**Definitions –**

* **XHTML**: (Extensible Hypertext Markup Language) is “a reformulation of HTML 4.0 as an application of the Extensible Markup Language (XML).”
* **CSS**: (Cascading Style Sheets) is a style sheet language used for describing the presentation of a document written in a markup language.
* **HTML 5**: is a markup language used for structuring and presenting content on the World Wide Web. It is the fifth and current version of the HTML standard.
* **Client-side JavaScript**: These two terms are used in the context of web. Client-side means that the JavaScript code is run on the client machine, which is the browser. Server-side JavaScript means that the code is run on the server which is serving web pages.
* **Flash/Action Script**: ActionScript is used primarily for the development of websites and software targeting the Adobe Flash Player platform, used on Web pages in the form of embedded SWF files. ActionScript 3 is also used with Adobe AIR system for the development of desktop and mobile applications.
* **PHP**: is a server-side scripting language for web development but is also used as a general-purpose programming language. PHP originally stood for *Personal Home Page*, but it now stands for the recursive backronym *PHP: Hypertext Preprocessor*.
* **MySQL**: is an open-source relational database management system (RDBMS).
* **“Classic ASP”**: Active Server Pages (ASP), later known as Classic ASP or ASP Classic, is Microsoft's first server-side script engine for dynamically generated web pages.
* **ASP.NET**: is an open source server-side Web application framework designed for Web development to produce dynamic Web pages. It was developed by Microsoft to allow programmers to build dynamic web sites, web applications and web services.
* **CGI**: (Common Gateway Interface) is a standard way for web servers to interface with executable programs installed on a server that generate web pages dynamically. Such programs are known as *CGI scripts* or simply *CGIs*; they are usually written in a scripting language, but can be written in any programming language.
* **Java Servlets and JSP**: JSP is a webpage scripting language that can generate dynamic content while Servlets are Java programs that are already compiled which also creates dynamic web content. Servlets run faster compared to JSP. JSP can be compiled into Java Servlets. It's easier to code in JSP than in Java Servlets.
* **Drupal**: is a free, open source software that can be used by individuals or groups of users -- even those lacking technical skills -- to easily create and manage many types of Web sites. The application includes a content management platform and a development framework.
* **Joomla**: is a free and open-source content management system (CMS) for publishing web content. It is built on a model–view–controller web application framework that can be used independently of the CMS.
* **WordPress**: is an online, open source website creation tool written in PHP. But in non-geek speak, it's probably the easiest and most powerful blogging and website content management system (or

**Tuesday 3/29 -**

* **Install WAMP server (version 2)** – Windows, Apache, MySQL (version 5.6), PHP (version 5), and phpMyAdmin on a Windows machine.
* Do a google search for WAMPServer
* Download and install it – accept all of the defaults
* On some windows machines, IIS will already be installed (and the first web server "listens" for http requests on port 80). In that case, you’ll have to configure apache to listen on port 81.
* Open httpd.config from the WAMPServer menu on the task bar
* Find the 2 occurrences of "80" in the file and change the port to 81.
* Save the file
* Start the Apache service from the WAMPServer menu
* **Open-source** – free and continuing to be developed.
* **PHP Script** – server-side code, code runs on server side therefore can’t be seen on the client-side, is not good for large scale websites.
* **Web Server** – software, whose job is to serve web pages.
* **Database Server** – software,
* **Email Server** –
* **HTTP request** – in order to interact with a web server and gain access to a file.
* **Script Processors** – interpreters, first thing web browser does is find server-side code and send to appropriate interpreter.
* **AMP** – Apache (web server software) MySQL (database) PHP
* **Foreign Key** – enforces referential integrity

**Thursday 3/31 –**

* <?php (**opening delimiter**)

echo "This is a test"; (**echo = a predefined PHP function that inserts text into the http response stream**)

?> (**closing delimiter**)

* <?php

phpinfo(); (**predefined function to call the “WAMP” software**)

?>

* **Function** – not object-oriented
* **Method** – object-oriented, relate to objects and classes
* **Both** – are blocks of code that does something and it’s either a part of a class or not.
* <form id="page1" name="page1" method="post" action="lab1cOutput.php"> (**method attribute = post or get (appends user input to URL), action attribute (code that processes the page) = “filename”)**
* Server side code generally uses the **“name” attribute** as opposed to an **“id” tag**.
* [**http://localhost/”folder”/”file.php**](http://localhost/”folder”/”file.php)**”**
* **No directory browser** **-** it allows other people to be able to browse through the files on your server.
* **/\* “” \*/** - comment
* **//** - single-line comment
* **$\_REQUEST** – contains arrays for both “post” and “get” and “cookies,” blaa blaa blaa, a catch-all for info being sent from the user to server.
* **Key** – is the name attribute from the HTML form.
* **.** – concatenate, + is still for addition
* Variable and parameter names start with **$**.
* **number\_format** – predefined function

**Tuesday 4/5 –**

* **PHP.Net** – use as reference for PHP functions
* **isset -** is a predefined function in PHP that checks to see if the variable has been initialized.
* **Delimiter/ed** –
* **empty** – is a predefined function in PHP that checks to see if the variable has been
* **filter\_input** – is a predefined function in PHP

**Thursday 4/7 –**

* **Data = Model**
* **User Interface = View**
* **Logic = Controller**
* PHP code in HTML when you want to show the value of a variable to the user.
* Controllers job is to grab the data from the database, put the values into some variables, and then display the view. (Get the model, process the data, display the view).

**Tuesday 4/12 –**

* Name radio buttons the same to make them mutually exclusive choices.
* **Get method** appends info to the address bar URL
* <http://localhost/cs295p/lab3/ch07_ex1/display_results.php>?
* email=AAkins83%40gmail.com&
* password=Testing&
* phone=541-971-3058&
* heard\_from=Friend&
* wants\_updates=on&
* contact\_via=text&
* comments=This+is+a+test.
* **OR** with no info provided
* <http://localhost/cs295p/lab3/ch07_ex1/display_results.php>?
* email=&
* password=&
* phone=&
* contact\_via=email&
* comments=
* **%40** - @
* Always use the filter\_input data to get data from the user so they can’t inject code into your app.

**Thursday 4/14 –**

* **<form action=”.”** – form gets submitted to index page.
* **foreach** – takes one element in the array and puts it in the variable.

**Tuesday 4/19 –**

* **Regular Expressions** – String in PHP, declarative in nature as opposed to procedural,
* **5-digit zip code: /^\d{5}$/ -** / = opening delimiter, ^ = begins with, \d = digit, {5} = 5 digits, $ = ends with, / = closing delimiter. (Without begins and ends with, will check for 5 digits anywhere as opposed to exactly 5 digits.
* **/^\d{5}-\d{4}$/ -** nine digit zip code,must type dash to match
* /^\d{5}**(-\d{4})**?$/ - Zip code either style, ? Optional/either/or
* /^\d{3}**[\s.\-]?\**d{4}$/ - phone number (no area code), [] pick one character out of set of characters within, \s = whitespace, \ = escape character
* **preg\_match() –** returns 1 if pattern matched given subject, 0 if it does not, or FALSE if an error occurred. Int preg\_match ( string $pattern , string $subject );
* **i =** case insensitive
* <?php  
  // The "i" after the pattern delimiter indicates a case-insensitive search  
  if (preg\_match("/php/i", "PHP is the web scripting language of choice.")) {  
      echo "A match was found.";  
  } else {  
      echo "A match was not found.";  
  }  
  ?>
* ^^^ Searches if “php” is in the string.
* **filter\_input for email addresses**

**Tuesday, 4/26 –**

**Thursday, 4/28 –**

* **Default Parameter** – var = value, null if you don’t have one.
* **strtotime** – converts string to time
* int strtotime ( string $time [, int $now = time() ] )
* echo strtotime("10 September 2000"), "\n";
* “d Month YYYY”
* **Include vs. include\_once –**
* **Loan amortization schedule (bankrate.com)** – Date, Payment, Principal, Interest, Total Interest, Balance.
* Checkbox – do you want a Loan amortization schedule? Yes or No?
* If yes, print on bottom of page.
* **MVC Design Pattern –** Model View Controller, data = model, what user sees = view, logic = Controller.
* **Data for Mortgage Calculator –** the payment and the schedule itself (2-dimensional array)
* **View -**
* **Controller –** Gets the data, does the calculations, displays view
* **Associative arrays -** keys rather than numbers.
* $line = array();
* $line["Month"] = $month\_number;
* $line["Payment"] = $payment;
* $line["Principal"] = $principal;
* $line["Interest"] = $monthly\_interest;
* $line["Total Interest"] = $total\_interest;
* $line["Balance"] = $balance;
* **&** - pass by reference

**Tuesday, 5/3/16 –**

* **checkbox** - <input type=”checkbox” checked >
* **synchronous vs**. **asynchronous** –
* **AJAX** – allows user to use client and server-side code together, strategy

**Thursday, 5/12 –**

* **Atomic** – shouldn’t have first and last name in same field. Each field should contain one piece of data.
* **Use natural fields for primary keys** – L# is our primary key; not natural…
* **Composite Key** – multiple primary fields
* **No plural database names**

**Tuesday, 5/31 –**

* Mysql or Mysqli
* “Writing code the target specific database platforms should be avoided.”
* **Database Abstraction Layer** – to enable connection to lots of different database back-ends in addition to MySQL without having to change PHP code. All are object oriented.
* Whenever you interact with a database from a prog. Lang. you have to
* Connect to the database
* Issue a SQL command
* If SQL select command, have to loop through set of records that is returned
* Disconnect from the database
* Syntax for creating an object from any class
* new ClassName (arguments);
* **How to connect to a MySQL database:**
* $dsn = 'mysql:host=localhost;dbname=my\_guitar\_shop1';
* $username = 'mgs\_user';
* $password = 'pa55word';
* // creates PDO object
* $db = new PDO($dsn, $username, $password);
* **Prepare –** method of PDO class for preparing a SQL statement
* **->** - references to objects
* **SQL injection** – why we don’t concatenate “category\_id” ourselves
* $query = 'SELECT \* FROM products
* WHERE categoryID = :category\_id';
* $statement = $db->prepare($query);
* $statement->bindValue(':category\_id', $category\_id);
* $statement->execute();
* **fetch()** – method you can execute on a statement, “fetch a record from the database,” returns 1-dimensional array
* **closeCursor() –**
* **try block** – around code that is likely to have errors
* **catch block** – around error handling code, to catch the error
* **Syntax for a try/catch statement** –
* try {
* // statements that might throw an exception
* } catch (ExceptionClass $exception\_name) {
* // statements that handle the exception
* }
* **dsn** – data source name
* **PDO** -
* **Lab 8** –
* Slides starting at #29
* Steal code from product\_viewer
* Forum

**Tuesday, 5/17 –**

* **Charfield – fixed length field**
* **Varcharfield – not fixed length field**
* **Null column – can’t be empty**
* **Auto-increment – auto number**
* **Download community addition MySQL workbench – is a java application much like Oracle SQL Developer that many of you used (or will use) in 275.** [**http://dev.mysql.com/downloads/workbench/**](http://dev.mysql.com/downloads/workbench/)
* **Lots of software exists to help you work with MySQL** [**http:// HYPERLINK "http://www.databasejournal.com/features/mysql/slideshows/top-10-mysql-gui-tools.html"HYPERLINK "http://www.databasejournal.com/features/mysql/slideshows/top-10-mysql-gui-tools.html" HYPERLINK "http://www.databasejournal.com/features/mysql/slideshows/top-10-mysql-gui-tools.html"www.databasejournal.com/features/mysql/slideshows/top-10-mysql-gui-tools.html**](http://www.databasejournal.com/features/mysql/slideshows/top-10-mysql-gui-tools.html) **is an “article” that briefly describes 10 most popular tools for MySQL.**
* **Before we look at MySQL client tools**

**Make sure mysql is installed and running on your machine**

**C:\wamp\bin\mysql\**

**Control panel services tool – wampmysqld - start/stop/automatically start on startup**

**This will be different for those of you who installed mysql manually or who are using xampp or some other #amp server products**

* **In mysql databases and tables are implemented as files, because of this if run on linex all of these things would be case sensitive.**
* **In bin folder are command line tools that I can use to interact with mysql**
* **-- database single line comment**

**Tuesday, 5/24 –**

* **Select statements are list of fields and from is table where is conditional (price is over $500) orderby how to sort , where and order by are optional**
* **Aggregate function to summarize all of the data in the table as part of the select list.**
* **Group by clause**
* **Alias?**
* **To count –**
* SELECT COUNT (\*) AS productCount
* FROM products
* Result = 10
* The only time you will use the \* as one of the parameters for an aggregate function is COUNT
* Ignore HAVING clause except to say, if you want to get a subset of aggregated data use this clause.

**Thursday, 5/26 –**

* **Chapter 18:**
* **Four main clauses of the** **SELECT** – select, from, where, and order by.
* **SELECT** – specifies how many columns to retrieve.
* **FROM** – specifies the *base table* or tables to retrieve the data from.
* **WHERE** – specifies the rows to retrieve, search conditions**.**
* **ORDER BY** – specifies how to sort the rows.
* **\*** - wildcard, indicates all should be retrieved.

**Tuesday, 5/31 –**

* Mysql or Mysqli
* “Writing code the target specific database platforms should be avoided.”
* **Database Abstraction Layer** – to enable connection to lots of different database back-ends in addition to MySQL without having to change PHP code. All are object oriented.
* Whenever you interact with a database from a prog. Lang. you have to
* Connect to the database
* Issue a SQL command
* If SQL select command, have to loop through set of records that is returned
* Disconnect from the database
* Syntax for creating an object from any class
* new ClassName (arguments);
* **How to connect to a MySQL database:**
* $dsn = 'mysql:host=localhost;dbname=my\_guitar\_shop1';
* $username = 'mgs\_user';
* $password = 'pa55word';
* // creates PDO object
* $db = new PDO($dsn, $username, $password);
* **Prepare –** method of PDO class for preparing a SQL statement
* **->** - references to objects
* **SQL injection** – why we don’t concatenate “category\_id” ourselves
* $query = 'SELECT \* FROM products
* WHERE categoryID = :category\_id';
* $statement = $db->prepare($query);
* $statement->bindValue(':category\_id', $category\_id);
* $statement->execute();
* **fetch()** – method you can execute on a statement, “fetch a record from the database,” returns 1-dimensional array
* **closeCursor() –**
* **try block** – around code that is likely to have errors
* **catch block** – around error handling code, to catch the error
* **Syntax for a try/catch statement** –
* try {
* // statements that might throw an exception
* } catch (ExceptionClass $exception\_name) {
* // statements that handle the exception
* }
* **dsn** – data source name
* **PDO** -
* **Lab 8** –
* Slides starting at #29
* Steal code from product\_viewer
* Forum