Review: PHP/MySQL Tutorial

Basic and Advanced principles

Goal of this tutorial

- Not to teach everything about PHP, but provide the basic knowledge
- Explain code of examples
- Provide some useful references

What is PHP?

- PHP == 'Hypertext Preprocessor'
- Open-source, server-side scripting language
- Used to generate dynamic web-pages
- PHP scripts reside between reserved PHP tags
  - This allows the programmer to embed PHP scripts within HTML pages

What is PHP (cont'd)

- Interpreted language, scripts are parsed at run-time rather than compiled beforehand
- Executed on the server-side
- Source-code not visible by client
  - 'View Source' in browsers does not display the PHP code
- Various built-in functions allow for fast development
- Compatible with many popular databases

WHAT DOES PHP CODE LOOK LIKE?

- Structurally similar to C/C++
- Supports procedural and object-oriented paradigm (to some degree)
- All PHP statements end with a semi-colon
- Each PHP script must be enclosed in the reserved PHP tag

```
<?php
     …
?>
```

Comments in PHP

- Standard C, C++, and shell comment symbols

```
// C++ and Java-style comment

# Shell-style comments

/* C-style comments
     These can span multiple lines */
```

Variables in PHP

- PHP variables must begin with a “$” sign
- Case-sensitive ($Foo != $foo != $fOo)
- Global and locally-scoped variables
  - Global variables can be used anywhere
  - Local variables restricted to a function or class
- Certain variable names reserved by PHP
  - Form variables ($_POST, $_GET)
  - Server variables ($_SERVER)
  - Etc.

Variable usage

```
<?php
$foo = 25;              // Numerical variable
$bar = “Hello”;         // String variable

$foo = ($foo * 7);      // Multiplies foo by 7
$bar = ($bar * 7);      // Invalid expression
?>
```

ECHO

- The PHP command '**echo**' is used to output the parameters passed to it
  - The typical usage for this is to send data to the client's web-browser
- Syntax
  - void **echo** (string arg***1*** [, string arg***n***...])
  - In practice, arguments are not passed in parentheses since **echo** is a language construct rather than an actual function

A "void" type indicates an *absence* of information — a function is said to have a return type of "void" if it does not return *any* value.

ECHO EXAMPLE

```
<?php
$foo = 25;              // Numerical variable
$bar = "Hello";         // String variable

echo $bar;              // Outputs Hello
echo $foo,$bar;         // Outputs 25Hello
echo "5x5=",$foo;       // Outputs 5x5=25
echo "5x5=$foo";        // Outputs 5x5=25
echo '5x5=$foo';        // Outputs 5x5=$foo
?>
```

- **Notice** how echo '5x5=$foo' outputs $foo rather than replacing it with 25
- Strings in single quotes (' ') are not interpreted or evaluated by PHP
- This is true for both variables and character escape-sequences (such as "\n" or "\\")

Arithmetic Operations

```
<?php
        $a=15;
        $b=30;
        $total=$a+$b;
        Print $total;
        Print "<p><h1>$total</h1>";
        // total is 45
?>
```

- $a - $b   // subtraction
- $a * $b   // multiplication
- $a / $b   // division
- $a += 5   // $a = $a+5 Also works for *= and /=

Concatenation

- Use a period to join strings into one.

```
<?php
$string1="Hello";
$string2="PHP";
$string3=$string1 . " " . $string2;
Print $string3;
?>
```

```
Hello PHP
```

Escaping the Character

- If the string has a set of double quotation marks that must remain visible, use the \ [backslash] before the quotation marks to ignore and display them.

```
<?php
$heading="\"Computer Science\"";
Print $heading;
?>
```

```
"Computer Science"
```

PHP Control Structures

- Control Structures: Are the structures within a language that allow us to control the flow of execution through a program or script.
- Grouped into conditional (branching) structures (e.g. if/else) and repetition structures (e.g. while loops).
- Example if/else if/else statement:

```
if ($foo == 0) {
            echo 'The variable foo is equal to 0';
}
else if (($foo > 0) && ($foo <= 5)) {
             echo 'The variable foo is between 1 and 5';
}
else {
             echo 'The variable foo is equal to '.$foo;
}
```

```
if ($foo == 0)
 {
            echo ‘The variable foo is equal to 0’;
}

else if (($foo > 0) && ($foo <= 5))
{
             echo ‘The variable foo is between 1 and 5’;
}
Else

 {
             echo ‘The variable foo is equal to ‘.$foo;
}
```

If … Else…

- If (condition)

```
{
        Statements;
}
Else
{
        Statement;
}
```

```
<?php
If($user=="John")
{
        Print "Hello John.";
}
Else
{
        Print "You are not John.";
}
?>
```

No THEN in PHP

While Loops

- While (condition)

  {

  Statements;

  }

```
<?php
$count=0;
While($count<3)
{
        Print "hello PHP. ";
        $count += 1;
        // $count = $count + 1;
        // or
        // $count++;}
?>
```

```
hello PHP. hello PHP. hello PHP.
```

The PHP do...while Loop: Syntax

```
DO
  {
  CODE TO BE EXECUTED;
  }
WHILE (CONDITION IS TRUE);
```

```
<?php
   $x=1;
   do
     {
     echo "The number is: $x <br>";
     $x++;
     }
   while ($x<=5)
   ?>
```

FOR STATEMENT

For loop

```
for($i=0;$i < 10;$i++) {
    echo(“the value is :”. $i);
  }
```

- Alternative Syntax

```
for($i=0;$i < 10;$i++)
// html code goes here
```

Functions

- Functions MUST be defined before then can be called
- Function headers are of the format

```
function functionName($arg_1, $arg_2, …, $arg_n)
```

  - Note that no return type is specified
- Unlike variables, function names are not case sensitive (foo(…) == Foo(…) == FoO(…))

Functions example

```
<?php
      // This is a function
      function foo($arg_1, $arg_2)
       {
         $arg_2 = $arg_1 * $arg_2;
         return $arg_2;
      }

      $result_1 = foo(12, 3);           // Store the function
      echo $result_1;                   // Outputs 36
      echo foo(12, 3);                  // Outputs 36
?>
```