# Dynamic Web Development

## Lecture 9 – Introduction to PHP 2

# Mixing HTML with PHP

```
<?php $idcode = "Kevin"; ?>
<html>
<head>
<title>My webpage</title>
</head>
<body>
<p>hello there </p>
<?php echo $idcode; ?>
<p> this is a webpage</p>
<p>with some text</p>
</body>
</html>
```

OR

```
<?php
$idcode = "Kevin";
echo"<html>";
echo "<head>";
echo "<title>My webpage</title>";
echo "</head>";
echo "<body>";
echo "<p>hello there </p>";
echo $idcode;
echo "<p> this is a webpage</p>";
echo "<p>with some text</p>";
echo "</body>";
echo "</html>";
?>
```

# *Revision*

Use echo or print to send output to the browser.

Variable names start with $.

The full stop is used as the string concatention operator.

Comments are marked by //  or  /* ... */

The standard arithmetic and logical operators are used.

Selection is done by using the if..else statement.

Iteration is done by using the while loop statement.

# *Overview*

- Switch Statement
- Using Includes
- Using Functions
- Arrays
- Foreach statement

# Switch statement

```
switch ($i)
{
case 0:
        echo "i equals 0";
        break;
case 1:
        echo "i equals 1";
        break;
case 2:
       echo "i equals 2";
       break;
}
```

This will test the value of $i against each case variable in turn and execute the matching code.

# Switch statement 2

```
switch ($i)
{
case "apple":
                echo "i is apple";
                break;
case "bar":
                echo "i is bar";
                break;
case "cake":
                echo "i is cake";
                break;
}
```

You can also match against strings.

# *Using Includes*

If you want the same message to appear at the top of each web page, to save typing it in each time, you can put it in a separate file, called (for example) header.php.

In the webpage, you can just add the line:

```
<?php  include "header.php";  ?>
```

and the PHP parser will insert the contents of the include file into the webpage.

# *Using Functions*

You can also use functions in PHP.

They are small self-contained sections of code.

```
<?php

function multiply( $num1, $num2 )
  {
    $total = $num1 * $num2;
    return $total;
  }

$mynum = multiply( 5, 10 );
echo $mynum;
?>
```

# *Variable Scope in Functions*

You have to use the GLOBAL command to make an outside variable's name accessible inside the function.

```
<?php

function foo( )
{
GLOBAL $bar;
$bar = "Kevin";
echo $bar;
}

$bar = "Helen";
foo();
echo $bar;

?>
```

# *Using Functions*

This does not apply to variables that are global by default ($_POST, $_GET etc).

They can be located anywhere within the script and called from anywhere.

It makes sense to list all of your functions together at the top of your page so that you know where to find them.

Better still, put them in an include file so that they can be used on several pages.

# Arrays

You can set up arrays of data items.

These are lists of data items which can be accessed by means of a subscript or key value.

| | firstname | lastname | age |
|---|---|---|---|
| husband | Albert | Einstein | 124 |

# Assigning values 1

```
<?php
$husband = array("firstname"=>"Albert",
                 "lastname"=>"Einstein",
                 "age"=>"124");

echo $husband["firstname"];
?>
```

Note use of => to assign values to the key of an array.

# Assigning values 2

Another way of doing the same thing is this:

```
<?php
$husband["firstname"] = "Albert";
$husband["lastname"] = "Einstein";
$husband["age"] = "124";

echo $husband["firstname"];
?>
```

# *Two Dimensional Arrays*

You can have two dimensional arrays, as shown below.

It is even possible to have three or more dimensions.

| | firstname | lastname | age |
|---|---|---|---|
| husband | Albert | Einstein | 124 |
| wife | Mileva | Einstein | 123 |

# Multi-dimensional Arrays

So you can do this:

```
<?php
$table1 = array(
        "husband" => array("firstname"=>"Albert",
                           "lastname"=>"Einstein",
                           "age"=>"124"),
        "wife" => array("firstname"=>"Mileva",
                           "lastname"=>"Einstein",
                           "age"=>"123")
                );

echo $table1["husband"]["firstname"];
?>
```

# Multi-dimensional Arrays

Another way of doing the same thing is this:

```
<?php
$table1["husband"] = array("firstname"=>"Albert",
                           "lastname"=>"Einstein",
                           "age"=>"124"
                          );

$table1["wife"] = array("firstname"=>"Mileva",
                        "lastname"=>"Einstein",
                        "age"=>"123"
                       );

echo $table1["husband"]["firstname"];
?>
```

# *Implicit keys*

If you don't want to give names to the keys, PHP will assign numeric integers starting at 0.

```
<php
   $flavour[] = "vanilla";
   $flavour[] = "raspberry ripple";
   $flavour[] = "chocolate";

   echo $flavour[1];
?>
```

# *Using the array () function*

Another way of doing the same thing, is this.

```
<php
$flavour = array("vanilla", "raspberry ripple", "chocolate");

echo $flavour[1];
?>
```

This is the easiest way to set up an array, if you know the values in advance.

# *Sorting Arrays*

PHP provides some functions for sorting arrays. These are the common ones.

| | |
|---|---|
| arsort(array) | Descending value order and maintains the key/value relationship. |
| asort(array) | Ascending value order and maintains the key/value relationship. |
| rsort(array) | Descending value order. |
| sort(array) | Ascending value order. |

See http://www.w3schools.com/php/php_ref_array.asp

The print_r function is also useful when using arrays.

# *foreach command*

The foreach command can be used to iterate through an array.

It can only be used on arrays.

```
foreach ( arrayname as variable)
    {
      echo variable ;
    }
```

# foreach command

```
<php

  $flavour[] = "vanilla";
  $flavour[] = "raspberry ripple";
  $flavour[] = "chocolate";

  echo "My favourite flavours are:<br>";

foreach ($flavour as $currentvalue)
   {
     echo $currentvalue . "<br>\n";
   }

?>
```

When PHP is processing arrays, it keeps track of which is the current array element by means of an internal pointer.

It starts at the first element, and moves down through the list until it is left pointing at the last element.

We will see how this can be made use of later in the course.

# Array Handling Functions

There is a range of functions to help you with processing data held in arrays:

| | |
|---|---|
| array_diff( $arr1, $arr2 ) | returns array containing all values of $arr1 not in $arr2 |
| array_flip( $arr1 ) | returns array which has keys as values and values as keys |
| array_intersect( $arr1, $arr2 ) | returns array containing all values of $arr1 also in $arr2 |
| array_keys( $arr1 ) | returns an array containing all of the keys in $arr1 |
| array_merge( $arr1, $arr2) | returns an array containing all values in $arr1 and $arr2 |
| array_rand( $arr1 ) | returns a value randomly selected from $arr1 |
| extract( $arr1 ) | converts elements in an array into variables in their own right |
| explode( separator, $arr1) | converts a string of values into an array |
| implode(separator, $arr1) | converts an array into a string, by inserting a separator |

These are not the only functions – there are many others.