PHP and MySQL for Dynamic Web Sites

Intro

Ed Crowley

Class Preparation

- If you haven't already, download the sample scripts from:

http://www.larryullman.com/books/php-and-mysql-for-dynamic-web-sites-visual-quickpro-guide-4th-edition/#downloads

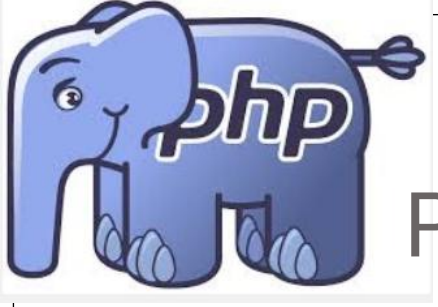
- Unzip sample scripts on local computer…
- Log into your hostgator account…

Dynamic Web Sites Overview

- In many cases, can be described as applications rather than sites.
  - Can respond to different parameters (time of day, version of the visitor's Web browser)
  - Have a "memory," allowing for user registration and login, e-commerce, and similar …
- Almost always integrate HTML forms, allowing visitors to perform searches, provide feedback…
  - Often have interfaces where administrators can manage site's content.
- Easier to maintain, upgrade, and build upon than static sites.
  - Don't always rely on a database, though many do.

PHP Well Suited for Web Development

- Now, means "PHP: Hypertext Preprocessor."
  - Originally stood for "Personal Home Page."
- PHP a "widely used general-purpose scripting language.
  - Can be embedded into HTML.
- PHP is a scripting as opposed to a compiled language.
- Designed to write Web scripts, not stand-alone applications (though that is possible).

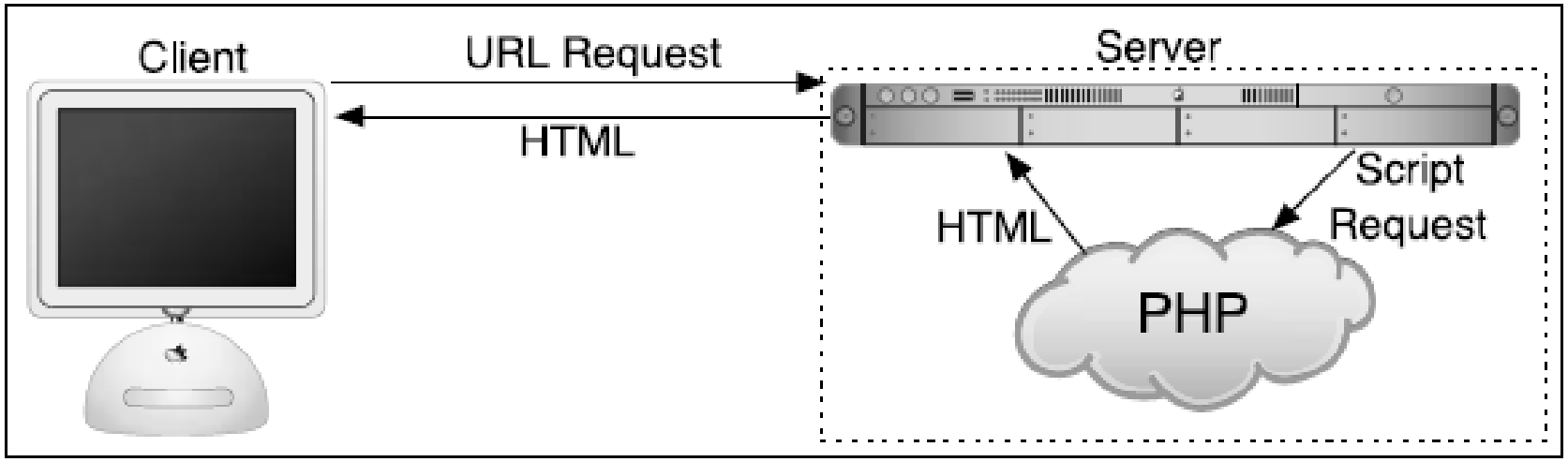


PHP Server-side and Cross-Platform

- PHP runs on many operating systems, including:
  - Windows
  - Unix (and its many variants)
  - Mac
- Tight integration with many databases
- Stable and portable
- Open source (no cost)
- Easier to learn than most alternatives.

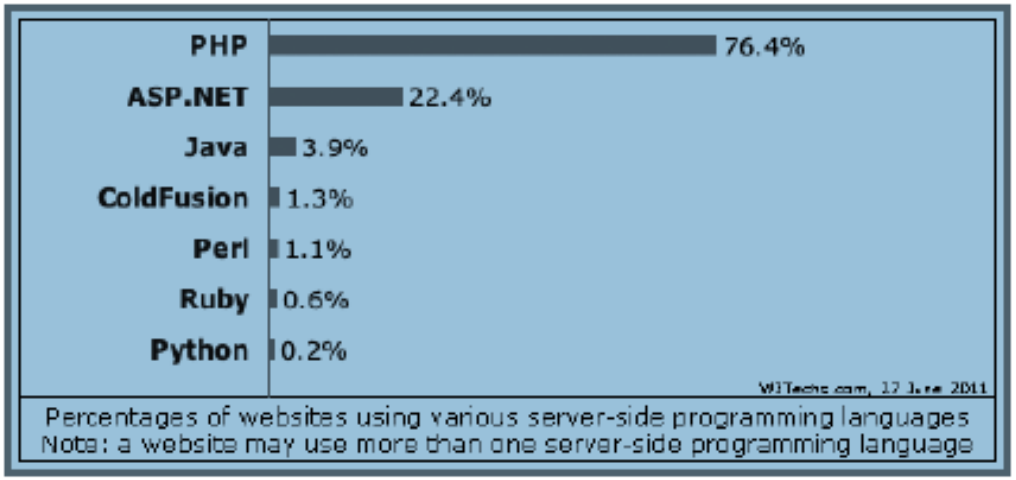
Server-Side

- PHP hosted on a server.
- Server sends Web pages to requesting visitors (you, the client, with your Web browser).
- When a visitor goes to a Web site written in PHP, the server reads the PHP code, then processes it.



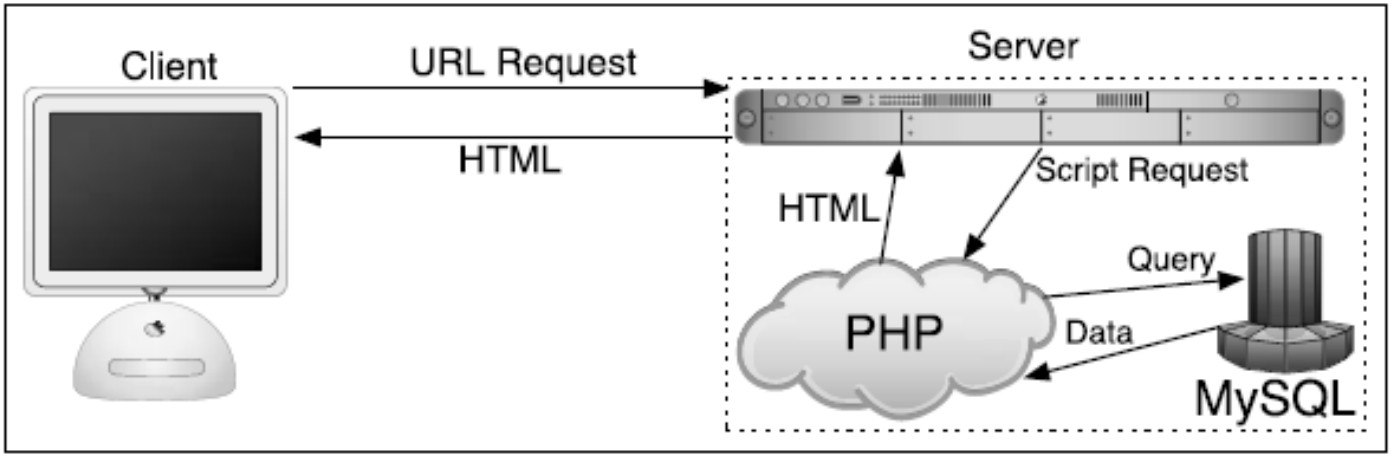
Server-side Technology of Choice.

- PHP code tells the server to send the appropriate data—HTML code—to the Web browser, which treats the received code as it would a standard HTML page.
- To the end user and the Web browser there is no perceptible difference between what home.html and home.php may look like
- But how that page's content was created will be significantly different.



MySQL world's most popular open-source database

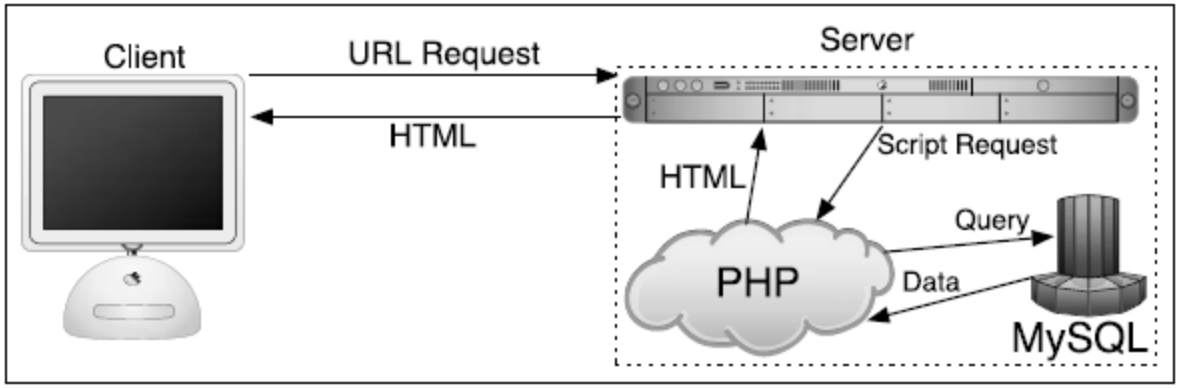
- By incorporating a database into a Web application, some of the data generated by PHP can be retrieved from MySQL .
- Further moves a site's content from static (hard-coded) to dynamic .



MySQL Open-Source

Consists of several components

- MySQL server (mysqld which runs and manages the databases)
- MySQL client (mysql, which gives you an interface to the server)
- Utilities, such as PHPMyAdmin



Understanding Encoding

- The encoding you use in a file dictates what characters can be represented.
- Some applications let you set the encoding in the preferences or options area; others set the encoding when you save the file.
- To indicate the encoding to the Web browser, there's the corresponding meta tag:

```
<!DOCTYPE html>
<meta charset="UTF-8">
```

Note: *New shorter HTML 5 encoding meta tag*.

- charset=utf-8 says that UTF-8 encoding is being used
  - 8-bit Unicode Transformation Format.

Chapter One Goals

At the end of this unit, you will be able to:

- Create a basic PHP script
- Execute a PHP script
- Send data to a Web browser
- Write comments in PHP
- Demonstrate how to use variables
- Work with string variables, including concatenation and a few string functions

Goals Two

At the end of this unit, you will be able to:

- Work with numeric variables, including arithmetic and formatting
- Work with constants
- Know how PHP treats the two quotation mark types differently
- Recognize common escape sequences
- Implement some basic debugging techniques

PHP Tags <?php ... ?>

- Anything written within these tags will be treated by the Web server as PHP, meaning the PHP interpreter will process the code.
- Any text outside of the PHP tags is immediately sent to the Web browser as regular HTML.
  - Because PHP is most often used to create content displayed in the Web browser, the PHP tags are normally put somewhere within the page's body.
- Lets make a php page…

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

.php File Type

- PHP files must have a proper extension.
- The extension tells the server to treat the script as a PHP page.
- Use .html for standard HTML pages and .php for PHP files.

PHP Attributes

- PHP has built-in functions to send data to the Web browser.
  - Most common: echo and print.
- Single or double quotation marks can be used.
  - There is a distinction between the two.
  - First quotation mark after the function name indicates the start of the message to be printed.
  - Next matching quotation mark (i.e., the next quotation mark of the same kind as the opening mark) indicates the end of the message to be printed.
- All PHP statements must end with a semicolon.

PHP Scripts

- PHP is case-insensitive when it comes to function names, so ECHO, echo, eCHo, and so forth will all work.

- Scripts 1.2 and 1.3
- Standard HTML and PHP

Understanding White Space

- With PHP, you send data (like HTML tags and text) to the Web browser, which, in turn, renders that data as a Web page.
- Often with PHP, you create the HTML source of a Web page.
- Three areas of notable white space (extra spaces, tabs, and blank lines) in:
  - your PHP scripts
  - your HTML source
  - the rendered Web page.

PHP Generally White Space Insensitive

- To make your scripts more legible, you can space out your code anyway you want.
  - The only white space in HTML that affects the rendered page is a single space (multiple spaces get rendered as one).
- To alter the spacing in a rendered Web page, use HTML tags
 or .
- To alter the spacing of the HTML source created with PHP, you can
  - Use echo or print over the course of several lines.

  or
  - Print the newline character (\n) within double quotation marks (equivalent to Enter or Return).

Comments

- PHP comments aren't sent to the Web browser at all.
  - Won't be viewable to the end user, even when looking at the HTML source.
- PHP supports three comment syntaxes.
- Use pound symbol (#):

```
# This is a comment.
```

- Use two slashes:

```
// This is also a comment.
```

- Both of these cause PHP to ignore everything that follows until the end of the line.

Comments

- Can be used to place a comment on the same line as some PHP code:

```
print 'Hello!'; // Say hello.
```

- A third style allows comments to run over multiple lines:

```
/* This is a longer comment

that spans two lines. */
```

- Script 1.4 comment.php …

Variables

- Containers used to temporarily store values.
  - Values can be numbers, text, or much more complex data.

PHP supports eight types of variables.

1. Boolean (TRUE or FALSE)
2. integer
3. floating point (decimals)
4. strings (characters);
5. arrays
6. objects
7. resources (which you'll see when interacting with databases)
8. NULL (which is a special type that has no value).

Variable's Name Must Start with a Dollar Sign ($)

- Variable's name can contain a combination of letters, numbers, and underscore, for example:

```
$my_report1
```

- First character after the dollar sign must be either a letter or an underscore (cannot be a number).
- Variable names in PHP are case-sensitive! … means that $name and $Name are different.
- To begin working with variables, this next script will print out the value of three predefined variables.
- Whereas a standard variable is assigned a value during the execution of a script, a predefined variable will already have a value when the script begins its execution.
- Most of these predefined variables reflect properties of the server as a whole, such as the operating system in use.

Variables

- First, variables can be assigned values using the equals sign (=), also called the assignment operator.
- Second, to display the value of a variable, you can print the variable without quotation marks:

```
print $some_var;
```

- Or variables can be printed within double quotation marks:

```
print "Hello, $name";
```

Variables and Quotation Marks

- You cannot print variables within single quotation marks:

Script 1.5 predefined.php

- A string is merely a quoted chunk of characters: letters, numbers, spaces, punctuation, and so forth.

These are all strings:

- 'Tobias'
- "In watermelon sugar"
- '100'
- 'August 2, 2011'
- To make a string variable, assign a string value to a valid variable name:

```
$first_name = 'Tobias';
$today = 'August 2, 2011';
```

Strings

- When creating strings, you can use either single or double quotation marks to encapsulate the characters, just as you would when printing text.
- Likewise, you must use the same type of quotation mark for the beginning and the end of the string.
- If that same mark appears within the string, it must be escaped:
- $var = "Define \"platitude\", please.";
- Or you can also use the other quotation mark type:
- $var = 'Define "platitude", please.';

Script 1.6

- To print out the value of a string, use either echo or print:

```
echo $first_name;
```

- To print the value of string within a context, you must use double quotation marks:

```
echo "Hello, $first_name";
```

- You've already worked with strings once—when using the predefined variables in the preceding section.
- In Script 1.6, string variables are created and their values are sent to the Web browser.

Concatenating Strings

- Concatenation is like addition for strings.
- Performed using the concatenation operator, which is the period (.)

```
$city= 'Seattle';
$state = 'Washington';
$address = $city . $state;
```

- The $address variable now has the value Seattle Washington, which almost achieves the desired result (Seattle, Washington). To improve upon this, you could write

```
$address = $city . ' ,  ' . $state;
```

- so that a comma and a space are concatenated to the variables as well.

Concatenation

- Because of how liberally PHP treats variables, concatenation is possible with strings and numbers.
- Either of these statements will produce the same result (Seattle, Washington 98101):

```
$address = $city . ' , ' . $state .,' 98101';
$address = $city . ' ,  ' . $state .,' ' . 98101;
```

- Let's modify strings.php to use this new operator.

- Script 1.7 Concat.php

PHP Manual

- Lists every PHP function and feature. Accessible online at:

http://php.net/manual/en/index.php

- Organized with general concepts (installation, syntax, variables) first, ends with the functions by topic (MySQL, string functions, and so on).

For each function, the manual indicates:

- Versions of PHP the function is available.
- How many and what types of arguments the function takes (optional arguments are wrapped in square brackets).
- What type of value the function returns.
- Manual also contains a description of the function.
- Critically important that you know what version of PHP you're running, as functions and other particulars of PHP do change over time.

Number Types

- PHP has both integer and floating-point (decimal) number types.
  - In my experience, though, these two types can be classified under the generic title numbers without losing any valuable distinction (for the most part).
- Valid number-type variables in PHP can be anything like

```
3.14
10980843985
4.2398508
```

Arithmetic Operators

- Along with the standard arithmetic operators you can use on numbers (Table 1.1), there are dozens of functions built into PHP

TABLE 1.1 Arithmetic Operators

Operator	Meaning
+	Addition
-	Subtraction
*	Multiplication
/	Division
%	Modulus
++	Increment
--	Decrement

Number Formats

- Two common ones are round( ) and number_format( ).
- The former rounds a decimal to the nearest integer:

```
$n = 3.14;
$n = round ($n); // 3
```

- It can also round to a specified number of decimal places:

```
$n = 3.142857;
$n = round ($n, 3); // 3.143
```

- The number_format( ) function turns a number into the more commonly written version, grouped into thousands using commas:

```
$n = 20943;
$n = number_format ($n); // 20,943
```

Number Format

- This function can also set a specified number of decimal points:

```
$n = 20943;
$n = number_format ($n, 2); //
20,943.00
```

- To practice with numbers, let's write a mock-up script that performs the calculations one might use in an e-commerce shopping cart.
- Script 1.8 numbers.php
- The numbers.php script performs basic mathematical calculations, like those used in an e-commerce application.

Constants

- Constants, like variables, are used to temporarily store a value, but constants and variables differ in many ways.
- For starters, to create a constant, you use the define( ) function instead of the assignment operator (=):

```
define ('NAME' value);
```

- Notice that, as a rule of thumb, constants are named using all capitals, although this is not required.
- Constants do not use the initial dollar sign as variables do (because constants are not variables).
- A constant can only be assigned a scalar value, like a string or a number:

```
define ('USERNAME' 'troutocity');
define ('PI' 3.14);
```

- Unlike variables, a constant's value cannot be changed.

Constants

- To access a constant's value, like when you want to print it, you cannot put the constant within quotation marks:

```
echo "Hello, USERNAME"; // Won't work!
```

- With that code, PHP literally prints Hello, USERNAME A and not the value of the USERNAME constant (because there's no indication that USERNAME is anything other than literal text).
- Instead, either print the constant by itself:

```
echo 'Hello, ';
echo USERNAME;
```

- or use the concatenation operator:

```
echo 'Hello, ' . USERNAME;
```

Predefined Constants

- PHP runs with several predefined constants, much like the predefined variables used earlier in the chapter.
- These include PHP_VERSION (the version of PHP running) and PHP_OS (the operating system of the server).
- This next script will print those two values, along with the value of a user-defined constant.
- Script 1.9 Constant.php

Quotation Marks

- In PHP it's important to understand how single quotation marks differ from double quotation marks.
- With echo and print, or when assigning values to strings, you can use either.
- But there is a key difference between the two types of quotation marks and when you should use which.
- In PHP, values enclosed within single quotation marks will be treated literally, whereas those within double quotation marks will be interpreted.

Double Quotes

TABLE 1.2 Escape Sequences

Code	Meaning
\"	Double quotation mark
\'	Single quotation mark
\\	Backslash
\n	Newline
\r	Carriage return
\t	Tab
\$	Dollar sign

- In other words, placing variables and special characters (Table 1.2) within double quotes will result in their represented values printed, not their literal values.

Special Characters

- For example, assume that you have

```
$var = 'test';
```

- The code echo "var is equal to $var";
- Will print out var is equal to test, but the code echo 'var is equal to $var';
- Will print out var is equal to $var.
- Using an escaped dollar sign, the code echo "\$var is equal to $var"; will print out $var is equal to test, whereas the code echo '\$var is equal to $var'; will print out \$var is equal to $var A.

Double Quotes

- As these examples should illustrate, double quotation marks will replace a variable's name ($var) with its value (test) and a special character's code (\$) with its represented value ($).
- Single quotes will always display exactly what you type, except for the escaped single quote (\') and the escaped backslash (\\), which are printed as a single quotation mark and a single backslash, respectively.
- As another example of how the two quotation marks differ, let's modify the numbers.php script as an experiment.
- Script 1.10 This, the final script in the chapter, demonstrates the differences between using single and double quotation marks.

Questions???