Dr.SNS RAJALAKSHMI COLLEGE OF ARTS AND SCIENCE

(Autonomous)

Coimbatore -641049

Accredited by NAAC (Cycle- III)with 'Á+' Grade

(Recognised by UGC, Approved by AICTE, New Delhi and

Affiliated to Bharathiar University , Coimbatore

PHP LECTURE NOTES: UNIT-II

PREPARED BY

Dr.A.DEVI

Associate Professor

Department of Computer Applications

DR.SNSRACS

UNIT -II

1. Arrays
2. Functions

Working with ArraysPHP
Indexed Arrays

- There are two ways to create indexed arrays:
- The index can be assigned automatically (index always starts at 0), like this:
- $cars = array("Volvo", "BMW", "Toyota");
- or the index can be assigned manually:
- $cars[0] = "Volvo";
  $cars[1] = "BMW";
  $cars[2] = "Toyota";
- The following example creates an indexed array named $cars, assigns three elements to it, and then prints a text containing the array values:

Example

```
<html>
<body>
<?php
$cars = array("Volvo", "BMW", "Toyota");
echo "I like " . $cars[0] . ", " . $cars[1] . " and " . $cars[2] . ".";
?>
</body>
</html>
```

OUTPUT:

I like Volvo, BMW and Toyota.

Get The Length of an Array - The count() Function

The count() function is used to return the length (the number of elements) of an array:

Example

```
<html>
<body>
<?php
$cars = array("Volvo", "BMW", "Toyota");
echo count($cars);
?>
</body>
</html>
```

OUTPUT:

3

Loop Through an Indexed Array

To loop through and print all the values of an indexed array, you could use a for loop, like this:

Example

```
<html>
<body>
<?php
$cars = array("Volvo", "BMW", "Toyota");
$arrlength = count($cars);
for($x = 0; $x < $arrlength; $x++) {
    echo $cars[$x];
    echo "<br>";
}
?>
</body>
</html>
```

OUTPUT:

Volvo
BMW
Toyota

PHP Associative Arrays

- Associative arrays are arrays that use named keys that you assign to them.
- There are two ways to create an associative array:

```
$age = array("Peter"=>"35", "Ben"=>"37", "Joe"=>"43");or
$age['Peter'] = "35";
$age['Ben'] = "37";
$age['Joe'] = "43";
```

- The named keys can then be used in a script:

Example

```
<html>
<body>
<?php
$age = array("Peter"=>"35", "Ben"=>"37", "Joe"=>"43");
echo "Peter is " . $age['Peter'] . " years old.";
?>
</body>
</html>
```

OUTPUT:

Peter is 35 years old.

Loop through an Associative Array

To loop through and print all the values of an associative array, you could use a for each loop, like this:

Example

```
<?php
$age = array("Peter"=>"35", "Ben"=>"37", "Joe"=>"43");
```

```
foreach($age as $x => $x_value)
{
   echo "Key=" . $x . ", Value=" . $x_value;
   echo "<br>";
}
?>
```

Sorting Arrays

The elements in an array can be sorted in alphabetical or numerical order, descending or ascending.

- sort() - sort arrays in ascending order
- rsort() - sort arrays in descending order
- asort() - sort associative arrays in ascending order, according to the value
- ksort() - sort associative arrays in ascending order, according to the key
- arsort() - sort associative arrays in descending order, according to the value
- krsort() - sort associative arrays in descending order, according to the key

Sort Array in Ascending Order - sort()

The following example sorts the elements of the $cars array in ascending alphabetical order:

Example

```
<html>
<body>
<?php
$cars = array("Volvo", "BMW", "Toyota");
sort($cars);

$clength = count($cars);
for($x = 0; $x < $clength; $x++) {
   echo $cars[$x];
   echo "<br>";
}
?>
</body>
</html>
```

OUTPUT:

BMW
Toyota
Volvo

The following example sorts the elements of the $numbers array in ascending numerical order:

Example

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

OUTPUT:

```
2
4
6
11
22
```

Sort Array (Ascending Order), According to Value - asort()

The following example sorts an associative array in ascending order, according to thevalue:

Example

```
<?php
$age = array("Peter"=>"35", "Ben"=>"37", "Joe"=>"43");
asort($age);
?>
```

OUTPUT:

```
Key=Peter, Value=35
Key=Ben, Value=37
Key=Joe, Value=43
```

Sort Array (Ascending Order), According to Key - ksort()

The following example sorts an associative array in ascending order, according to thekey:

Example

```
<?php
$age = array("Peter"=>"35", "Ben"=>"37", "Joe"=>"43");
ksort($age);
?>
```

OUTPUT:

```
Key=Ben, Value=37
Key=Joe, Value=43
Key=Peter, Value=35
```

Sort Array (Descending Order), According to Value - arsort()

The following example sorts an associative array in descending order, according to thevalue:

Example

```
<?php
$age = array("Peter"=>"35", "Ben"=>"37", "Joe"=>"43");
arsort($age);
?>
```

OUTPUT:

Key=Joe, Value=43 Key=Ben, Value=37 Key=Peter, Value=35

Sort Array (Descending Order), According to Key - krsort()

The following example sorts an associative array in descending order, according to thekey:

Example

```
<?php
$age = array("Peter"=>"35", "Ben"=>"37", "Joe"=>"43");
krsort($age);
?>
```

OUTPUT:

Key=Peter, Value=35Key=Joe, Value=43 Key=Ben, Value=37

- An array stores multiple values in one single variable.
- In the following example $cars is an array. The PHP var_dump() function returns the data type and value:

Example

```
<html>
<body>
<?php
$cars = array("Volvo","BMW","Toyota");
var_dump($cars);
?>
</body>
</html>
```

OUTPUT:

array(3) { [0]=> string(5) "Volvo" [1]=> string(3) "BMW" [2]=> string(6) "Toyota" }

Functions

- The real power of PHP comes from its functions; it has more than 1000 built-in functions.

PHP User Defined Functions

- Besides the built-in PHP functions, we can create our own functions.
- A function is a block of statements that can be used repeatedly in a program.
- A function will not execute immediately when a page loads.
- A function will be executed by a call to the function.

Create a User Defined Function in PHP

A user-defined function declaration starts with the word function:

Syntax

function *functionName*() {
code to be executed;
}

- Function names are NOT case-sensitive.

In the example below, we create a function named "writeMsg()". The opening curly brace ({) indicates the beginning of the function code and the closing curly brace (}) indicates the end of the function. The function outputs "Hello world!". To call thefunction, just write its name:

Example

```
<html>
<body>
<?php
function writeMsg()
echo "Hello world!";
}
writeMsg();
?>
</body>
</html>
```

OUTPUT:

Hello world!

PHP Function Arguments

- Information can be passed to functions through arguments. An argument is just like a variable.
- Arguments are specified after the function name, inside the parentheses. You can add as many arguments as you want, just separate them with a comma.
- The following example has a function with one argument ($fname). When the familyName() function is called, we also pass along a name (e.g. Jani), and the name is used inside the function, which outputs several different first names, but an equal last name:

Example

```
<html>
<body>
<?php
function familyName($fname) {
   echo "$fname Refsnes.<br>";
}
familyName("Jani");
familyName("Hege");
familyName("Stale");
familyName("Kai Jim");
familyName("Borge");
?> </body> </html>
```

OUTPUT:

Jani Refsnes.
HegeRefsnes.
Stale Refsnes.
Kai Jim Refsnes.
Borge Refsnes.

The following example has a function with two arguments ($fname and $year):

Example

```
<html>
<body>
<?php
function familyName($fname, $year)
 {
   echo "$fname Refsnes. Born in $year <br>";
}
familyName("Hege","1975");
familyName("Stale","1978");
familyName("Kai
Jim","1983"); ?>
</body>
</html>
```

OUTPUT:

Hege Refsnes. Born in 1975
Stale Refsnes. Born in 1978 Kai
Jim Refsnes. Born in 1983

PHP Default Argument Value

- The following example shows how to use a default parameter. If we call the function setHeight() without arguments it takes the default value as argument:

Example

```
<html>
<body>
<?php
function setHeight($minheight = 50)
 {
echo "The height is : $minheight <br>";
}
setHeight(350);
setHeight();
setHeight(135);
setHeight(80);
?>
</body>
</html>
```

OUTPUT:

The height is : 350
The height is : 50
The height is : 135
The height is : 80

PHP Functions - Returning values

- To let a function return a value, use the return statement:

Example

```
<html>
<body>
<?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);
?>
</body>
</html>
```

OUTPUT:

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