prefix, and all browsers ignore rules with prefixes they do not recognize.

Safari: -website-

FireFox: -moz-

Opera: -o-

Konqueror: -khtml-

IE: -ms-

Chrome: -chrome-

Example: Safari and FireFox both render the border-radius only with their own vendor specific version. But Opera, IE and Chrome will all render the rule without the prefix. When using prefixes you should always list the vendor specific rules first. This way, due to the cascade, should a safari or firefox implement the rule as a standard (sans prefix) then you guarantee that the final rule is the one that will be used.

article {

-webkit-border-radius: 10px;

-moz-border-radius: 10px;

border-radius: 10px;

}

This is a good thing. In the old days CSS hacks to get the page to render properly in every browser required extra slashes, dashes and underbars in and around multiple copies of the same rule. These were known as 'parser exploits' and were ugly, inconsistent, did not validate, and broke every time a browser updated. These vendor-specific prefixes stop this issue and are compliant.

<http://blogs.sitepoint.com/abolish-css-vendor-prefixes/>

<http://www.alistapart.com/articles/prefix-or-posthack/>

**CSS3 Generators**

With all of these different browser specific modules coding these gets a bit overwhelming at times, luckily, generous developers out there have developed web based generators. Here are a few of the most popular:

<http://css3generator.com/>

<http://www.css3maker.com/>

<http://www.noupe.com/tools/css3-and-html5-toolbox-starter.html>

<http://www.impressivewebs.com/css3-click-chart/>

<http://coding.smashingmagazine.com/2009/01/08/push-your-web-design-into-the-future-with-css3/>

<http://sixrevisions.com/css/css3-techniques-you-should-know/>

**Selectors**

Remember those selectors and pseudo-selectors we used in DIGM-240 like \*, +, and :hover? Well in CSS3 they have added a lot more.

<http://www.456bereastreet.com/archive/200601/css_3_selectors_explained/>

<http://tools.css3.info/selectors-test/test.html>

**Other CSS3 Features**

Other CSS3 features that we will not be diving into in this class include Transforms, Transitions, and Animations. You can use them in your project.

**Do Websites Need to be Experienced Exactly The Same**

**in Every Browser?**

<http://dowebsitesneedtobeexperiencedexactlythesameineverybrowser.com>

**HTML5/CSS3 Reset**

Remember that reset that we used in DIGM-240? Well, with new languages come new rules, and every rule needs to be reset. Here is an article and one of the most common HTML5 resets out there:

<http://www.cssreset.com/scripts/html5-doctor-css-reset-stylesheet/>

**HTML5/CSS3/JS Boilerplate**

HTML5 Boilerplate is the professional badass's base HTML/CSS/JS template for a fast, robust and future-proof site.

After more than three years in iterative development, you get the best of the best practices baked in: cross-browser normalization, performance optimizations, even optional features like cross-domain Ajax and Flash. A starter apache .htaccess config file hooks you the eff up with caching rules and preps your site to serve HTML5 video, use @font-face, and get your gzip zipple on.

Boilerplate is not a framework, nor does it prescribe any philosophy of development, it's just got some tricks to get your project off the ground quickly and right-footed.

<http://html5boilerplate.com/>

**Remember**

<http://www.html5rocks.com>

---

**Homework:**

**Read:**

<http://diveintohtml5.org/>

**Beginning WordPress 3: Make Great Websites the Easy Way**

• Chapter 6 - Creating a Basic Theme

• Chapter 7 - Creating an Advanced Theme

• Chapter 12 - Custom Content Types, Taxonomies and Fields

**Professional WordPress: Design and Development**

• Chapter 5 - The Loop

• Chapter 8 - Theme Development

**Wicked WordPress Themes**

• Chapter 4 - Theme frameworks

**Build:**

Keep making progress on your design ideas

Bring them into class for instructor review and grading

**Website Design Comps**

**Description:**

Redesign your portfolio website with the idea of conversion for use with database driven Content Management System. These should look exactly as you want them to look when the theme is complete.

**File Requirements:**

* design for 1280x960 screen (beware the safe area<http://www.designerstoolbox.com/designresources/safearea/> )
* 72dpi
* RGB
* jpg

**Name:**

* WBDV243-###\_FinalComp-PageName\_xxx.jpg

**Replace:**

* ### with your section number
* PageName with the type of page show (ie: Home, BlogEntry, Contact, etc..)
* xxx with your drexel id
* Example: WBDV243-001\_FinalComp-Home\_abc123.jpg

**Submit:**

**Midterm Project:**

HTML5/CSS3 Portfolio Website Redesign Comps

Due Midnight before Week 5

Description: Redesign your portfolio website with the idea of conversion for use with database driven Content Management System. These should look **exactly** as you want them to look when the theme is complete.

Submit: 72dpi, RGB jpg files

Name: WBDV243-###\_FinalComp-PageName\_xxx.jpg

Replace:

• ### with your section number

• PageName with the type of page show (ie: Home, Blog Entry, Contact, etc..)

• xxx with your drexel id

Example: WBDV243-001\_FinalComp-Home\_abc123.jpg

**Next week: Midterm Critique.**

Critiques are opportunities to learn and share what you’ve learned through demonstration and constructive criticism. When you present your work, you must be prepared to discuss it and address any questions concerning it. You should also take whatever criticisms you receive from faculty or students with the intent in which they are given, which is for you to *improve*. If you’re not presenting, it is expected that you give the presenter and their work your undivided attention and you are encouraged to ask questions, offer opinions and suggestions for improvement. You must take advantage of critiques for self improvement as an artist, and your participation in critiques will contribute to your class participation grade. Weekly assignments will be critiqued in class, as will the final project.

http://www.webdesignerstoolkit.com/