PHP

- Originally stood for “Personal Home Page”, but now has a recursive acronym “PHP Hypertext Preprocessor”
- PHP is a server side scripting language
- Components for an interactive website:
  - HTML – The tag structure that makes up the actual page
  - CSS – The styling applied to the HTML tags
  - JS – Client side scripting language
  - PHP – Server side scripting language
  - Next Week: MySQL – Database engine

PHP

- With PHP being a server side language, we can use it to generate content to be sent to the client, or perform any number of tasks before the page is rendered and sent to the client
- One of the most powerful features of using a server side language, is the ability to access a database for information before rendering the HTML page
- We will look at this in more detail next week when we get into MySQL
- Today, we are going to focus on the basic structure of PHP

PHP Setup

- For the next several weeks, we will be working with PHP
- The first change we need to make, is that the file extension we have been using will not work, we need to switch to using ".php" so the server knows to run the file as PHP code before sending the results to the browser
- In Dreamweaver
  - Edit->Preferences->New Document->Default Document-> Change to PHP

PHP Structure

- Now that we have changed our default document type, if we right click in the file explorer and chose “New File”, it will automatically have the “.php” at the end instead of “.html”
- When we first open this new file, we notice that there is no difference between this and the HTML file we were creating before
- PHP code is designed to allow being written in line with standard HTML

PHP Hello World

- Let’s do a simple “Hello World” example:
    - Put the following code somewhere inside of your <body> tags

```
<?php
     echo “Hello World”;
?>
```

PHP Hello World

- Now, lets enhance that by adding <h1> tags around the Hello World text
- Now move the <h1> tags outside of the code block
- We are executing the code in place, and replacing the code blocks with whatever is supposed to be rendered as HTML content, which is what is sent to the clients browser

PHP Comments/Variables

- So, we can use PHP to write HTML, CSS, and JS in the page if we chose
- Comments in PHP:
  - // single line
  - /* can be used over multiple lines */
  - # alternative single line (not seen very often)
- Like in JS, variables in PHP do not require you to declare a data type
- To create a variable, you just need to assign it a value
  - $variable = “Hello World”;

PHP Variables

- To output variables, we can use just the variable, concatenate them with strings, or perform an automatic concatenation using double quotes
  - echo $var;
  - echo “My var “ . $var . “ printed here”;
  - echo “My var $var printed here”;
    - Putting the variable inside of single quotes prints it as text
    - echo ‘My var $var printed here’;

PHP Variables

- PHP arrays can be created several different ways:
  - $arr = array(5, 6, 7);
  - $arr = array(“age” => 23, 5 => “name”, “street” => “str”)
  - $arr[] = “val1”
  - $arr[“key2”] = “val2”;
- Accessing array items involves knowing the key for direct access, or using for/foreach loops
- We can also use the print_r() statement
- Multidimensional arrays are also supported

PHP Variables

- Just like in JS, PHP automatically converts variables between types based on the actions being taken with them
  - $var1 = $var2 + $var3;//treats all like numbers
  - $var1 = $var2.$var3;//concatenates like strings
  - if($var1){}//treats like a boolean (zero/null is false, everything else is true)
- Also like JS, if we want the same value and data type, we need to use === operator or !== operator
  - Some functions can return false or zero, where zero is valid and false means no solution – strpos()

PHP Comparisons

- if, else, elseif, switch
- ==
- ===
- !=
- !==
- <>
- >
- <
- >= <=

- + - * /
- %
- **
- ++ --
- &&
- ||
- !
- xor

PHP Variables

- Variable scope is very similar to JS
  - Variables created outside of a function are global
  - Variables created inside of a function are local
  - To access a global variable inside of a function, the global keyword must be used

```
function myFunction()
{
    global $var1;
}
```

PHP Functions

- Just like in JS, we use the keyword function to declare new functions
- We can create functions with or without parameters
- Just like in JS, we don’t need any kind of data type assigned to the parameters

PHP Functions

- There are a lot of built in PHP functions, so just like with JS, I will touch on a few common ones, but as you are working on projects, be sure to look and see if there are prebuilt functions for tasks you are trying to do
- for
- foreach
- while
- do while

PHP Functions

- explode / implode
- lcfirst / ucfirst / ucwords
- str_replace / str_ireplace
- str_split
- str_word_count
- strstr / stristr
- strpos / stripos
- strlen
- strrev
- strrpos / strripos
- strtolower / strtoupper
- substr
- substr_replace
- substr_count

PHP Functions

- array_fill
- array_key_exists
- array_keys
- array_merge
- array_merge_recursive
- array_search / in_array
- array_unique
- asort / arsort
- count / sizeof
- ksort / krsort
- sort / rsort
- shuffle

PHP Functions

- header
- time / microtime
- date
- mktime
- strtotime
- exit
- uniqid

Form Handling

- When creating an HTML form, there are several attributes we can set that determine how we will handle the data from the form when it is submitted
  - Action
    - Default is same page
  - Method
    - Default is “GET”, often changed to “POST”
    - GET shows up in the URL, POST does not
  - Enctype
    - Set to “multipart/form-data” if you have a file upload

Form Handling

- On the PHP side, the data submitted by the form is stored in predefined arrays:
  - GET : $_GET[]
  - POST: $_POST[]
  - Both: $_REQUEST[]
- You should only use $_REQUEST if you need your code to accept GET or POST data
- Normally we chose one method or the other for any given form submission

Form Handling

- Once a form is submitted, we need to have PHP validate the data in each field to confirm we are getting what we expect
  - We often use strlen() and isset() with other basic logic to ensure each field has the data we expect
- If we are handling the form submission on the same page as the form, we also want to check and see if the form was even submitted before trying to validate
  - if(isset($_POST[‘submitButtonName’]))

Form Handling

- Let’s set up a simple form and do some basic validation tasks

File Handling

- There are two types of File Handling I want to discuss today
  - Files uploaded via a form submission
  - Working with files on the server

File Uploads

- As discussed earlier, we need to set the enctype of the form so it will accept the file uploads
- When the form is submitted, the server takes the file and stores it in a temporary location
- PHP then builds another array variable with information about the file uploaded and the current temporary location/filename called $_FILES[]
  - name, type, tmp_name, error, size

File Uploads

- The primary tasks we need to perform for a file upload are:
  - Set form to accept files
  - Validate that the file is of an allowed type/size/etc
  - Move the file from the temp location to a permanent location
    - During this move, we should generally set a new unique name (often we use a timestamp)

File Uploads

- Validating a file type is usually done based on the file extension, and not based on the “type” array element
  - PHP provides a function for this:
    - pathinfo(“filename.ext”, PATHINFO_EXTENSION)
- Generally we want to chose a small set of file types to allow, as opposed to choosing a few not to allow
- This is a security practice to prevent a user from uploading a malicious script/page of some type to your server

File Uploads

- Setting Folder Permissions
- Path to folder
  - G:\\PleskVhosts\\[your_root domain_here]\\[sub_domain_folder]\\[upload_folder]\\
  - G:\\PleskVhosts\\csci3000.com\\sp2019.csci3000.com\\uploads\\
  - double slashes used because they are escape characters

File Uploads

- When moving the file out of the temporary directory, we need to have a directory coded in for where it will go, and a way to ensure the filename is unique (often adding a timestamp)
  - uploads_directory/filename_unixtimestamp.ext
- Then we can use a PHP function to get the file moved to where it belongs
  - move_uploaded_file($tempName, $newPathAndName)

File Uploads

- Depending on what we are having the user upload, sometimes we will want the upload directory to not be visible from the web, so we create the folder at the same level as httpdocs
- Other times, we want to make the files accessible (often this is the case with images), so we will put the upload folder somewhere inside of httpdocs

File Handling

- Now that we have a file on the server, we want to access it for use on our site
- One simple task we can do is list all the files in a given directory
  - scandir()
  - is_dir()

File Handling

- Another task we sometimes need to do is force a file to be downloaded, instead of being opened in the browser
- This process can also hide the true path to the file, or access files that are not normally accessible by the browser
  - http://php.net/manual/en/function.readfile.php

File Handling

- The readfile() command can also be used to store the contents of a file into a variable for processing by PHP (maybe reading a text file for instance)
- We can also use encryption and decryption functions to store the files on the server in an encrypted way, and only decrypt them when sending them as a download to an authorized user

Includes

- Sometimes the code for a site can get very bulky, or you end up with several functions you want to use on different pages
- To handle this, we can create a PHP file that isn’t designed to render as web content, but is more of a library of functions we want to be able to use
  - include / include_once
  - require / require_once

Includes

- It is very common to have these pages of functions, and often some additional constants we need to set up
  - We can help secure these kinds of pages we don’t want the user to browse to by using:
    - get_included_files()
  - Which returns an array of all the pages being included
  - We can then use this to tell if we should redirect the user, or allow the page to be included

Classes / Object Oriented

- Sometimes it can be helpful to build our PHP with classes to be Object Oriented
- The basic structure is very similar to how you might build a class using Java

```
class MyClass
{
}
```

Classes / Object Oriented

- To create an object of the class, we use the ‘new’ keyword
  - $obj = new MyClass;
- To access instance data or class methods, we use an arrow
  - $obj->data1;
  - $obj->afunction();

Classes / Object Oriented

- Some functions we will want to implement in our classes:
  - public function __construct()
  - public function __toString()
- We can also build classes from others (parent/child, inheritance)
  - class MyOtherClass extends MyClass
  - public function __construct()

    ```
    {
       parent::__construct();//calls parent constructor
    }
    ```

Classes / Object Oriented

- public/private/protected work the same as in Java
  - public is accessible from the created object
  - private is only accessible inside the given object
  - protected is only accessible inside the given object and inside child objects

Classes / Object Oriented

- Sometimes it can be helpful to build our PHP with classes to be Object Oriented
- The basic structure is very similar to how you might build a class using Java

```
class MyClass
{
}
```

Classes / Object Oriented

- To create an object of the class, we use the ‘new’ keyword
  - $obj = new MyClass;
- To access instance data or class methods, we use an arrow
  - $obj->data1;
  - $obj->afunction();

Classes / Object Oriented

- Some functions we will want to implement in our classes:
  - public function __construct()
  - public function __toString()
- We can also build classes from others (parent/child, inheritance)
  - class MyOtherClass extends MyClass
  - public function __construct()

```
{
  parent::__construct();//calls parent constructor
}
```

Classes / Object Oriented

- public/private/protected work the same as in Java
  - public is accessible from the created object
  - private is only accessible inside the given object
  - protected is only accessible inside the given object and inside child objects

Databases

- What is a database?
  - A structured set of data
- In the case of MySQL, our database is organized into tables, and each table has columns
  - Think about how Excel holds data in rows/columns as being a table, and each sheet is a new table

Create a DB

- We will need to log in to our hosting provider
- Manage our hosting account
- This should put us on the screen with our various sub-domains
- We can click “Add New Database”
- Choose a DB name “name_class_examples”
- Go ahead and create an account for connecting to the database (we will be putting this username/password into PHP code later)

Create a DB

- Now we have our first database
- Click on the “Webadmin” link
  - At this point, it should automatically log you in, the account you created earlier will be used when we connect from PHP later, or if you want to use another tool to manage your DB
- Click on your database on the left side “name_class_examples”
  - The first item we see is that there are no tables, so it is asking us to create one

Create Tables

- Choose a table name
- Start with four(4) columns

- Now you should see a screen to name the columns and set some additional properties
  - First Column:
    - Name: ID
    - Type: INT
    - Index: PRIMARY
    - A_I: Checked (auto-increment)

Create Tables

- Second Column:
  - Name: FIRST_NAME
  - Type: VARCHAR
  - Length/Values: 50
- Third Column:
  - Name: LAST_NAME
  - Type: VARCHAR
  - Length/Values: 50
- Fourth Column:
  - Name: EMAIL
  - Type: VARCHAR
  - Length/Values: 100
- Fifth Column:
  - Name: BIRTHDAY
  - Type: DATE

Create Tables

- We can click the “Preview SQL” to see the command that is about to be run to create our table
- Then click “Save”
- Now in the left hand menu, we can expand the new table and see the columns and indexes, as well as options to add new

MySQL Queries

- The first SQL Query we want to look at is inserting, so we can put some data into our table
- From phpMyAdmin, we can click on the table name, and then click on “Insert” for a GUI version
- We can also click on SQL, and write out the actual SQL Query
    - There are some helpers in phpMyAdmin to get us started
    - Click on “INSERT” at the bottom of the window to pre-populate the basic insert query

MySQL Queries

- Now, you can see the basic insert queries consists of some reserved words:
  - INSERT INTO
- Followed by the table name, in back ticks (key to the left of the number 1 on the keyboard)
  - \`ex_first_table\`
- Then a list of the columns in parenthesis
- Another reserved word:
  - VALUES
- And finally the list of values to insert

MySQL Queries

- In this particular query, we can remove the \`ID\` column, as we have set that to auto-increment, so it will automatically become the next integer value
- Go ahead and fill in the fields with some information (doesn’t need to be yours, just make something up)

MySQL Queries

- Obviously we don’t want to sit down and add a bunch of data by hand
- Normally we might set up a web form to collect the data, and put it into this table, or get the data from another source
- I found a tool online to generate random data, so we will use that to get our table filled with plenty of data for us to work with
- http://www.generatedata.com/

MySQL Queries

- Now that we have a bunch of data inserted into our table, lets look at using queries to select a portion of that data
- To begin with, we can just select everything:
  - SELECT * FROM `ex_first_table`
- Now, lets add a WHERE clause, to narrow our results
  - SELECT * FROM `ex_first_table` WHERE `EMAIL` LIKE ‘%.edu’
  - Using the LIKE comparison, with a wild card (%), we can select all the emails that end in ‘.edu’

MySQL Queries

- The LIKE keyword performs a case insensitive search in MySQL (as well as MSSQL, but not in Oracle)
- If we are still getting to many results, we can limit these:
  - SELECT * FROM `ex_first_table` LIMIT 10
    - Note: our LIMIT command must always come at the end of the statement
- We can also sort the data
  - ORDER BY `LAST_NAME` DESC

MySQL Queries

- When working with dates, we can use standard math operators:
  - SELECT * FROM `ex_first_table` WHERE `BIRTHDAY` < ‘1950-01-01’
- Instead of selecting all of the columns, we can also select only the columns we are interested in
  - SELECT `FIRST_NAME`, `LAST_NAME` FROM `ex_first_table`

MySQL Queries

- We may also want unique rows, based on some criteria
  - SELECT DISTINCT `BIRTHDAY` FROM `ex_first_table`
- When I tried this before, there were no duplicates, so to find unique years of birth:
  - SELECT DISTINCT SUBSTRING(`BIRTHDAY`, 0, 4) AS `BIRTH_YEAR` FROM `ex_first_table` ORDER BY `BIRTH_YEAR`

Relational Databases

- The concept behind a relational database, is that multiple tables store information that relates to each other
- Basically, instead of having a single table with 100+ columns and a lot of repeated values, we will have several tables and use index numbers to connect a row from one table to a row in each other table

Relational Databases

- Table 1 – Has student information: name, email, birthday, etc
- Table 2 – Has a list of courses
- Table 3 – Is a connection between the student and what courses they have taken, so there are multiple rows in this table that point at each student with different courses
- Let’s use our first table we built as an example for table 1 and set up the table 2 and 3 to see how this works

Relational Databases

- New Table
  - We need a primary index and a course name field
- New Table
  - We need a primary index, and two more integer fields, one will point to a given student, and the other will point to a given course
- Normally we would have the database reinforce the relationships, but this is currently disabled in MySQL on GoDaddy

Relational Databases

- Let’s use the online tool to create some dummy data for us again
- Once that is all imported, we want to select from one table, and join the other tables to the one we select from

Relational Databases

```
SELECT
		`ex_second_table`.`COURSE`,	`ex_first_table`.`FIRST_NAME`,
		`ex_first_table`.`LAST_NAME`
FROM `ex_third_table`
JOIN `ex_first_table`
		ON `ex_first_table`.`ID` =
			`ex_third_table`.`STUDENT_ID`
JOIN `ex_second_table`
		ON `ex_second_table`.`ID` =
			`ex_third_table`.`COURSE_ID`
ORDER BY `ex_first_table`.`ID`
```

MySQL

- Here are a few MySQL resources (I'll post these in D2L as well)
- Tutorial:
  - http://www.tutorialspoint.com/mysql/mysql-select-query.htm
- MySQL 5.5 reference:
  - https://dev.mysql.com/doc/refman/5.5/en/index.html

PHP Data Object (PDO)

- PDO is the currently preferred way to access a database
- The object can be configured to connect to several different database types, allowing you to learn one set of functions for multiple databases

PDO Connection

- $db = new PDO($connectionString, $username, $password)
  - $connectionString:
    - mysql:host=123.123.123.123:123;dbname=database;
- Once the object is created, you can invoke several methods to invoke SQL queries

PDO Basic Select

- $query = $db->query(“SELECT * FROM table”);
  - This creates a query object, but does not actually run the query
- $results = $query->fetch(…);//return next row

  ->fetchAll(…);//return all rows
- Inside of the fetch command, we want to specify how the data should be returned
  - PDO::FETCH_ASSOC
  - PDO::FETCH_NUM
  - PDO::FETCH_BOTH

PDO Basic Select

- Often times when using fetch(), we will use a while loop
  - while($row = $query->fetch(PDO::FETCH_ASSOC))
- And we may end the while loop early, meaning we don’t actually retrieve all the rows, saving some processing time
- If we need to know the number of results, or know we need to get everything, we can use fetchAll()
  - foreach($query->fetchAll(PDO::FETCH_ASSOC) AS $key => $val)

PDO Security

- Using the query method can be a simple way to get your query executed, but if you are using any data the user provided to build your query, there is a chance of SQL Injection
- To avoid this, we use the prepare() method, with bindParam() methods
- The bindParam() method will ensure that the data you are adding to the query is properly sanitized before the SQL is executed

PDO Prepare

- The prepare() method is also helpful in building an object that contains a query we want to run multiple times, for instance, when inserting several rows at a time
- $select = $db->prepare(“SELECT * FROM table WHERE ID = ?”);
- $select->bindParam(1, $id);
- $id = 5;
- $select->execute();
- $results = $select->fetchAll();

PDO Prepare

- $select = $db->prepare(“SELECT * FROM table WHERE ID = :id”);
- $select->bindParam(“:id”, $id);
- $id = 5;
- $select->execute();
- $results = $select->fetchAll();
- $id = 6;
- $select->execute();
- $results = $select->fetchAll();

PDO Insert

- Insert/Update/Delete commands are all executed the same way as the select examples we did, using query() or prepare()
- There are a few additional methods that are helpful to us with these items:
  - $db->lastInsertId();
  - $db->errorCode();
  - $db->errorInfo();
  - $update->rowCount();//doesn’t work with select queries
  - errorCode and errorInfo can also be run off of prepare() objects