

Web Development From the Ground Up, a Series for Novice Computer Users

Lecture 4

Lecturer: Philip Matuskiewicz

Thursday: October 6, 2009 / 7pm / Bell 221

Email: phil@famousphil.com or pjm35@buffalo.edu

Lecture Website:

<http://famousphil.com/09web>

Got a quick question?

I'm outside NSC225 WF from 3:10-3:50pm

(Check my calendar/email me to verify, I occasionally cancel)

Recap

- In the Last Lecture, We covered
 - Photoshop CS4 and creating HTML from PS
 - more advanced CSS/HTML Tricks of the trade
 - Mostly site design
- This lecture
 - Verify that the website validates
 - PHP development introduction
 - Security aspects
 - A simple website form using PHP
- Next Lecture
 - Will show how to get a page embedded in another page
 - Install a really simple text document manager that is written in PHP
 - will focus on installing and integrating Wordpress into an existing website
 - API Integration
 - Should take about 30 minutes to get through that material
- Lecture 6 will go into developing the gallery at <http://hsm3.famousphil.com>
 - PHP Data Structures
 - Other PHP knowledge (ternary operator, sessions, cookies, helpful design strategies)
 - MySQL queries / security knowledge!
 - Recommended prereqs that will help you understand this lecture fully: CSE116, 250 (coding and data structure experience)

PHP – Pre Hypertext Processor

- widely-used
- Open Source
- general-purpose scripting language
- Well suited for Web development
 - Easily embeddable into HTML
- Server Side Language
 - Unlike JavaScript, PHP generally won't output errors without entirely refreshing the page

PHP Advantages

- The client (user) will never see the PHP code
 - PHP will output the HTML required to display the page
 - Hard to spot vulnerabilities in an application unlike JavaScript
- Possible to write normal HTML and insert PHP in required positions
- Can have several PHP tags in the same document (not nested)
- Can call other PHP parts in another document (includes)
- Great documentation : <http://php.net>

PHP Disadvantages

- Can become tricky to write for novice users
- Easy to write programs with huge security flaws
- Requires a client side language for anything more fancy than loading pages for the clients
 - Interactive confirmation prompts

A Simple PHP Program

```
<? echo "HELLO WORLD!"; ?>
```

- <? Tells the server to begin PHP interpretation here
- ?> ends the PHP interpreter
- echo and print will print out whatever is between the quotes
- All statements in PHP are terminated with a ;

Getting the PHP code to work

- Apache will only pass files with the extension php to the PHP interpreter.
 - If the file ends in htm or html, `<??>` will be printed out exactly as it appears in the file
 - I typically use PHP includes (.inc files) which PHP will also interpret
- This works provided Apache is configured to work with PHP on the server

A bit more about PHP

- PHP is an interpreted language
 - It takes a series of statements
 - Apache will see the `<??>` and pass whatever is between these to the PHP interpreter
 - Each statement is executed sequentially
 - It will stop interpreting on errors
- Compiled languages like Java are different.
 - The program is verified prior to running it
 - Error checking prior to run

Creating your first PHP Script

- Make a new file named info.php on the server
 - Lets do this now in Vim
- Put the following in it and load the page:

```
<?php  
    phpinfo();  
?>
```

What the heck is PHPInfo()?

- creates a web page
 - Validates against transitional XHTML
 - Contains technical information
 - What is installed on the server
 - Very useful for determining if a certain function will work on the server
- Great way to test if PHP is working
 - File permissions can get off.
 - Normally `chmod 755 filename` is what you want your permissions set to

PHP Statements

- Normal text is split into sentences, PHP scripts are split into statements
- A PHP script can contain multiple statements
- Each statement tells the interpreter to do something
- Each statement is ended by a semicolon;

PHP Functions

- `Phpinfo()` is a function
- Functions are one of the most fundamental concepts in computer programming.
- A function takes in parameters and does something with them, then returns something
- `phpinfo()` has no parameters

PHP in HTML

- Take the template file, rename it to something like template.php, then add the following between the body tags:

```
<div>  
<?php  
print("\Hello, world!\");  
?>  
</div>
```
- You should see Hello, world! Outputted to the screen when it is ran in a browser from the server
- We will discuss the \ in a few slides

Comment on last slide

- Alternatively, you could have written:

```
<?php  
print "<div>";  
print "\"Hello, world!\"";  
print "</div>";  
?>
```

Single vs. Double Quotes

- You can use both interchangeably.
- Sometimes, it will be easier to use one over the other. This is especially true when you want to avoid using the escape character.

What the heck is this \" ?

- The \ character is the escape character for PHP.
- Since echo and other commands in PHP are surrounded by quotes, the PHP interpreter may terminate the statement prematurely when you meant to echo a “
- Simply add a \ character before the “ and it will escape the “ in the code and print out a “ to the screen

Good Coding Techniques

- Write each statement on a new line.
- Properly indent your code
 - If you have a loop, tab in for everything that runs within that loop
- Comment your code!
 - `//` the rest of the line is a comment
 - `#` the rest of a line is a comment
 - `/*` this is a comment
That can span over multiple lines `*/`

Variables

```
<?php
$greet="Hello, world!";
print "<div>$greeting</div>";//dangerous – explained
  later
?>
```

- Here \$greeting is a variable that is set to the string: "Hello, world!".
- All PHP variables begin with a \$

Dangers with variables in Strings

Concatenation

```
<?php
$greet="Hello, world!";
print "<div>".$greeting."</div>";
?>
```

- PHP may not recognize you have a variable within a string. Although it isn't required, I generally put a "." around the variable if I am within another string
- This concatenates (combines) the strings together.
- Technically, functions can also be put between the "." also

More about variables

- A variable must start with either a letter or an underscore. They can contain letters, digits, and underscores.
- Variables are case sensitive
- `$this` is reserved (Used in Object Oriented PHP to refer to the class we are in)
- Variables should have meaningful names!

PHP Strings, Numbers, Booleans

- A string is a set of characters in PHP
 - `$string = "I am a string";`
- Numbers in PHP don't require quotes
 - `$number = 3;`
- Boolean has a value of true or false

Increment / Decrease a number

- ++ is an operator that adds one.
 - Example:
 - \$b = 1;
 - \$b++;
 - echo \$b; //will print 2
- -- will remove one

isset()

- `isset($variable)` will return true if that variable was set to a value
- `isset($variable)` can be used to find out if the variable *variable* was set before using it.
 - When getting information from a PHP form, this is very useful in validating if the code is correct
 - Prevents against errors going into the Apache error log

Comparisons

`$truth == true ; // checks for equality`

`$truth = false ; // this sets $truth to be false`

- Other comparisons

`<` smaller than

`<=` smaller or equal than

`>` larger than

`>=` larger or equal than

Logical Operators

- `if($brand=="Coke" or $brand=="pepsi")`
 - This will run if either is true
- `if($brand=="Coke" and $brand=="pepsi")`
 - This will run if both are true
- Or can also be `||`
- And can also be `&&`

If statements

- `if(condition)` evaluates the expression inside, if its true, it runs that expression, else it runs what is in the else part

```
if($drunk) {  
    print "Don't drive!\n";  
}else{  
    print "It is ok to drive!\n";  
}
```

Loops

```
While (condition){  
    //run until the condition is no longer true  
}
```

```
for ($i=0; $i<10; $i++){  
    echo $i."\n"; // \n is a new line character  
    //counts from 0-9  
}
```

A very simple Form

- The HTML will have something like

```
<form action="hello.php" method="get"><p>  
  your name: <input type="text" name="lastname" />  
</p></form>
```

- PHP file hello.php will then be loaded:

```
<?php  
  print "Hello ";  
  print $_GET['lastname'];//form method is get  
?>
```

Checking for submission

Include a hidden element

```
<input type="hidden" name="submitted" value="1"/>
```

Have the script check for submission

```
if(isset($_GET['submitted'])) {  
    if ($_GET['submitted'] == "1"){//never a bad idea to do this twice  
        //it will avoid errors from popping up in a log somewhere  
        // work on the data that was submitted  
    }  
}  
else {  
    // print form  
}
```

Use `$_REQUEST`

- PHP programmers will typically use this because it is safer and more secure.

Getting the current PHP page

```
$_SERVER[PHP_SELF]
```

- Be careful when using this
 - I've never been able to answer why, but many experienced programmers will say this opens up many vulnerabilities

The Switch in PHP – an elegant if/else

- Switch (\$string)
 - Case “a string”: //(if \$string == “a string”)
 - Do this
 - Break;//exit the switch
 - Case “another String”
 - Do this
 - Break;
 - Default:
 - //nothing else matched, do this (else)

Stripslashes function

- This will remove any whitespaces or escape characters that were inserted into the `$_REQUEST` string automatically

Md5 function

- This will return the md5 encrypted string of whatever was passed to it
- Passwords should never be stored as an md5 string because of the rainbow tables
 - Basically if a hacker gets the hash, they can do a reverse lookup which has 25 million phrases cached or so (it's a lot)
 - Typically, programmers add a “salt” which makes this harder to reverse
 - Only way to crack an MD5 is by going forward and guessing right!

Regular Expressions

- Necessary evil in any programming language
- I use the eregi function to validate emails
 - Case insensitive regular expression match
- I won't go much further into this as these take me forever to get correct!
 - John Ciacia wrote a few excellent tutorials
 - <http://forum.codecall.net/php-tutorials/1751-php-tutorial-email-verification.html>
 - <http://forum.codecall.net/php-tutorials/9960-regular-expressions.html>

A few Demos

- Lets break the code up into include files
 - This will make editing the site much easier
 - Changing the entire site will mean changing a single header or footer file in the includes folder
- Lets get a contact form on the site!
 - This is more advanced, I will only go over what everything is doing (time permitting)