PHP: Variables

Syntax

The syntax for PHP variables is similar to C and most other programming languages. There are three primary differences:

1. Variable names must be preceded by a dollar sign (`$`).
2. Variables do not need to be declared before being used.
3. Variables do not need to be explicitly typed (e.g., `int`, `float`, etc.).

Types

Numeric types can either be integers (e.g., 7, -35) or floating-point numbers (e.g., 4.23, -35.0).

Strings can be denoted by single (`'`) or double (`"`) quotation marks. The ability to use either type of quotation mark is useful if a string contains one of these marks. As with C you can use the `\'` and `\"` escape codes to explicitly produce the desired quotation mark. The single and double quotation mark delimiters for strings have the same effect as they do in Perl. Characters within single quotation marks are treated as literals, and characters within double quotation marks are interpolated. Interpolation means that values will be substituted for variable names, and escape codes will be replaced with the appropriate characters within the string.

```
<?php
  $greeting = 'Hello';
  $message1 = "$greeting, world!\n";  // Double quotation marks
  $message2 = '$greeting, world!\n';  // Single quotation marks
  print "1: $message1\n";
  print "2: $message2\n";
?>
```

This is how the output appears before being sent to the browser.

```
1: Hello, world!

2: $greeting, world!\n
```

Note that `$message2` neither contains the value for `$greeting` nor interprets the newline character as calling for a new line.

As in Perl, the concatenation character is the dot (`.`) operator:

```
<?php
  $a = "Hello, world" . 35 . "!\n";
  print $a;
?>
```

This is how the output appears before being sent to the browser.

```
Hello, world35!
```

There are two types of arrays in PHP: indexed and associative. An indexed array works just like an array in C. An associative array is similar to the hash data structure in Perl. Note that unlike Perl, you do not have to prefix the variable name with the at-sign (`@`) or percent-sign (`%`) character to indicate that the given variable is an array or hash respectively. For completeness, the other variable type is an object. Objects are defined by a structure (i.e., a class) of which instances (i.e., objects) are created. Objects will not be covered in this course. The syntax and usage for indexed and associative arrays will be discussed in a later document.

The `var_dump()` function in PHP is used to print the current type and values of one or more variables. Arrays and objects are printed recursively with values indented to show structure.

```
<?php
  $a = 35;
  $b = "Programming is fun!";
  $c = array(1, 1, 2, 3, 5, 8);
  var_dump($a,$b,$c);
?>
```

Here's the output from the above code.

```
int(35)
string(19) "Programming is fun!"
array(6) {
  [0]=>
  int(1)
  [1]=>
  int(1)
  [2]=>
  int(2)
  [3]=>
  int(3)
  [4]=>
  int(5)
  [5]=>
  int(8)
}
```

The variable `$a` is an integer with value `35`. The variable `$b` is a string that contains 19 characters. The variable `$c` is an array with six elements: element zero is an integer whose value is `1`, and so on.

PHP also supports type casting, using more or less the same syntax as C.

```
<?php
  $a = 100;
  $b = 324.75;
  $sum = $a + (int) $b;
  print "$a + $b = $sum";
?>
```

This is how the output may appear in the browser.

```
100 + 324.75 = 424
```

It's also convenient that in PHP, strings that contain valid numeric values can be used in arithmetic expressions.

```
<?php
  $a = "100";
  $b = "324.75";
  $sum = $a + $b;
  print "$a + $b = $sum";
?>
```

This is how the output may appear in the browser.

```
100 + 324.75 = 424.75
```

Common error messages

When working with small PHP scripts, the most common error messages occur because of parse errors, such as the missing semicolon in line 3 below:

```
1  <?php
2    $a = 35;
3    $b = "Programming is fun!"
4    print "$a\n$b\n";
5  ?>
```

This is how the output may appear in the browser.

Parse error: syntax error, unexpected T_PRINT in **/home/username/www-home/variables.php** on line **4**

Despite the output indicating an error on line 4, the error is on the third line: a missing semicolon. A parse error typically occurs because of a missing semicolon or unbalanced quotation marks. Interestingly, undefined variables can be treated as either having default values that depend on the context in which the variable is used or the interpreter may choose to print an error message. On the EECS/Claxon Web server, the interpreter chooses to assign default values to the variable: zero (`0`) if the variable is used in a numeric context, the empty string (`""`) if the value is used in a string context, and `false` if the variable is used in a boolean context. For example:

```
<?php
  $a = 35;
  $b = "Programming is fun!";
  $a = $a + $foo;
  print "$a\n";
  print $b . $foo . "\n";
  if (!$foo)
    print "foo has played me false\n";
?>
```

Here's the output of the above code:

```
35
Programming is fun!
foo has played me false
```