CS571: Programming Languages

Overview: PHP

- PHP is a popular server-side scripting language
  - PHP stands for "Hypertext Preprocessor"
  - Good for creating dynamic websites
- Applications
  - Generate dynamic content
  - Store cookies
  - Send email
  - store data in a database,

PHP

- Syntax based on Perl and C
- File ends in .php
- PHP code is specified with the <?php … ?> tag
- Program lines end in ";" or you get an error
- Comments:
  - "//" at the beginning of each line,
  - /* … */"

PHP: Data Types

Data type	Description
int, integer	Whole numbers
float, double	Real numbers
string	Text enclosed in either single (' ') or double (" ") quotes.
bool	True or false.
array	Group of elements of the same type.

PHP: Variables (test.php)

- Variables: Begin with "$"
- Can have different types at different time
- Assignment by value: **$bar = $foo;**
- Assigned by reference: **$bar = &$foo;**

```
<?php
  $name = "bil";
  echo "my name is $name";
  if ($name == "bil")
    {echo "got a match!"; } ?>
```


PHP: Variables (test.php)

- Variables: Begin with "$"
- Can have different types at different time
- Assignment by value: **$bar = $foo;**
- Assigned by reference: **$bar = &$foo;**

```
<?php
  $name = "bil";
  echo "my name is $name";
  if ($name == "bil")
    {echo "got a match!"; } ?>
```

Output:
my name is bilgot a match!

Operators (op.php)

- Arithmetic (+, -, *, /, %, +=, -=, *=, /=, %=) and

```
$a = 1 + 2;
echo "$a <br>";
$a += 4;
echo "$a <br>";
```

- Logic (&&, ||, !)
- Comparison (==, !=, <, >, <=, >=)

String

- Single-quoted string or double-quoted string
- . : Concatenation

```
$b = "Hello" . "world";
```

- strlen(): returns the length of a string

```
echo strlen("Hello world!");
```

- strpos(): search for a string. If a match is found, then return the position of the first match; otherwise, return FALSE.

```
echo strpos("Hello world!","world");
```


String

- Single-quoted string or double-quoted string
- . : Concatenation

```
$b = "Hello" . "world";
```

- strlen(): returns the length of a string

```
echo strlen("Hello world!");
```

output: 12

strpos(): search for a string. If a match is found, then return the position of the first match; otherwise, return FALSE.

```
echo strpos("Hello world!","world");
```


String

- Single-quoted string or double-quoted string
- . : Concatenation

```
$b = "Hello" . "world";
```

- strlen(): returns the length of a string

```
echo strlen("Hello world!");
```

output: 12

strpos(): search for a string. If a match is found, then return the position of the first match; otherwise, return FALSE.

```
echo strpos("Hello world!","world");
```

output: 6

Control Structures

```
if (condition) {
   statements
}elseif {
   statements
} else{
   statements
}
```

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

```
for (initialization; condition; modifier){
   statements
}
```


Arrays

- $cars=array("Ford", "BMW", "Toyota");
- Index starts at 0
- To access an element: $arrayname[$index] e.g. $cars[0]
- count(): return the length of an array

```
echo count($cars);
```

Arrays

- $cars=array("Ford", "BMW", "Toyota");
- Index starts at 0
- To access an element: $arrayname[$index]
  e.g. $cars[0]
- count(): return the length of an array

```
echo count($cars);
```

Output: 3

Arrays (pop.php)

- array_pop(): remove the last element of the array
- array_push(): add an element to the end of the array

```
<?php
    $a=array("red","green");
    array_push($a,"blue","yellow");
    print_r($a);
    array_pop($a);
    print_r($a);
?>
```


Arrays (pop.php)

- array_pop(): remove the last element of the array
- array_push(): add an element to the end of the array

```
<?php
    $a=array("red","green");
    array_push($a,"blue","yellow");
    print_r($a);
    array_pop($a);
    print_r($a);
?>
```

Output:

```
Array ( [0] => red [1] => green [2] => blue [3] => yellow )
Array ( [0] => red [1] => green [2] => blue )
```


Sorting an Array (sort.php)

- sort(): sort an array

```
<?php
    $numbers=array(4,6,2,22,11);
    sort($numbers);
    print_r($numbers);
?>
```


Sorting an Array (sort.php)

- sort(): sort an array

```
<?php
    $numbers=array(4,6,2,22,11);
    sort($numbers);
    print_r($numbers);
?>
```

Output:

```
Array ( [0] => 2 [1] => 4 [2] => 6 [3] => 11 [4] => 22 )
```


Arrays (test3.php)

- Example: print all elements in the array $first

```
<?php
    $first= array(1, 2, 3, 4, 5);
    for ( $i = 0; $i < count( $first ); $i++ ) {
        print( "Element $i is $first[$i] <br >" );}
?>
```

Arrays (test3.php)

- Example: print all elements in the array $first

```
<?php
    $first= array(1, 2, 3, 4, 5);
    for ( $i = 0; $i < count( $first ); $i++ ) {
        print( "Element $i is $first[$i] <br >" );}
?>
```

Output:

```
Element 0 is 1
Element 1 is 2
Element 2 is 3
Element 3 is 4
Element 4 is 5
```


Arrays (test4.php)

- Use array to implement the hash table

```
<?php    $fourth = array(
    "January"  => "first",  "February" => "second",
    "March"    => "third",  "April"    => "fourth",
    "May"      => "fifth",  "June"     => "sixth",
    "July"     => "seventh", "August"  => "eighth",
    "September" => "ninth",  "October" => "tenth",
    "November"  => "eleventh","December" => "twelfth" );

    foreach ( $fourth as $key => $value )
        print( "$key is the $value month <br>" );
?>
```


Arrays (test4.php)

Output:

```
January is the first month
February is the second month
March is the third month
April is the fourth month
May is the fifth month
June is the sixth month
July is the seventh month
August is the eighth month
September is the ninth month
October is the tenth month
November is the eleventh month
December is the twelfth month
```


Functions (function.php)

- function functionName() {
  code to be executed;
  }
- A function name can start with a letter or underscore (not a number).
- Function names are NOT case-sensitive.
- Example:

```
<?php
    function writeMsg() { echo "Hello world!";}
    writeMsg(); // call the function
?>
```


Functions (function.php)

- function functionName() {
  code to be executed;
  }
- A function name can start with a letter or underscore (not a number).
- Function names are NOT case-sensitive.
- Example:

```
<?php
    function writeMsg() { echo "Hello world!";}
    writeMsg(); // call the function
?>
```

Output: Hello world

Functions with Arguments (function1.php)

```
<?php
    function familyName($fname,$year) {
    echo "$fname was born in $year <br>";  }

    familyName("Tom","1975");
    familyName("Alice","1978");
?>
```

Functions with Arguments (function1.php)

```
<?php
   function familyName($fname,$year) {
   echo "$fname was born in $year <br>";  }

   familyName("Tom","1975");
   familyName("Alice","1978");
?>
```

Output:

Tom was born in 1975
Alice was born in 1978

Returning Values

```
<?php
   function sum($x,$y) {
     $z=$x+$y;
     return $z;
   }
   echo "5 + 10 = " . sum(5,10) . "<br>";
   echo "7 + 13 = " . sum(7,13) . "<br>";
   echo "2 + 4 = " . sum(2,4);
?>
```


Returning Values

```
<?php
   function sum($x,$y) {
     $z=$x+$y;
     return $z;
   }
   echo "5 + 10 = " . sum(5,10) . "<br>";
   echo "7 + 13 = " . sum(7,13) . "<br>";
   echo "2 + 4 = " . sum(2,4);
?>
```

Output:

5 + 10 = 15
7 + 13 = 20
2 + 4 = 6

Pass by Reference

```
<?php
   function inc($num)
   {   $num ++; }

   $n = 1;
   inc( &$n );
   echo "$n<br />";
   ?>
```


Pass by Reference

```
<?php
   function inc($num)
   {   $num ++; }

   $n = 1;
   inc( &$n );
   echo "$n<br />";
   ?>
```

Output: 2

Regular Expression

Quantifier	Matches
{n}	Exactly n times.
{m,n}	Between m and n times.
{n,}	n or more times.
+	One or more times (same as {1,}).
*	Zero or more times (same as {0,}).
?	Zero or one time (same as {0,1}).
[ ]	Range
^	beginning
$	end

Form (form.html, get -- form1.html)

- Example: prompt the user to enter the phone # and check if the # is valid using the regular expression

```
<html>
<body>
//after clicking register, the data will be sent to form.php
 <form method = "post" action = "form.php">
 <input type = "text" name = "phone" /><br />
 <input type = "submit" value = "register"/>
</body>
</html>
```


Form (form.php, get - form1.php)

- Example: prompt the user to enter the phone # and check if the # is valid

```
<?php
//$_POST: An array of variables passed to .php via the
//HTTP POST method.
foreach ( $_POST as $element => $value )
    print( "$element is $value <br>" );
extract($_POST);
//check if the phone # is (xxx)xxx-xxxx
if ( !ereg( "^\([0-9]{3}\)[0-9]{3}-[0-9]{4}$",$phone ) ){
     print("invalid phone number <br />");}
else {print("valid phone number<br />");}
die(); //terminate script execution
?>
```


File Operation

- fopen (filename, modes): open a file
  - r: read
  - w: write
  - a: append
- fread(filename, size): read a file
  - size: the maximum number of bytes to read
- fgets(filename): read one line
- fgetc(filename): read one character
- fclose(filename): close the file
- feof: check end-of-file

File Operation & Dynamic Content Generation (product.html)

- Example: prompt the user to enter the product name and return the product information

```
<html>
<body>
 <form method = "post" action = "product.php">
 <input type = "text" name = "product" /><br />
 <input type = "submit" value = "search"/>
</body>
</html>
```


File Operation & Dynamic Content Generation (product.php)

- Example: prompt the user to enter the product name and return the product information

```
<?php
 extract($_POST);
 if ( !( $file = fopen( "product.txt","r" ) ) ) {
    print( "Could not open the product file " );
    die();
 }
```


File Operation & Dynamic Content Generation (product.php)

- Example: prompt the user to enter the product name and return the product information

```
print("<table border = \"1\">");
while (!feof($file)) {
     $line = fgets( $file ); // read a line from file
     // remove newline character from end of line
     $line = chop( $line );
     //Split string into substrings given a certain delimiter
     $field = split( ",", $line, 3 );
     if ( $product == $field[ 0 ] ) {
          print("<tr>"); print("<td>$field[1]</td>");
          print("<td>$field[2]</td>"); print("</tr>");}}
     print("</table>");  fclose($file); ?>
```

Sending An Email

- Example: prompt the user to enter his/her email address and send an email to the user.
- **email.html:**

```
<html>
<body>
<form method = "post" action = "email.php">
<input type = "text" name = "email" /><br />
<input type = "submit" value = "send email"/>
</body>
</html>
```

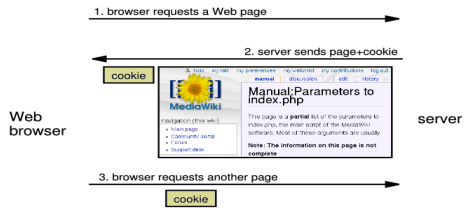

Sending An Email

- Example: prompt the user to enter his/her email address and send an email to the user.
- **email.php:**

```
<?php extract($_POST);
$email .= ', cs5712013@gmail.com';
$subject = "Hi!";
$body = "Hi,\n\nHow are you?";
$headers = 'From: cs5712013@gmail.com' . "\n" .
'Reply-To: cs5712013@gmail.com' . "\n" .
'X-Mailer: PHP/';
if (mail($email, $subject, $body, $headers)) {
      echo("<p>Email successfully sent!</p>");
} else { echo("<p>Email delivery failed!<p>"); } ?>
```


Cookies

- Cookie: a small piece of data sent from a website to a user's web browser.



- Record the user's browsing activities (e.g. logging in, clicking particular buttons)

Cookie: Example (cookie.html)

- Example: prompt the user to enter his/her name and the age, and then store a cookie containing such information on the disk.

```
<html> <body>
 <form method = "post" action = "cookie.php">
 Name:  <input type = "text" name = "name" /><br />
 Age:  <input type = "text" name = "age" /><br />
 <input type = "submit" value = "set cookie"/>
</body> </html>
```


Cookie: Example (cookie.php)

- Example: prompt the user to enter his/her name and the age, and then store a cookie containing such information on the disk.

```
<?php
   extract($_POST);
   //(name of the cookie, value, expiration time).
   setcookie("name", $name, time()+60*60*24*5);
   setcookie("age", $age, time()+60*60*24*5);
?>
```

IE: C:\Documents and Settings\<user name>\Local Settings\Temporary Internet Files

Example: Accessing a Database

- Example: access database "Products".

```
<? php
// open a connection to a MySQL server
//$database: MySQL link identifier on success or false on failure.
if ( !( $database = mysql_connect( 'localhost', 'user', 'passwd' ) ) )
      die( "Could not connect to database" );

// specifies the database to be queried
if ( !mysql_select_db( "Products", $database ) )
      die( "Could not open Products database" );
```

Example: Accessing a Database

Example: access database "Products".

```
//sends a query
if ( !( $result = mysql_query( $query, $database ) ) ) {
      print( "Could not execute query! <br >" );
       die( ); }

//returns an array containing the elements of each row in the result of the
query.
for ( $counter = 0; $row = mysql_fetch_row( $result ); $counter++ ){
      foreach ( $row as $key => $value )
            print( "$value" );}
 mysql_close();
?>
```

43

References

- http://www.w3schools.com/php/