PHP: Quickly

Ira Ray Jenkins

PHP:

- stands for PHP Hypertext Preprocessor
- is open-sourced
- was a server-side scripting language
- has become a general purpose scripting language
- supports OO programming
- has tons of open-source libraries
- used by ~80% of websites that we know use server-side scripting[1]
- linked to ~30% of all vulnerabilities listed on National Vulnerability Database[2]

[1] HTTP://W3TECHS.COM/TECHNOLOGIES/OVERVIEW/PROGRAMMING_LANGUAGE/ALL
[2] HTTP://WWW.COELHO.NET/PHP_CVE.HTML

RUNNING PHP

- **Interactive interpreter:**
  - php –a
  - Leave interpreter with ‘exit’ or ‘ctrl-D’

- **Command Line Interface (CLI):**
  - php <file.php> <args>

- **Common Gateway Interface (CGI):**
  - http://file.php

DELIMITERS

- PHP only parses code within special delimiters
  - <?php … ?>
  - <script language="php"> … <script>
  - <?= … ?>
  - <? … ?>
  - <% … %>
- PHP is a free-form language
  - Whitespace doesn’t matter, like in C
  - Exception: whitespace after closing delimiter may break HTML
- Statements end with a semicolon ‘;’
- PHP supports C and shell-style comments
  - One-liners: // or #
  - Multi-lines: /*   */

HELLO WORLD

```
<?php
    // print "Hello World"
    print("Hello World");
?>
```

```
<?php
    echo ‘Hello World’;
?>
```

```
<?php print ‘Hello World’;?>
```

```
<%
    /* echo "Hello World" */
    echo("Hello World");
%>
```

- **print vs. echo**
- print returns a value

VARIABLES

- PHP is loosely typed, i.e., types are implicit and “context dependent”
  - boolean, integer, float, string, array, object, NULL
- Variables are created when they are assigned values
- Variables are signified via ‘$’
- Naming rules:
  - Alpha-numeric with underscores
  - Must begin with letter or underscore
  - Are case-sensitive, i.e., $x is not the same as $X

```
$foo = 10;                           // $foo is integer (10)
$foo = TRUE;                         // $foo is boolean (TRUE)
$foo = NULL;                         // $foo is NULL    (NULL)
$foo = 1 + "10 Small Pigs";          // $foo is integer (11)
$foo = 4 + "10.2 Little Piggies"; // $foo is float   (14.2)
```

STRINGS

- A sequence of characters, i.e., a byte array
- Enclosed by single (‘) or double (“) quotes
- Anything between single quotes will not be parsed, except \’ and \\
- Anything between double quotes will be parsed, including escape characters
- Interpretted as numbers by the following rules:
  - If it starts with a number, use it
  - If it has ‘.’, ‘e’, or ‘E’ then it is a float
  - Default to zero (0)
- Concatenate with ‘.’
- Lots of built-in functions, like strlen(), etc…
- Be careful about international character encoding!
  - Many functions support a multi-byte format, mb_strlen()

SCOPING

- Local
  - Variables declared inside a function
  - Deleted when function completes
- Global
  - Variables declared outside any function
  - Not accessible from within a function without “global” keyword
  - All globals are stored in associative array: $GLOBALS[“x”]
- Static
  - Like C static, maintains function local variables between calls
- Parameters/Arguments
  - Local variables whose values are passed into functions

OPERATORS

- Arithmetic
  - Add (+), subtract (-), multiply (*), divide (/), modulus (%), and concatenate (.)
- Assignment
  - (=) and (+=) etc…
- Post- and pre-increment and decrement (++/- -)
- Logical
  - and/&&, or/||, !, xor
- Comparison
  - <, >, <=, >=, ==
  - <>, !=
  - === and !== (identical; equal to and same type or same key/value pairs)

CONTROL STRUCTURES

- if, if… else, and if… else if… else statements
- switch statements
  - break and continue do the same thing
  - continue 2 does what you think continue does
  - default case
  - does “loose” comparisons
- while, do-while, for, and foreach loops

```
for ($i=1; $i<=5; $i++)
{
  // code to be executed
}
```

```
foreach ($array as $value)
{
  // code to be executed
}
```

HTTP://WWW.PHP.NET/MANUAL/EN/TYPES.COMPARISONS.PHP#TYPES.COMPARISIONS-LOOSE

ARRAYS

- Created with array()
- 0-based indexing, i.e., like C
- count() returns length
- Three different types
  - Indexed
  - Associative
  - Multi-dimensional
- Tons of built-in functions
  - sort(), rsort(), etc…

```
$numerals = array(1,2,3);
$x[0];          //access 0th element


$ordinals = array("zero"=>0, "one"=>1);

foreach($ordinals as $x=>$x_value)
{
        // access $x and $x_value
}
```

FUNCTIONS

- Have same naming conventions as variables
- Are only executed when called
- Have global scope
- Can have variable parameters and default arguments
- Parameters are passed by value by default, can be pass by reference with (&)
- Can have return statement
- Can be defined in other functions
- Do not need to be defined before being referenced
- Cannot overload or redefine

FUNCTION EXAMPLE 1

```
<?php
   function say_hello_to($name = "World")
   {
      print("Hello ".$name."\n");
    }

    say_hello_to();              // Hello World
    say_hello_to("Foo");         // Hello Foo
?>
```

FUNCTION EXAMPLE 2

```
<?php
    function swap(&$first = NULL, &$second = NULL)
    {
        $temp = $first;
        $first = $second;
        $second = $temp;
    }

    $x = 1; $y = 2;
    print($x . ", " . $y . "\n");   // 1, 2
    swap($x,$y);
    print($x . ", " . $y . "\n");   // 2, 1
?>
```

FUNCTION EXAMPLE 3

```
<?php
    function get_mean()
    {
        $result = 0; $num_args = func_num_args();
        if($num_args == 0) { return; /* do nothing */}

        foreach(func_get_args() as $arg)
        {
            $result += $arg;
        }
        return $result / $num_args;
    }

    print("Mean: " . get_mean(1,2,3,4,5));  // Mean: 3
?>
```

FUNCTION EXAMPLE 4

```
<?php
    function foo()
    {
        print("In foo\n");

        function bar()
        {
            print("In bar\n");
        }
    }

    // bar();    // undefined bar()
    foo();       // "In foo"
    bar();       // "In bar"
?>
```

PHP EXTRAS

- **Object Oriented support since PHP 3.0**
  - Classes with private, public, protected, static, final, etc…
  - Namespaces
  - Interfaces
  - $this references the instance
  - self:: references the class (for globals and statics)
- **Exception handling since PHP 5.0**
  - try, throw, and catch statements

HTML AND PHP EXAMPLE 1

```
<!DOCTYPE html>
<html>
<body>
<?php if(date("H") < 12) { ?>
      <h1>Good morning!</h1>
<?php } else if(date("H") > 11 && date("H") < 18) { ?>
      <h1>Good afternoon!</h1>
<?php } else { ?>
      <h1>Good evening!</h1>
<?php } ?>
</body>
</html>
```

HTML AND PHP EXAMPLE 2

```
<?php
    $start = "<!DOCTYPE html>\n<html>\n<body>\n<h1>Good ";
    $end = "!</h1></body></html>”;

    if(date("H") < 12) {
        $msg = "morning";
    } else if(date("H") > 11 && date("H") < 18) {
        $msg = "afternoon";
    } else {
        $msg = "evening";
    }

    print($start . $msg . $end);
?>
```

HTML FORMS

- You can separate forms into HTML and PHP or combine them into one file
  - Can get/post to self with `$_PHP_SELF`
- PHP has global associative arrays for form data
  - `$_GET`
  - `$_POST`
  - `$_COOKIES`
  - `$_REQUEST`

MYSQL AND PHP

- Same syntax as MySQL C API
  - `mysql_connect();`
  - `mysql_close()`
  - `mysql_select_db();`
  - `mysql_query();`
  - `mysql_error();`
  - `mysql_fetch_row();`
- Original MySQL API is deprecated as of PHP 5.5.x
  - CS servers are not up-to-date, so we continue using older API
- New APIs are MySQLi (MySQL Improved) and PDO (PHP Data Objects)

GOTCHAS

- Use <?php … ?>
- Don’t forget $ before variables
- Single quoted strings are not parsed, but double quoted strings are
- Newlines after closing tags might affect HTML
- Always use === and !== for NULL comparisons
- PHP is not very good with internationalization support
  - If there is a chance of international characters in your code:
    - mb_internal_encoding(’’UTF-8’’);
  - or your HTML output:
    - mb_http_output(’’UTF-8’’);
  - or MySQL data:
    - mysql_set_charset("utf8mb4");

REFERENCES

- http://php.net
- http://www.w3schools.com/php/default.asp
- http://en.wikipedia.org/wiki/PHP
- http://www.utexas.edu/learn/php/index.shtml
- http://phpbestpractices.org