CST8285

Introduction to PHP

What is PHP?

- PHP is a recursive acronym for PHP: HyperText Processor
  - Used to be called Personal Home Page, but that was boring.
- PHP is a **server-side** scripting language
- PHP is free to use, well maintained, and widely available
  - Most (dare I say all) hosting companies support PHP

Why use PHP?

- It makes pages easier to update
  - You can store repeated pieces of code (web page header, footer, navigation) and re-use them across pages.
- You can save user information
  - Data sent to the server can be saved in a database, and recalled later
- You can change your page without changing your code.

How to create a PHP file

- A PHP file is defined by the **.php** extension
- Any HTML page can be turned into a PHP page by changing the extension to **.php**

Adding PHP to a PHP file

- PHP code is contained in the following tags:

  ```
  <?php          ?>
  ```
- These tags can be placed anywhere inside a PHP document.
- An entire file can be a PHP file by placing the opening tag at the beginning of a file, and the closing tag at the end.

XAMPP, Apache and the htdocs file

- XAMPP should have installed Apache, MySQL and PHP (the AMP in XAMPP) on your computer
- Apache is a free, open source, and popular web server
- By default, XAMPP designates the **htdocs** folder in the XAMPP installation directory as the Apache web directory

Using a PHP file

- PHP files must be on a server to run.
- Unlike JavaScript, opening a PHP file directly in a browser will do nothing.
- To run a PHP file on your computer, simply copy it to the **htdocs** foder.
- Open XAMPP, and make sure that the Apache service is running.
- Navigate to http://localhost/yourFileName.php

echo and print()

- echo and print() both write text inside the HTML document **before** it is sent to the browser
- There are a few differences between the two, but for the most part, they have the same functionality.
- Example!

Declaring variables

- Variables are defined using the dollar sign (`$`)
  - ex: `$variableName`
- Variable names must start with an underscore (_) or a letter (A-Z, a-z)
  - `$3names` is an invalid variable, because it starts with a number
  - `$_3names`, however, is valid, because it starts with an underscore

Assigning Variables

- PHP uses the equals sign (=) as the assignment operator
- The default way to assign variables in PHP is to assign by **value.**
  - If $x = $y, then $x will be assigned the **value** of $x.
- However, PHP does offer a way to assign by **reference**, using the ampersand (&).
  - If $x = **&**$y, then $x will be a **reference**, or alias, for $y.
- Example!

Declaring functions

- A function is defined using the following syntax:

```
function myFunction($arg1, $arg2){
    echo “Do something here;
    return $aValue
}
```

- Like variables, function names must start with a letter or an underscore
- Any valid PHP code can be put in a function
- Example!

Variable Scope

- **PHP variable scope works different than JavaScript or Java!**
- The scope of a variable is the context in which the variable is defined
- That means that unless specified, variables declared outside of a custom function **are not available within that function!**
- To use an out of context variable, you can use the **global** keyword

PHP Data Types

- PHP is **loosely typed.** Meaning, you do not need to specify the data type.
- Having said that, PHP does support different primitive data types:
  - Boolean
  - Integer
  - Float (double)
  - String
- There are others, but we'll save those for later.

Boolean

- Holds a true or false value.
- Defined by assigning a variable either the true or false keyword (**not the word in a string)**

- Ex: $myBool = true;

- The true and false keywords are **case-insensitive.**

Integer

- Can be specified in decimal, hexadecimal, octal or binary notation
- Defined by assigning an integer value to a variable
  - Ex: `$myInt = 8675309;`
- PHP does not have a division operator for integers. An integer divided by an integer will yield a **float.**

Float

- Defined by assigning a float value to a variable
  - Ex: $float1 = 11.567;
  - $float2 = 1.9e4;
  - $float3 = 9E-8;
- Floating point numbers have issues with precision when dealing with very large numbers.

String

- Strings can be defined in four ways:
  - Using single quotes - $myVar = ‘this is a string’;
  - Using double quotes = $myVar = “this is also a string”;
  - Using Heredoc (not covered)
  - Using Nowdoc (not covered)
- To concatenate strings, use the dot .

ex: `$helloWorld = “Hello “ . “World!”;`

PHP gettype()

- If you need to know the data type of a variable in PHP, use getType.
- Ex:
  ```
  $x = 10;
  echo gettype($x); //would
  output ‘integer’
  ```
- The output for gettype() that is passed a *float* data type will return “**double”.** This is due to historical reasons.
- Example!

Single vs. Double quotes

- In PHP, there is a difference between using single and double quotes
  - If you use single quotes, the string will be interpreted literally.
  - If you use double quotes, certain evaluations will occur. Most notably, variables will be evaluated.
- Example time!

PHP Arithmetic Operators

Given $y = 5

Symbol	Function	Example	Result ($x)
+	Addition	$x = $y + 2	7
-	Subtraction	$x = $y – 2	3
*	Multiplication	$x = $y * 5	25
/	Division	$x = 25 / y	5
%	Modulus	$x = $y % 3	2

PHP Comparison Operators

Given $x = 5, and $y = 10

Operator	Operation	Example	Result
==	Equal (value)	$x == "5"	true
===	Identical (value and type)	$x === "5"	false
!=	Not Equal (value)	$x != $y	true
<>	Not Equal (value)	$x <> $y	true
!==	Not Identical (value and type)	$x !== "5"	true
<	Less Than	$y < $x	false
>	Greater Than	$y > $x	true
<=	Less than or equal to	$y <= 10	true
>=	Greater than or equal to	$x >= $y	false

PHP Logical Operators

Given `$x` = 5 , and `$y` = 10

Operator	Function	Example	Result
and	And	`$x < $y and $x > 2`	true
or	Or	`$x > $y or $x > 2`	true
xor	XOR	`$x < $y xor $x  > 2`	false
!	Not	`!($x > 10)`	true
&&	And	`$x < $y && $x > 2`	true
||	Or	`$x > $y || $x > 2`	true

Please visit the php.net Operator Precedence Page for more information about the difference between the two **'and'**, and two **'or'** operators.

http://www.php.net/manual/en/language.operators.precedence.php

Conditional Statement (if)

- Two ways of defining an if statement
  - If you only want to execute one statement, you can write it as follows:

```
if($x > $y)
    echo ‘$x is the larger number’;
```

  - If you want to execute one or more statements, the syntax is as follows:

```
if ($x > $y){
  echo ‘$x is the larger number’;
  $y++;
}
```

$_GET and $_POST

- $_GET and $_POST are associative arrays that give PHP scripts access to form data sent from a browser
- $_GET and $_POST are **superglobals**, which means that they are accessible anywhere in a PHP script.

$_GET

- When a URL is sent to the server, it can contain **parameters.**
  - Ex: http://localhost/hello.php**?yourName=Matt**
- The parameters are passed to PHP scripts in an associative array ($_GET)
- To get the above parameter in PHP, do the following:
  - $userName = $_GET["yourName"];

$_POST

- $_POST is an associative array of variables sent to the server via the HTTP POST method.
  - If an HTML form has its method set to “post”, the form values are in the $_POST array.
- To access a form field’s value in PHP, the **name attribute** of the form field is used to search the $_POST array
  - Ex: $firstName = $_POST[“firstName”];

isset()

- isset() is a PHP function that takes a variable as a parameter. It returns true if the variable is set (ie: has a value), otherwise it returns false.