CSCI 2910
Client/Server-Side Programming

Topic: Intro to PHP

Reading: Chapters 1 and 2

Today's Goals

Today's lecture will cover:

- Differences between server-side and client-side operation
- Format of a PHP file
- Syntax of PHP code and similarities between PHP code and JavaScript code
- Data types

Web Scripting with PHP

PHP has many benefits:

- Open Source
- Flexible for integration with HTML – HTML easily supports the use of the PHP scripting tags
- Suited to complex projects – database support, vast libraries, and the power of the server allow PHP to satisfy very complex programming needs.
- Fast at running scripts – Even though PHP is a scripting language, the architecture of the scripting engine provides for fast execution.
- Platform and O/S portable – PHP runs on a variety of different platforms and operating systems

General Format of a PHP File

- A block of PHP script is embedded within an HTML file using the <?php and ?> tags.
- Can also use <script language="PHP"> and </script>
- The PHP script engine does not like the tag <?xml version="1.0" encoding="ISO-8859-1"?>
  - Engine is interpreting <? ... ?> as executable script
  - Remove them from your XML template to create a PHP template.
  - Could use PHP script to generate <?xml> tag

General Format of a PHP File (continued)

- Just like JavaScript, whitespace is ignored.
- Just like JavaScript, end lines with semicolon (;).
- Unlike JavaScript, PHP code is executed at server and ***replaced*** with resulting output.
- The file must have the extension ".php". Server needs this in order to know to run the file through the PHP script engine before sending output to client.

PHP "Hello, World!"

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">

<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">

<head>
<title>Simple XHTML Document</title>
</head>
<body>
<h1>
<?php
    print "Hello, World!";
?>
</h1>
</body>
</html>
```

JavaScript "Hello, World!"

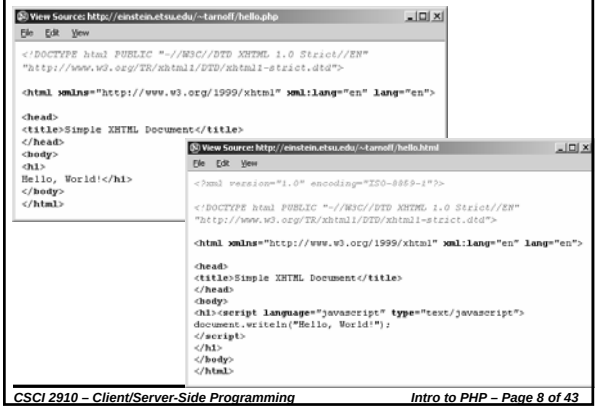
```
<?xml version="1.0" encoding="ISO-8859-1"?>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">

<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">

<head>
<title>Simple XHTML Document</title>
</head>
<body>
<h1><script language="javascript" type="text/javascript">
document.writeln("Hello, World!");
</script>
</h1>
</body>
</html>
```


The Difference – "View Source"



PHP Comments

- As in JavaScript, there are two methods for inserting comments into your code. They are:
  - Block comments
  - End of line comments
- We don't need to comment out code for browsers since code is executed on server.

Block Comments

```
/* This is a block comment. It
is surrounded by the
slash/asterisk and
asterisk/slash that indicate the
beginning and ending of a
comment. A block comment may
span multiple lines. */
```


End of Line Comments

```
// This is an end of line comment.
// Everything from the double
// slashes to the end of the line
// is ignored.
// To use this method over
// multiple lines, each line must
// have its own set of double
// slashes.

# This is also an end of line
# comment
```


Outputting Results

- Just like JavaScript, there are a number of ways to output results that are to become part of the HTML file.
- The earlier example uses the print command to output a string.
- print can also be used to output values or variables.
- The following slide presents examples of valid print statements.

print Examples

```
print "Hello, World";

print 123;  // Outputs "123"

$outputString = "Hello, World!";
print $outputString;
```


echo Statement

- The echo statement similar to print
- echo, however, can take on a sequence of arguments separated by commas.
- Example:

```
$outputString = "The answer is ";
echo $outputString, 42, "!";
```

- This outputs "The answer is 42!"
- print cannot combine elements like this.

Escape Characters

- Because of the number of reserved characters in PHP, escaping is necessary.
- Characters are escaped by preceding them with a backslash (\).
- Characters that need escaped include ', ", \, $, and ?.
- Whitespace including carriage returns are allowed as part of a string, but they are then output as part of the string. Of course, in HTML, carriage returns are considered whitespace and are ignored.
- As in JavaScript, single quotes can be used without escaping within double quoted strings and vice versa.

Printing Characters Not Available on Keyboard

- Escaping can also be used to display ISO-8859-1 characters that are not present on the keyboard.
- This is done by taking the ISO-8859-1 hex value and placing it after "\x".
- The ISO-8859-1 hex values can be found using the Character Map application found in Accessories → System Tools.
- For example, the character "¼" has the hexadecimal ISO-8859-1 value bc_{16}. This can be represented with \xbc.
- print "\xbc tsp" prints the string "¼ tsp"

In-Class Exercise

- Earlier it was stated that the PHP script engine does not like the tag <?xml version="1.0" encoding="ISO-8859-1"?>.
- How might we still incorporate this tag in the file we send to the browser without causing problems for the PHP engine?

Variable Declarations

- PHP interprets the dollar sign ($) followed by a letter or an underscore (_) to be a variable name.
- Variables do not need to be declared before you use them.
- Example: $var1 = 25;
- To help set off a variable identifier within a string, you can surround it with curly brackets.
- This will become helpful when we start discussing arrays and objects.
- Example: echo "The value is {$var1}." will display "The value is 25."

Data Types

- Scalar types
  - boolean
  - float
  - integer
  - string
- Compound types
  - array
  - object

Using Scalar Types

- A boolean variable can be assigned only values of ***true*** or ***false***.

```
$answer = false;
$finished = true;
```

- An integer is a whole number (no decimal point)

```
$age = 31;
```


Using Scalar Types (continued)

- A float has a decimal point and may or may not have an exponent

```
$price = 12.34;
$avog_num = 6.02e23; //6.02x10^23
```

- A string is identified as a sequence of characters

```
$name = "John Smith";
```

- Strings can be concatenated using a dot (.)

```
$name = "John" . " Smith";
```


Constants

- Constants associate a name with a scalar value.
- Constants are defined using the function ***define()***.

```
define("PI", 3.141593);
```

- There are a number of predefined constants. These include:
  - M_E = 2.718281828459
  - M_PI = 3.1415926535898
  - M_2_SQRTPI = 1.1283791670955 (Square root of pi)
  - M_1_PI = 0.31830988618379 (Square root of 1/pi)
  - M_SQRT2 = 1.4142135623731 (Square root of 2)
  - M_SQRT1_2 = 0.70710678118655 (Square root of ½)

Arithmetic Operators

Operator	Operation	Example	Result
+	Addition	`$y = 2 + 2;`	$y will contain 4
–	Subtraction	`$y = 3;` `$y = $y – 1;`	$y will contain 2
/	Division	`$y = 14 / 2;`	$y will contain 7
*	Multiplication	`$z = 4;` `$y = $z * 4;`	$y will contain 16
%	Modulo	`$y = 14 % 3;`	$y will contain 2
++	Increment	`$y = 7;` `$y++;`	$y will contain 8
--	Decrement	`$y = 7;` `$y--;`	$y will contain 6

Bitwise Logical Operations

- ~ Bitwise NOT operator: Inverts each bit of the single operand placed to the right of the symbol
- & Bitwise AND: Takes the logical-bitwise AND of two values
- | Bitwise OR operator: Takes the logical-bitwise OR of two values
- ^ Bitwise XOR: Takes the logical-bitwise exclusive-OR of two values

Bitwise Shift Operations

- << Left shift: Shifts the left operand left by the number of places specified by the right operand filling in with zeros on the right side.
- >> Sign-propagating right shift: Shifts the left operand right by the number of places specified by the right operand filling in with the sign bit on the left side.
- >>> Zero-fill right shift operator: Shifts the left operand right by the number of places specified by the right operand filling in with zeros on the left side.

Flow Control

- As in JavaScript, flow control consists of a number of reserved words combined with syntax to allow the computer to decide which parts of code to execute, which to jump over, and which to execute multiple times.
- For the most part, the flow control that you learned for JavaScript is the same for PHP.

If-Statement

- The code below represents the syntax of a typical *if-statement*:

```
if ($grade > 93)
     print "Student's grade is A";
```

- If grade was 93 or below, the computer would simply skip this instruction.

If-Statement (continued)

- Just like JavaScript, multiple instructions may be grouped using curly brackets. For example:

```
if ($grade > 93)
{
   print "Student's grade is A";
   $honor_roll_value = true;
}
```


If-Statement (continued)

- As in JavaScript, the programmer can string together if-statements to allow the computer to select from one of a number of cases using *elseif* and *else*. (Note that JavaScript allows ***else if*** while PHP uses ***elseif***.)
- For example:

```
if ($grade > 93)
   print "Student's grade is an A";
elseif ($grade > 89)
   print Student's grade is an A-";
else
   print "Student did not get an A";
```


Comparison Operators

- > Returns true if the first value is greater than the second
- >= Returns true if the first value is greater than or equal to the second
- < Returns true the first value is less than the second
- <= Returns true if the first value is less than or equal to the second
- == Returns true if first value is equal to second
- != Returns true if first value is not equal to second

Logical Operators

- ! Returns true if its operand is zero or false
- && Returns false if either operand is zero or false
- || Returns false if both operands are zero or false

Switch-Statement

- The switch statement can be used as an alternative to the if, elseif, else method.

```
switch($menu)
{
    case 1:
          print "You picked one";
          break;
    case 2:
          print "You picked two";
          break;
    default:
          print "You did not pick one or two";
          break;
}
```


Switch-Statement (continued)

- Note that if a break is not encountered at the end of a case, the processor continues through to the next case.
- Example: If $var1=1, it will print both lines.

```
switch($var1)
{
    case 1:
          print "The value was 1";
    default:
          print "Pick another option";
          break;
}
```


While-loop

- PHP uses the *while-loop* just like JavaScript.
- Like the if-statement, this format also uses a condition placed between two parenthesis
- As long as the condition evaluates to true, the program continues to execute the code between the curly brackets in a round-robin fashion.

While-loop (continued)

- Format:

```
while(condition)
{
  statements to execute
}
```

- Example:

```
$count = 1;
while($count < 72)
{
  print "$count ";
  $count++;
}
```


do … while loop

- The do … while loop works the same as a while loop except that the condition is evaluated at the end of the loop rather than the beginning
- Example:

```
$count = 1;
do
{
   print "$count ";
   $count++;
}while($count < 72);
```

for-loop

- In the two previous cases, a counter was used to count our way through a loop.
- This task is much better suited to a for-loop.

```
for ($count = 1; $count < 72; $count++)
{
   print "$count ";
}
```

- A "break" can be used to break out of a loop earlier.

In-Class Exercise

Convert the JavaScript shown below to PHP?

```
<script language="javascript" type="text/javascript">
<!--
  value = 34.5;
  for(i = 0; i < 10; i++)
  {
    document.writeln("34.5/(2^" + i + ") is " + value);
    document.writeln("<br />");
    value = value/2;
  }
//-->
</script>
```


Type Conversion

- Different programming languages deal with variable types in different ways. Some are strict enforcing rules such as not allowing an integer value to be assigned to a float.
- The process of converting from one data type to another is called "casting".
- To convert from one type to another, place the type name in parenthesis in front of the variable to convert from.
- In some cases, there are functions that perform the type conversion too.

Some Examples of Type Conversion

- $ivar = (int) $var;
- $ivar = (integer) $var;
- $ivar = intval($var);
- $bvar = (bool) $var;
- $bvar = (boolean) $var;
- $fvar = (float) $var;
- $fvar = floatval($var);
- $svar = (string) $var;
- $svar = stringval($var);

Examples of Type Conversion (continued)

Value	Cast to int	Cast to bool	Cast to string	Cast to float
null	0	false	""	0
true	1	true	"1"	1
false	0	false	""	0
0	0	false	"0"	0
3.8	3	true	"3.8"	3.8
"0"	0	false	"0"	0
"10"	10	true	"10"	10
"6 feet"	6	true	"6 feet"	6
"foo"	0	true	"foo"	0

Type Conversion (continued)

- PHP can automatically convert types too.
- If a variable is used as if it were a different type, the PHP script engine assumes a type conversion is needed and does it for you.
- Examples:

```
$var = "100" + 15;   // var$ set to integer = 115
$var = "100" + 15.0; // var$ set to float = 115
$var = 15 + " bugs"; // var$ set to integer = 15
$var = 15 . " bugs"; // var$ set to string = "15 bugs"
```

In-Class Exercise

Identify the errors in the following PHP script.

```
<?php
strvar1 = "<h1 align="center">Integer
                    Squares from 0 to 9</h1>"
prints strvar
prints "<ul>"
for(i = 0; i < 9; i+)
{
   isquared = i * i
   prints "<li>Square root of " + i +
                " is " + isquared + "</li>"
}
prints "</ul>"
?>
```