COMP519 Practical 11
PHP (1)

Introduction

- This worksheet contains exercises that are intended to familiarise you with PHP Programming. While you work through the exercises below compare your results with those of your fellow students and ask for help and comments if required.
- You might proceed more quickly if you cut-and-paste code from this PDF file. Note that a cut-and-paste operation may introduce extra spaces into your code. It is important that those are removed and that your code exactly matches that shown in this worksheet.
- The exercises and instructions in this worksheet assume that you use the Department's Linux systems to experiment with PHP.
- To keep things simple, we will just use a text editor, a terminal, and a web browser. You can use whatever text editor and web browser you are most familiar or comfortable with.
- If you do not manage to get through all the exercises during this practical session, please complete them in your own time before the next practical takes place.

Exercises

1. Let us start by re-creating the 'Hello World' PHP script that you have seen in the lectures.

   It is assumed that you have already created a directory

   `$HOME/public_html/`

   in a previous practical session and that the directory is both readable and executable by everyone. Make sure that this is so before you proceed.

   a. Create a file `php11A.php` in your `$HOME/public_html/` directory with the following content:

```
<!DOCTYPE html>
<html lang='en-GB'>
  <head>
    <title>PHP 11A</title>
  </head>
  <body>
    <h1>Our first PHP script</h1>
<?php
  $user = "<your name>";
  print ("<p><b>Hello $user<br>\nHello World!</b></p>\n");
?>
  </body>
</html>
```

   Replace `<your name>` with your own name.

b. Strictly speaking, the file `php11A.php` does not need to have any particular file permissions except that the owner (you) must have read permission. Still, we should make sure that nobody else can access the content of `php11A.php`.

   Open a terminal and execute the command

   ```
   chmod u-x,og-rx $HOME/public_html/php11A.php
   ```

   You will only have to do so once. File permissions should not change while you continue to edit the file.

c. In the terminal, go to the directory in which the file has been stored, and execute the PHP script by using the command

   ```
   php ./php11A.php
   ```

   Check that there are no syntax errors and that the script produces the output

```
<!DOCTYPE html>
<html lang='en-GB'>
  <head>
    <title>PHP 11A</title>
  </head>
  <body>
    <h1>Our first PHP script</h1>
<p><b>Hello <your name><br>
Hello World!</b></p>
  </body>
</html>
```

   (Obviously, with your name appearing instead of `<your name>`.)

d. Now open a web browser and access the URL

   `https://student.csc.liv.ac.uk/~`*`<user>`*`/php11A.php`

   where *`<user>`* should be replaced by your user name.

   Make sure that the web page you are shown corresponds to the HTML code you have seen in Exercise 1c above.

e. It is good practice to add information about who created a script/web page, who claims copyright and when a script/web page was created.

   Add these three pieces of information to the script using `meta` declarations.

2. The following exercises deal with *scalar types* in PHP.

   a. Add the following code to the end of the current PHP script in `php11A.php`:

```
echo "<h2>Exercise 2a</h2>\n";
$text = "stop!";
echo 'Single-quotes: ','don\'t \'don\'t\' "don\'t" $text', "<br>\n";
echo "Double-quotes: ","don't 'don't' \"don\'t\" $text", "<br>\n";
echo 'Single-quotes: ','glass\\table glass\table glass\ntable', "<br>\n";
echo "Double-quotes: ","glass\\table glass\table glass\ntable", "<br>\n";
```

   Save the file and reload the web page again in your web browser. Make sure that you understand why the output looks like it does.

   b. Add the following code to the end of the current PHP script in `php11A.php`:

```
echo "<h2>Exercise 2b</h2>\n";
$student_id = "200846369";
```

```
$staff_id   = "E10481370";
echo "1: \$student_id = $student_id<br>";
echo "1: \$staff_id = $staff_id    <br>";
$student_id++;
$staff_id++;
echo "2: \$student_id = $student_id<br>";
echo "2: \$staff_id = $staff_id    <br>";
$student_id = (int)$student_id + 1;
$staff_id   = (int)$staff_id + 1;
echo "3: \$student_id = $student_id<br>";
echo "3: \$staff_id = $staff_id    <br>";
```

Save the file and reload the web page in your web browser. What happened to `$staff_id` in the last operation?

c. Add the following code to the end of the current PHP script in `php11A.php`:

```
echo "<h2>Exercise 2c</h2>\n";
if (FALSE) {
   error_reporting( E_ALL );
   ini_set('display_errors', 1);
   ini_set('display_startup_errors', 1);
}
echo "Logarithm of 0:    ", log(0),   "<br>\n";
echo "Square root of -2: ", sqrt(-2), "<br>\n";
echo "Dividing 1 by 0:   ", 1/0,      "<br>\n";
echo "Dividing 0 by 0:   ", 0/0,      "<br>\n";
```

Save the file and reload the web page in your web browser. Pay particular attention to the output produced by the last to lines of code. According to the lecture notes, in PHP 5, the result of both `1/0` and `0/0` would be `FALSE` while in PHP7 it would be `INF` and `NAN`, respectively. What does the output tell you about the PHP version you are working with?

d. According to the lecture notes notes there should also be error messages. Check that this is true by executing the script in a terminal.

e. PHP error messages end up in a server log (to which you do not have access). For debugging purposes it makes sense to make them visible to you. We can do so by changing the setting for error reporting and displaying. The three statements in the body of the conditional statement we have introduced in Exercise 2c do that, we just need to execute them: Change `FALSE` to `TRUE` in the condition. Then save the file and reload the page in the web browser. You should now see the two warnings caused by the division operations.

f. Let us explore some mathematical operations more closely. Add the following code to the end of the current PHP script in `php11A.php`:

```
echo "<h2>Exercise 2f</h2>\n";
function sign($n) {
   return ($n > 0) - ($n < 0);
}
function toString($res) {
   return ($res) ? 'TRUE' : 'FALSE';
}
```

```
foreach (array(-4,0,4) as $i) {
   echo "<h3>Computations for $i</h3>\n";
   // $d is the square root of $i
   $d = NULL;
   // $e is 4 divided by $d
   $e = NULL;
   // $f is $e rounded
   $f = NULL;
   echo "sqrt($i)  = $d<br>\n";
   echo "4/$d      = $e<br>\n";
   echo "round($e) = $f<br>\n";
   echo "$e > 0  : ",toString($e > 0), "<br>\n";
   echo "$e <= 0 : ",toString($e <= 0),"<br>\n";
   for ($g = 0.4; $g < 0.7; $g = $g + 0.1) {
      $r = $i + $g * sign($i);
      // $s is $r rounded
      $s = NULL;
      echo "round($r) = $s<br>\n";
   }
}
```

Replace each occurrence of `NULL` with code that implements a computation as described in the comment above it. Save the file then reload the page in your web browser. Check the output produced by the new code and compare it with that of the file `jsDemo06B.html` you have created for Practical 6.

g. The precision of 64-bit floating-point numbers is necessarily limited. This problem is aggravated by the fact that the binary representation of these floating-points numbers allows to precisely represent a lot of binary fractions, e.g. 1/8 and 1/128, but not the decimal fractions that humans use, e.g. 1/10 and 1/100. Let us see whether PHP has the same weaknesses as JavaScript in that respect.

Add the following code to the end of the current PHP script in `php11A.php`:

```
echo "<h2>Exercise 2g</h2>\n";
$x = 0.3 - 0.2;
$y = 0.2 - 0.1;
echo "\$x = $x<br>";
echo "\$y = $y<br>";
echo "(\$x == \$y) = ",toString($x == $y),"<br>\n";
echo "(\$x == 0.1) = ",toString($x == 0.1),"<br>\n";
echo "(\$y == 0.1) = ",toString($y == 0.1),"<br>\n";
```

Save the file then reload the page in your web browser. Check the output produced by the new code and compare it with that of the file `jsDemo06C.html` you have created for Practical 6.

3. The following example deals with *types*, *type casting*, and the identification of types in PHP.

a. Open a text editor, enter the HTML markup and PHP code below, and save it as a file `php11B.php` in `$HOME/public_html/`.

```
<!DOCTYPE html>
<html lang='en-GB'>
```

```
  <head>
    <title>PHP 11B</title>
  </head>
  <body>
    <h1>Types and Type Casting</h1>
<?php
  error_reporting( E_ALL );
  ini_set('display_errors', 1);
  ini_set('display_startup_errors', 1);
  echo "<h2>Types and Type Casting</h2>\n";
  $mode = rand(1,4);
  if ($mode == 1)
    $randvar = rand();
  elseif ($mode == 2)
    $randvar = (string) rand();
  elseif ($mode == 3)
    $randvar = rand()/(rand()+1);
  else
    $randvar = (bool) $mode;
  echo "Random scalar: $randvar<br>\n";
?>
  </body>
</html>
```

Make sure that the file permissions are set as in Exercise 1b.

b. Execute the script several times using the web browser. See how the value of `$randvar` changes every time that the script is executed.

c. Extend the script with code that displays

- `This is an integer` if `$randvar` stores an integer
- `This is a floating-point number` if `$randvar` stores a floating-point number
- `This is a string` if `$randvar` stores a string
- `I don't know what this is` in any other case

after the line `Random scalar: ...`. You are not allowed to use the value of `$mode` for this purpose.

Hints: Recall from the lecture notes what `gettype()`, `is_int()`, `is_string()`, etc do. Also have a look at switch statements in PHP:

http://php.net/manual/en/control-structures.switch.php

d. Test your solution by execute the script several times using the web browser.

4. The following example deals with *comparison operators* and the generation of HTML tables in PHP.

a. Open a text editor, enter the HTML markup and PHP code below and save it as a file `php11C.php` in `$HOME/public_html/`.

```
<!DOCTYPE html>
<html lang='en-GB'>
  <head>
    <title>PHP 11C</title>
```

```
  </head>
  <body>
    <h1>HTML Tables and Comparisons</h1>
<?php
  error_reporting( E_ALL );
  ini_set('display_errors', 1);
  ini_set('display_startup_errors', 1);

  echo "<h2>Exercise 4</h2>\n";
  $array = [0,123,1.23e2,"123",TRUE,FALSE];
  echo "<table border='1' cellpadding='5'>\n";
  foreach ($array as $a)
    foreach ($array as $b) {
      $val_a = gettype($a)."(".var_export($a,true).")";
      $val_b = gettype($b)."(".var_export($b,true).")";
      echo "<tr>";
      printf("<td>%20s == %20s -> %5s</td>", $val_a,$val_b,
             ($a == $b) ? "TRUE" : "FALSE");
      printf("<td>%20s === %20s -> %5s</td>",$val_a,$val_b,
             ($a === $b) ? "TRUE" : "FALSE");
      printf("<td>%20s != %20s -> %5s</td>", $val_a,$val_b,
             ($a != $b) ? "TRUE" : "FALSE");
      printf("<td>%20s !== %20s -> %5s</td>",$val_a,$val_b,
             ($a !== $b) ? "TRUE" : "FALSE");
      echo "</tr>";
   }
 echo "</table>\n";
?>
  </body>
</html>
```

Make sure that the file permissions are set as in Exercise 1b.

b. Save the file and execute the script using the web browser. Analyse the table that the additional code produces and make sure that you understand its contents. Also make sure that you understand how the code above produces that table, including how `printf` works. (Generating tables is a typical task for which PHP code is used.)