CSC-318
Web Technology
(BSc CSIT, TU)

Ganesh Khatri
kh6ganesh@gmail.com

Chapter6 : PHP Functions

- Inbuilt PHP Functions
- Custom Functions
- Arguments
- Return Values

PHP Inbuilt Functions

- A function is a piece of code that can be run
- Usually it will perform an action that produces a result
- Every function has a name
- You can run the function using the name followed by brackets
- Functions can be used in the same way as variables

PHP Inbuilt Functions

- The rand() function can be used to generate a random number

```php
<?php
echo rand();
?>
```

```
Output:
185902316
```

- Each time you run the script it will generate a random number

PHP Inbuilt Functions

- The pi() function calculates mathematical PI to 14 decimal places

```php
<?php
echo pi();
?>
```

```
Output:
3.1415926535898
```

str_replace()

- The str_replace() function is used to replace part of a string with another string
- It takes three arguments:
  - A string to look for
  - A string to replace it with
  - The original string

```php
<?php
echo str_replace('NAME', 'Bob', 'Hello NAME');
?>
```

```
Output:
Hello Bob
```

Str_replace() arguments

```php
<?php
echo str_replace('NAME', 'Bob', 'Hello NAME');
?>
```

```
Output:
Hello Bob
```

- This can be read as:
  - Take the string 'Hello NAME’
  - Look for 'NAME' in that string
  - Replace it with 'Bob'

Date Function

- The date function can be used to display the current date based on the system clock
- It takes a single argument:
  - A specially formatted string that represents a date format.
  - For example: d/m/Y means day/month/year

```php
<?php
echo date('d/m/Y');
?>
```

```
Output:
28/10/2015
```

Date Function

- Changing the string supplied to the date function will change its output.
  - H:i is used to display the time in Hours:Minutes format

```php
<?php
echo date('H:i');
?>
```

```
Output:
13:44
```

- For a full list of supported formats, see: https://www.php.net/manual/en/function.date.php

Other Inbuilt Functions

```php
<?php
echo strtoupper('Hello World');
?>
```

```
Output:
HELLO WORLD
```

```php
<?php
echo strtolower('Hello World');
?>
```

```
Output:
hello world
```

```php
<?php
echo str_pos('Hello World', 'o');
?>
```

```
Output:
4
```

```php
<?php
echo strlen('Hello World');
?>
```

```
Output:
11
```

Custom Functions

- You can create your own functions
- To create a function you must give it a name
- At its most basic, a function can just be repeat a block of code throughout your program

```php
<?php
function hello() {
    echo '<p>Hello World</p>';
}
?>
```

Custom Functions

- Declaring a function alone will have no effect
- After the function is defined, you must call it
- You can call a custom function like an inbuilt function

```php
<?php
function hello() {
    echo '<p>Hello World</p>';
}

hello();
?>
```

```
Output:
<p>Hello World</p>
```

Custom Functions

```php
function navigation() {
    echo '<ul>';
    echo '  <li><a href="index.php">Home</a></li>';
    echo '  <li><a href="about.php">About</a></li>';
    echo '  <li><a href="contact.php">Contact</a></li>';
    echo '</ul>';
}

?>
<!DOCTYPE html>
<html>
        <head>
                <title>My Site</title>
        </head>
        <body>
                <nav>
                        <?php navigation(); ?>
                </nav>

                <main>
                        <p>Lorem ipsum...</p>
                </main>

                <footer>
                        <?php navigation(); ?>
                </footer>
        </body>
</html>
```

```html
<!DOCTYPE html>
<html>
<head>
     <title>My Site</title>
</head>
<body>
     <nav>
          <ul>
               <li><a href="index.php">Home</a></li>
               <li><a href="about.php">About</a></li>
               <li><a href="contact.php">Contact</a></li
          </ul>
     </nav>
          <main>
          <p>Lorem ipsum...</p>
     </main>
          <footer>
          <ul>
               <li><a href="index.php">Home</a></li>
               <li><a href="about.php">About</a></li>
               <li><a href="contact.php">Contact</a></li
          </ul>
     </footer>
</body>
</html>
```

Arrays and User Input

- Introduction to arrays
- User input and Super Globals
- Useful array functions

Arrays

- Arrays allow you to store more than one value inside a single variable
- This is useful for many tasks whenever you need a collection of values
- An array lets you store a value under a key
- An array lets you store a value under a key
- To create an array, create a variable using two square brackets:

```php
<?php

$myArray = [];

?>
```

Arrays

- Once you have declared a variable as an array, you can write values to it under keys
- For example:

```php
<?php

$myArray = [];

$myArray[1] = 'Value 1';
$myArray[2] = 'Value 2';
$myArray[3] = 'Value 3';

?>
```

Arrays

- Once you've written to an array key you can read the value back out of the array using the same square bracket notation
- You can use array values like any other variable

```php
<?php

$myArray = [];

$myArray[1] = 'Value 1';
$myArray[2] = 'Value 2';
$myArray[3] = 'Value 3';


echo '<p>' . $myArray[2] . '</p>';
?>
```

```
Output:
<p>value 2</p>
```

Numerical arrays

- Instead of:

```php
<?php
$myArray = [];

$myArray[1] = 'Value 1';
$myArray[2] = 'Value 2';
$myArray[3] = 'Value 3';
?>
```

- You can achieve the same result with the shorthand notation:

```php
<?php

$myArray = ['Value 1', 'Value 2', 'Value 3'];

echo '<p>' . $myArray[2] . '</p>';

?>
```

Arrays

```php
<?php

$myArray = ['Value 1', 'Value 2', 'Value 3'];

echo '<p>' . $myArray[2] . '</p>';
```

```
Output:
<p>Value 3</p>
```

- Note: Arrays start from zero!

```php
<?php

$myArray = ['Value 1', 'Value 2', 'Value 3'];

echo '<p>' . $myArray[0] . '</p>';
echo '<p>' . $myArray[1] . '</p>';
echo '<p>' . $myArray[2] . '</p>';
?>
```

```
Output:
<p>Value 1</p>
<p>Value 2</p>
<p>Value 3</p>
```

Array Sizes

- Every array has a size
- This is the number of elements in the array
- This can be read using the count() function on the array variable

```php
<?php

$myArray = ['Value 1', 'Value 2', 'Value 3'];

echo count($myArray);

?>
```

```
Output:
3
```

Looping through an array

- You can use a for loop along with count to loop through each element in an array

```php
$myArray = ['Value 1', 'Value 2', 'Value 3'];

for ($i = 0; $i < count($myArray); $i++) {
        echo '<p>' . $myArrray[$i] . '</p>';
}
```

```
Output:
<p>Value 1</p>
<p>Value 2</p>
<p>Value 3</p>
```

Array Size

- In PHP an array does not have a fixed size. Elements can be added at any point during the program
- You can append an element to the next available index using the code:

```
$myArray[] = 'next value';
```

Adding to the end of the array

```php
<?php

$myArray = [];
var_dump($myArray);

$myArray[] = 'value 1';
var_dump($myArray);

$myArray[] = 'value 2';
var_dump($myArray);

$myArray[] = 'value 3';
var_dump($myArray);
?>
```

```
Output:
array (size=0)
  Empty

array (size=1)
  0 => string 'value 1' (length=7)

array (size=2)
  0 => string 'value 1' (length=7)
  1 => string 'value 2' (length=7)

array (size=3)
  0 => string 'value 1' (length=7)
  1 => string 'value 2' (length=7)
  2 => string 'value 3' (length=7)
```

foreach loop

- There is a second type of for loop which will loop through all elements in an array
- This is foreach and it does not use a counter
- The foreach loop takes an array and loops over it

foreach loop

```php
<?php

$myArray = [];

$myArray[0] = 'Zero';
$myArray[1] = 'One';
$myArray[2] = 'Two';
$myArray[3] = 'Three';
$myArray[4] = 'Four';

$myArray[7] = 'Seven';
$myArray[8] = 'Eight';
$myArray[9] = 'Nine';


foreach ($myArray as $key => $value) {
        echo '<p>Key : ' . $key . ', Value: ' . $value  '</p>';
}

?>
```

```
Output:
<p>Key: 0, Value: Zero</p>
<p>Key: 1, Value: One</p>
<p>Key: 2, Value: Two</p>
<p>Key: 3, Value: Three</p>
<p>Key: 4, Value: Four</p>
<p>Key: 7, Value: Five</p>
<p>Key: 8, Value: Eight</p>
<p>Key: 9, Value: Nine</p>
```

foreach loop

- Foreach is usually simpler than for to loop through an array
- You should use foreach instead of for when looping over an array
- Foreach will only loop over keys that are set

Array Keys

- In PHP array keys can be either integers or strings

```php
<?php

$myArray = [];

//String Key
$myArray['stringKey'] = 'value 1';

//Numeric key
$myArray[5] = 'value 2';

var_dump($myArray);

?>
```

```
Output:

array (size=2)
  'stringKey' => string 'value 1' (length=7)
  5 => string 'value 2' (length=7)
```

Array Keys

- You can also use foreach with arrays that have string keys

```php
<?php

$myArray = [];

$myArray['one'] = 'value 1';
$myArray['two'] = 'value 2';


foreach ($myArray as $key => $value) {
        echo '<p>Key : ' . $key . ', Value: ' . $value . '</p>';
}

?>
```

```
Output:

<p>Key: one, Value: value 1</p>
<p>Key: two, Value: value 2</p>
```

User input - Basics

- PHP has several “superglobals” which are variables that are available at any point in the program's code
- These are generally used for input from users
- There are two ways of capturing input from users over the web:
  - GET
  - POST

$_GET

- GET is a URI Variable and part of the HTTP protocol
- You can change the value of the PHP $_GET variable by changing the URI
- Until now you have accessed PHP scripts using the file name
- e.g. http://example.com/file.php

$_GET

- $_GET is an array that uses string keys
- Instead of setting the array contents in the PHP code, PHP automatically sets $_GET based on the URL the page has been accessed on
- You can access the page on
- file.php?name=John and it will set the $_GET variable's name key to John

$_GET

- file.php contents

```php
<?php

echo 'Hello ' . $_GET['name'];

?>
```

- Going to http://domain.com/file.php will generate the output:

```
Hello ERROR: Undefined Index “name”
```

$_GET

- It will cause an error because the name key has not been set
- To set a key you can amend the URL with ?name=John
- http://domain.com/file.php?name=John

```php
<?php

echo 'Hello ' . $_GET['name'];

?>
```

```
Hello John
```

Multiple $_GET Variables

- You can specify more than one variable by separating them with am ampersand (&)
- file.php?key1=value1&key2=value2

```php
<?php

var_dump($_GET);

?>
```

```
array (size=1)
  'key1' => string 'value1' (length=6)
  'key2' => string 'value2' (length=6)
```

```php
<?php

echo '<p>' . $_GET['key1'] . '</p>';

echo '<p>' . $_GET['key2'] . '</p>';

?>
```

```
<p>value1</p>
<p>value2</p>
```

Removing elements from an array

- Once an array has been created, you can remove elements using the unset function

```php
<?php

$myArray = [];

$myArray['key1'] = 'value 1';
$myArray['key2'] = 'value 2';
$myArray['key3'] = 'value 3';


var_dump($myArray);

unset($myArray['key2']);

var_dump($myArray);
?>
```

```
array (size=3)
  'key1' => string 'value 1' (length=7)
  'key2' => string 'value 2' (length=7)
  'key3' => string 'value 3' (length=7)


array (size=2)
  'key1' => string 'value 1' (length=7)
  'key3' => string 'value 3' (length=7)
```

PHP Form Handling

- The PHP Super Globals
- $_GET – gets data from a form that uses GET method
- $_POST - gets data from a form that uses POST method

$_GET forms

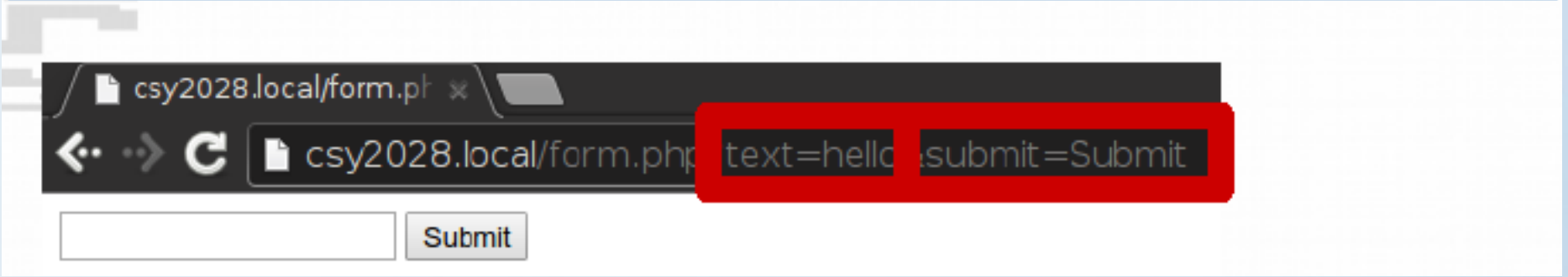
- When you set the form method to GET (it is case sensitive!) and the form is submitted, the url query string is automatically appended with form data
- When a GET form is submitted, it serializes all the form elements and automatically generates the URL parameters and navigates to the page:

```
<form action="form.php" method="GET">

        <input type="text" name="text" />

        <input type="submit" value="Submit" name="submit" />

</form>
```



$_GET forms

- $_GET array automatically set when GET form is submitted

```
<form action="form.php" method="GET">

        <input type="text" name="input1" />
        <input type="text" name="input2" />

        <input type="submit" value="Submit" name="submit" />

</form>

<?php
var_dump($_GET);
?>
```

```
array (size=3)
  'input1' => string 'text box 1' (length=10)
  'input2' => string 'text box 2' (length=10)
  'submit' => string 'Submit' (length=6)
```

$_POST Forms

- Post forms work in an almost identical way to GET forms
- You can change the method attribute to POST (uppercase!) to make the form a POST form:
- Form data is not append in URL while submitting POST form

```
<form action="form.php" method="POST">

        <input type="text" name="input1" />
        <input type="text" name="input2" />

        <input type="submit" value="Submit" name="submit" />

</form>
```

Differences between GET and POST forms

- POST forms do not amend the URL
- You cannot send values to POST forms via the URL
- With GET forms, you can submit the form and then hyperlink to the results by copying the URL
- With POST you must fill the form manually each time

- Generally POST forms should be used:
  - When something on the server needs to be changed (e.g. inserting/updating information, logging in)
- Generally GET forms should be used:
  - When a set of results should be generated (e.g. search results, switching page)
- If unsure, use POST
- GET form can not upload files while POST form can

Exercise 1

- Write a program which simulates a game of rock-paper-scissors.
- When the page loads there should be a choice of 3 links, one for rock, one for paper and one for scissors
- When a link is clicked it should display something like:
  - "You chose rock. The computer chose scisssors, you win!"
- Hint: You will need to use the random number generator to make the computer pick a move.
- If both players chose the same print "It's a tie"
- Grade B: Can you do this with a single .php file?
- Grade A: Extend the final screen to include the first page with the words "Play again?" and links to select your move.

- Amend the rock-paper-scissors game to use a <select> element and a POST form instead of links

Exercise 2

- Use an array to store the following people and their extension numbers
  - John –389
  - Kate –012
  - Sue –586
  - Dave –675
  - Jo –434
- Using a loop print out each of the names as links in a list
- When you click on a name, display (for example) "John is on extension 389"