



Web Technologies and Programming

Lecture 23

Introduction to PHP (Part-2)

Summary of Previous Lecture

- **Setting the environment**
- **PHP overview**
 - What is a PHP File
 - Open-source
 - Platform independent
 - What Can PHP Do?
 - Why PHP?
 - Basic PHP Syntax
 - Writing and Executing PHP Code
- **PHP constants**
 - Constants are Global
- **PHP variables**
 - Local
 - Global
 - Static
 - Type Determination
- **PHP Strings**
- **PHP is a Loosely Typed Language**

Today's Lecture Outline

- **Operators in PHP**
- **Conditional Statements in PHP**
- **Looping Statements**
- **Arrays in PHP**

1. Operators in PHP

- **Operators are used** to perform operations on variables and values.
- **PHP divides the operators in the following groups:**
 - Arithmetic operators
 - Assignment operators
 - String operators
 - Increment/Decrement operators
 - Logical operators
 - Comparison operators
 - Equality Operators
 - Array operators

1. Operators in PHP

- **Arithmetic Operators:**
 - The PHP arithmetic operators are used with numeric values to perform common arithmetical operations, **such as** addition, subtraction, multiplication etc.
 - **+, -, *, /, %, ****

1. Operators in PHP

- **Assignment Operators:**
- **The PHP assignment operators** are used with numeric values to write a value to a variable.
- **The basic assignment operator** in PHP is "=". It means that the left operand gets set to the value of the assignment expression on the right.

– =

– += (\$a += \$b), *= , /=

– .= (\$a .= \$b)

1. Operators in PHP

Assignment	Same as...	Description
<code>x = y</code>	<code>x = y</code>	The left operand gets set to the value of the expression on the right
<code>x += y</code>	<code>x = x + y</code>	Addition
<code>x -= y</code>	<code>x = x - y</code>	Subtraction
<code>x *= y</code>	<code>x = x * y</code>	Multiplication
<code>x /= y</code>	<code>x = x / y</code>	Division
<code>x %= y</code>	<code>x = x % y</code>	Modulus

1. Operators in PHP...

- **String Operators:**

- **PHP has two operators** that are specially designed for strings.

- **. , .=**

- **\$a="abcd"."efgh";** → **\$a=abcdefgh**

- **\$a.="ijk";** **\$a=abcdefghijk**



1. Operators in PHP...

- **String Operators:**

Operator	Name	Example	Result
<u>.</u>	<u>Concatenation</u>	<u>\$txt1</u> <u>.</u> <u>\$txt2</u>	Concatenation of \$txt1 and \$txt2
<u>.=</u>	<u>Concatenation assignment</u>	<u>\$txt1</u> <u>.=</u> <u>\$txt2</u>	Appends \$txt2 to \$txt1

1. Operators in PHP...

First Variable

Second Variable

```
<?php
```

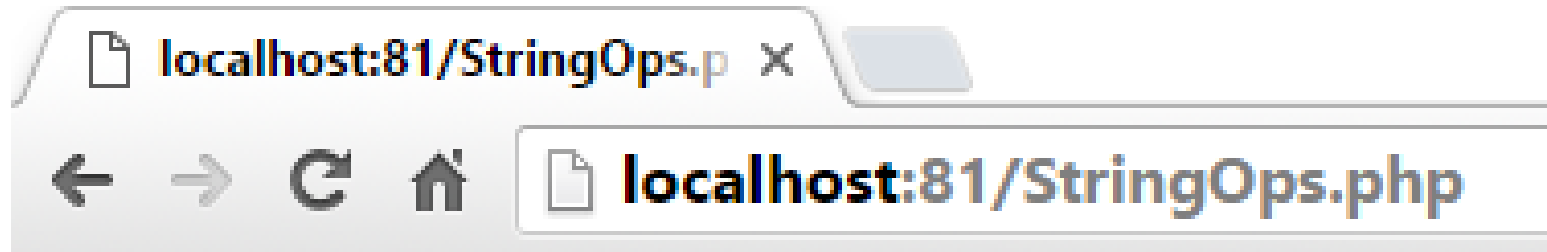
```
echo "<h1>String Operations!</h1>";  
$firstName = "Tehseen";  
$lastName = "Abbasi";  
echo $fullName = $firstName." ".$lastName;  
echo "<br>";  
echo $firstName.="! Welcome!";
```

Concatenation

```
?>
```

Using .=

1. Operators in PHP...



String Operations!

Tehseen Abbasi

Tehseen! Welcome!

1. Operators in PHP...

```
<?php
```

```
echo "<h1>Addition & Concatenation!</h1>";
```

```
$a = 10;
```

```
$b = 10.5;
```

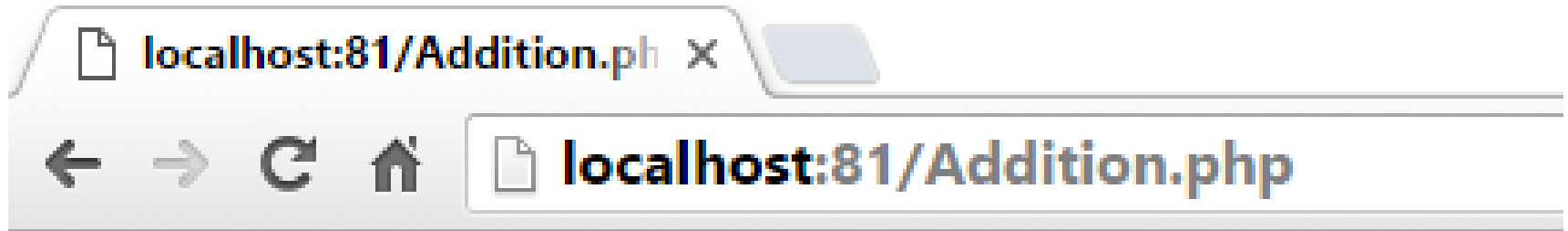
```
echo $a += $b; ← Adds $b in $a
```

```
echo "<br>";
```

```
echo $a .= $b; ← Concatenates $b with $a
```

```
?>
```

1. Operators in PHP...



Addition & Concatenation!

20.5

20.510.5

1. Operators in PHP...

- **Increment/decrement Operators:**
- **The PHP increment** operators are used to increment a variable's value.
- **The PHP decrement** operators are used to decrement a variable's value.

++ , --

- \$b=\$a++
- \$b=++\$a

1. Operators in PHP...

- **Increment/decrement Operators:**

Operator	Name	Description
<code>++\$x</code>	Pre-increment	Increments <code>\$x</code> by one, then returns <code>\$x</code>
<code>\$x++</code>	Post-increment	Returns <code>\$x</code> , then increments <code>\$x</code> by one
<code>--\$x</code>	Pre-decrement	Decrements <code>\$x</code> by one, then returns <code>\$x</code>
<code>\$x--</code>	Post-decrement	Returns <code>\$x</code> , then decrements <code>\$x</code> by one

1. Operators in PHP...

```
<?php
```

```
echo "<h1>Increment Techniques!</h1>";
```

```
$a = 10; ← Variable Declared
```

```
echo ++$a; ← Incremented Before Display
```

```
echo "<br>";
```

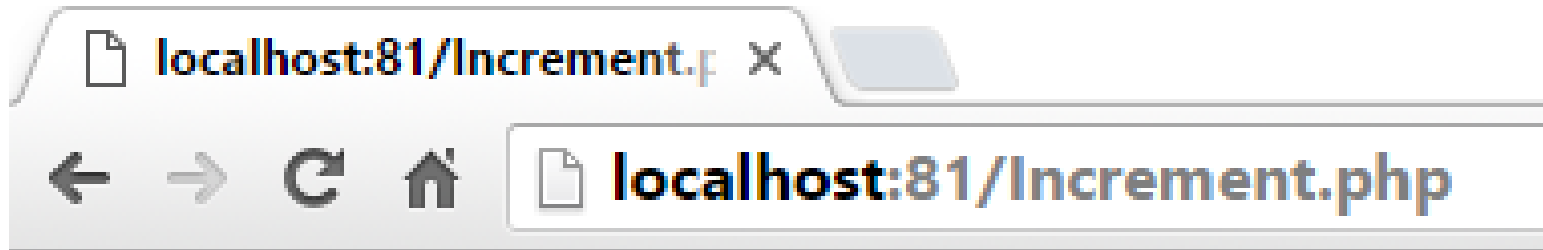
```
echo $a++; ← Incremented After Display
```

```
echo "<br>";
```

```
echo $a; ← Displaying Incremented Value
```

```
?>
```

1. Operators in PHP...



Increment Techniques!

Handwritten notes illustrating increment techniques:

$a = 10$

11
11
12

$a = 1 + 10 = 11$

$a++$

$a = a + 1$
 $a = 11 + 1$
 $a = 12$

Arrows indicate the flow of the increment process from the initial value 10 to the final value 12.

1. Operators in PHP...

- **Logical Operators:**
 - The PHP logical operators are used to combine conditional statements.
 - **AND, OR, NOT, XOR**
 - **&&, ||, !**

1. Operators in PHP...

- **Logical Operators:**

Operator	Name	Example	Result
and	And	<u>\$x</u> and <u>\$y</u>	True if both \$x and \$y are true
or	Or	<u>\$x</u> or <u>\$y</u>	True if either \$x or \$y is true
xor	Xor	\$x xor \$y	True if either \$x or \$y is true, but not both
&&	And	\$x && \$y	True if both \$x and \$y are true
	Or	\$x \$y	True if either \$x or \$y is true
!	Not	!\$x	True if \$x is not true

1. Operators in PHP...

- **Comparison Operators:**

- The PHP comparison operators are used to compare two values (number or string):
- **>, <, <=, >=**

1. Operators in PHP...

- **Comparison Operators:**

Operator	Name	Example	Result
<u>==</u>	Equal	<code>\$x == \$y</code>	Returns true if \$x is equal to \$y
<code>===</code>	Identical	<code>\$x === \$y</code>	Returns true if \$x is equal to \$y, and they are of the same type
<code>!=</code>	Not equal	<code>\$x != \$y</code>	Returns true if \$x is not equal to \$y
<code><></code>	Not equal	<code>\$x <> \$y</code>	Returns true if \$x is not equal to \$y
<code>!==</code>	Not identical	<code>\$x !== \$y</code>	Returns true if \$x is not equal to \$y, or they are not of the same type
<code>></code>	Greater than	<code>\$x > \$y</code>	Returns true if \$x is greater than \$y
<code><</code>	Less than	<code>\$x < \$y</code>	Returns true if \$x is less than \$y
<code>>=</code>	Greater than or equal to	<code>\$x >= \$y</code>	Returns true if \$x is greater than or equal to \$y
<code><=</code>	Less than or equal to	<code>\$x <= \$y</code>	Returns true if \$x is less than or equal to \$y

1. Operators in PHP...

- **Equality Operators:**

- ==, !=, ===

1. Operators in PHP...

```
<?php
```

```
echo "<h1>Equality!</h1>";
```

```
$a = 10; ← Integer Value
```

```
$b = "10"; ← String Value
```

```
if($a == $b)
```

```
{  
    echo "Equal";  
    echo "<br>";  
}
```

```
else
```

```
{  
    echo "Not Equal";  
    echo "<br>";  
}
```

Compares Only Values

```
if($a === $b)
```

```
{  
    echo "Equal";  
    echo "<br>";  
}
```

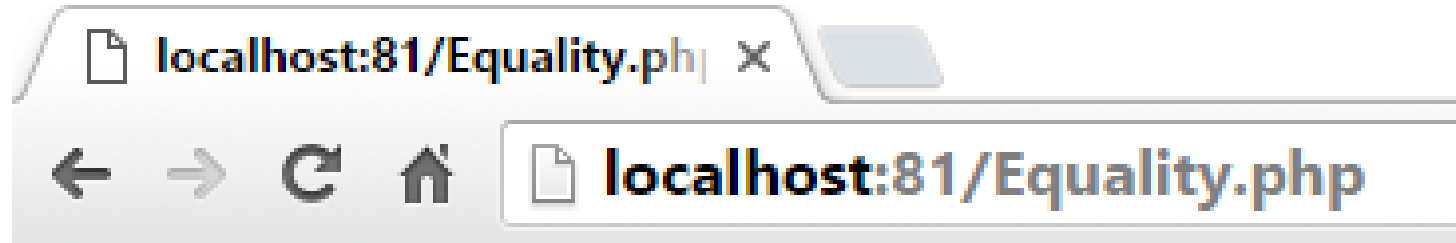
```
else
```

```
{  
    echo "Not Equal";  
    echo "<br>";  
}
```

Strict Comparison, Data Types Should Also Match

```
?>
```


1. Operators in PHP...



Equality!

Equal

Not Equal

2. Conditional Statements

- When you want to perform different actions for different conditions. You can use conditional statements in your code to do this.
- **In PHP we have the following conditional statements:**
 - **if statement** - executes some code if one condition is true
 - **if...else statement** - executes some code if a condition is true and another code if that condition is false
 - **if...elseif....else statement** - executes different codes for more than two conditions
 - **switch statement** - selects one of many blocks of code to be executed

2. Conditional Statements

- **if statement:**
- **The if statement executes some code if one condition is true.**

```
if(condition)
```

```
{
```

```
code to be executed if condition is true;
```

```
}
```

2. Conditional Statements

- **if statement Example:**
- ```
<?php
$t = date("H");

if ($t < "20")
{
 echo "Have a good day!";
}
?>\
```

Result:

Have a good day!

## 2. Conditional Statements

- **if-else statement:**
- The if....else statement executes some code if a condition is true and another code if that condition is false.

```
if (condition) {
 code to be executed if condition is true;
} else {
 code to be executed if condition is false;
}
```

## 2. Conditional Statements

- **if-else statement Example:**

- ```
<?php
$t = date("H");

if ($t < "20") {
    echo "Have a good day!";
}
else
{
    echo "Have a good night!";
}
?>
```

Result:

Have a good day!

2. Conditional Statements

- **The if...elseif...else Statement:**
- The if...elseif...else statement executes different codes for more than two conditions..

```
if (condition)
{
    code to be executed if this condition is true;
}
elseif (condition)
{
    code to be executed if this condition is true;
}
else
{
    code to be executed if all conditions are false;
}
```

2. Conditional Statements

- **The if...elseif...else statement Example:**

- ```
<?php
$t = date("H");

if ($t < "10") {
 echo "Have a good morning!";
} elseif ($t < "20") {
 echo "Have a good day!";
} else {
 echo "Have a good night!";
}
?>
```

Result:


The hour (of the server) is 05, and will give the following message:

Have a good morning!



## 2. Conditional Statements...

- **switch statement:**
- Use the switch statement to **select one of many blocks of code to be executed**

```
switch(variable)
{
 case option:
 action 
 break;
 .
 .
}
```

## 2. Conditional Statements...

- ~~switch (n) {~~
  - case label1:**  
*code to be executed if n=label1;*  
*break;*
  - case label2:**  
*code to be executed if n=label2;*  
*break;*
  - case label3:**  
*code to be executed if n=label3;*  
*break;*
  - ...**
  - default:**  
*code to be executed if n is different from all labels;*  
**}**

## 2. Conditional Statements...

```
<?php
echo "<h1>Switch Statement!</h1>";
\$a = 10;
\$opt = \$a%2;
switch($opt)
{
 case 0:
 echo $a." is Even";
 break;
 case 1:
 echo $a." is Odd";
 break;
}
```

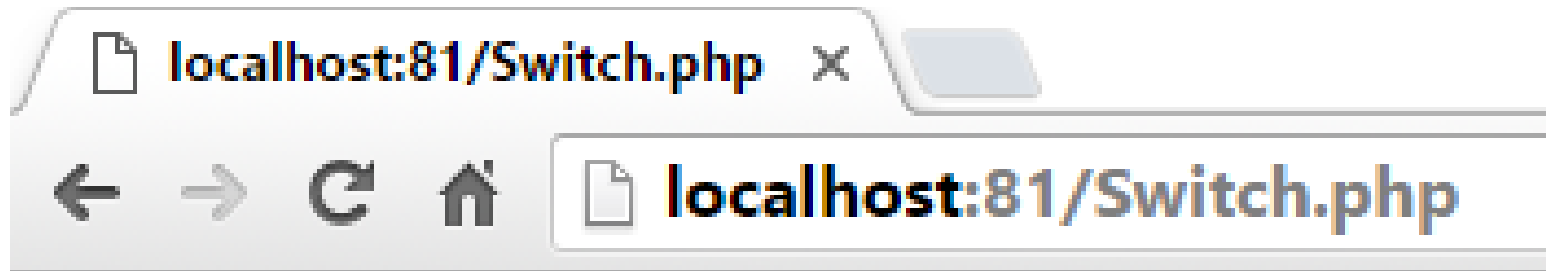
**Switch Starts** →

**Case 0**

**Case 1**

**?>**

## 2. Conditional Statements...



# Switch Statement!

10 is Even

# 3. Looping Statements

- **For Loop**
- **While Loop**
- **Do-While Loop**
- **ForEach Loop**

# 3. Looping Statements

- **for loop**
- The for loop is used when you know in advance how many times the script should run.

Syntax:

```
for (init counter; test counter; increment counter) {
 code to be executed;
}
```

```
.....
for($a=0; $a<10; $a++)
{
 //statements
}
```

# 3. Looping Statements

- **for loop Example:**

- `<!DOCTYPE html>`  
`<html>`  
`<body>`

```
<?php
for ($x = 0; $x <= 10; $x++) {
 echo "The number is: $x
";
}
?>
```

```
</body>
</html>
```

Result:

```
The number is: 0
The number is: 1
The number is: 2
The number is: 3
The number is: 4
The number is: 5
The number is: 6
The number is: 7
The number is: 8
The number is: 9
The number is: 10
```

# 3. Looping Statements

- **while loop**
- The while loop executes a block of code as long as the specified condition is true.

```
while(condition is true)
{
 //Statements
 //Increment/decrement
}
```



# 3. Looping Statements

- **while loop Example:**

```
<?php
$x = 1;

while($x <= 5) {
 echo "The number is: $x
";
 $x++;
}
?>
```

Result:

```
The number is: 1
The number is: 2
The number is: 3
The number is: 4
The number is: 5
```

# 3. Looping Statements...

- **do-while loop**
- **The do...while loop** will always execute the block of code once, it will then check the condition, and repeat the loop while the specified condition is true.

**do**

**{**

**//Statements**

**//Increment/decrement**

**}**

**While(condition is true);**

# 3. Looping Statements...

- **do-while loop Example:**

```
<?php
$x = 1;

do {
 echo "The number is: $x
";
 $x++;
} while ($x <= 5);
?>
```

Result:

```
The number is: 1
The number is: 2
The number is: 3
The number is: 4
The number is: 5
```

# 3. Looping Statements...

- foreach loop
- The foreach loop works only on arrays, and is used to loop through each key/value pair in an array
  - is used to read an entire array

```
foreach ($array as $value)
{
 code to be executed;
}
```

# 3. Looping Statements...

- foreach loop example:

- ```
<?php  
$colors = array("red", "green", "blue", "yellow");
```

```
foreach ($colors as $value) {  
    echo "$value <br>";  
}  
?>
```

Result:

```
red  
green  
blue  
yellow
```

3. Looping Statements...

```
<?php
```

```
echo "<h1>Loops!</h1>";
```

```
for($a=0; $a<5; $a++)
```

```
{
```

```
    echo "I Love To Work With PHP";
```

```
    echo "<br>";
```

```
}
```

For Loop

```
echo "<br>";
```

```
$b=0;
```

```
while($b<5){
```

```
    echo "That Is Good If You Can Work With Loops in PHP!";
```

```
    echo "<br>";
```

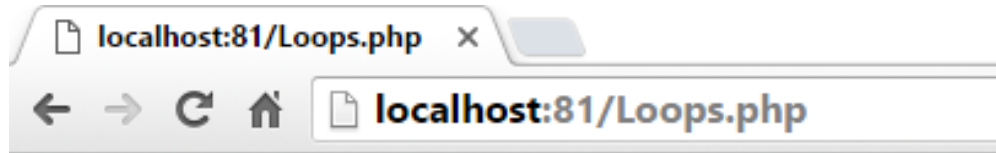
```
    $b++;
```

```
}
```

While Loop

```
?>
```

3. Looping Statements...



Loops!

I Love To Work With PHP
I Love To Work With PHP
I Love To Work With PHP
I Love To Work With PHP
I Love To Work With PHP

Output From For Loop

That Is Good If You Can Work With Loops in PHP!
That Is Good If You Can Work With Loops in PHP!
That Is Good If You Can Work With Loops in PHP!
That Is Good If You Can Work With Loops in PHP!
That Is Good If You Can Work With Loops in PHP!

Output From While Loop

3. Looping Statements...

Array Declaration

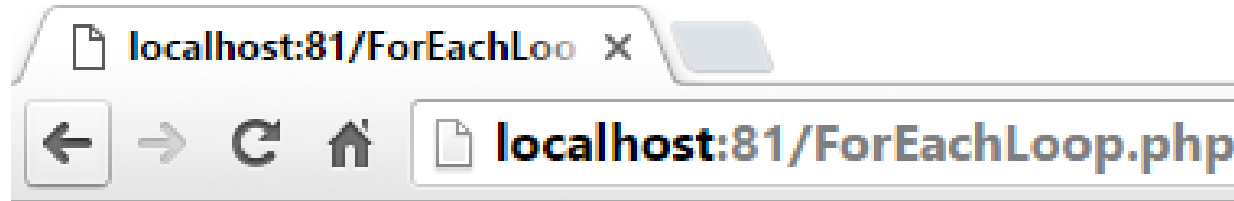
```
<?php
    echo "<h1>For Each Loop!</h1>";

    $Players = array("Shahid Khan Afridi", "Muhammad Amir", "AB de Villers");

    echo "<h3 style=\"color:blue\">My Favourite Players</h3>";

    foreach($Players as $Player) ← ForEach Loop Starts
    {
        echo $Player; ← Using Obtained Value
        echo "<br>";
    }
?>
```


3. Looping Statements...



For Each Loop!

My Favourite Players

Shahid Khan Afridi
Muhammad Amir
AB de Villers

4. Arrays in PHP

- An array stores multiple values in one single variable
- An array is a special variable, which can hold more than one value at a time.
- An array is traditionally defined as a **group of items** that **share** certain characteristics
- Each item consists of **two** components:
 - **Key**
 - **Value**
- PHP **doesn't** require that you assign a **size** to an array at creation time

4. Arrays in PHP...

- Declaring an array:
 - `$array_name[key] = value;`
 - `$players[0] = "Shahid Khan Afridi";`
- Adding element in an array:
 - `$players[1] = "Muhammad Amir";`
- Accessing element in an array
 - `echo $players[0];`

4. Arrays in PHP...

```
<?php
```

```
echo "<h1>Arrays!</h1>";
```

```
$Players[0] = "Shahid Khan Afridi";
```

 ← Declaring Array

```
$Players[4] = "Muhammad Amir";
```

 ← Adding Elements

```
$Players[2] = "AB de Villers";
```

```
echo "<h3 style=\"color:blue\">My Favourite Players</h3>";
```

```
foreach($Players as $Player)
```

```
{
```

```
    echo $Player;
```

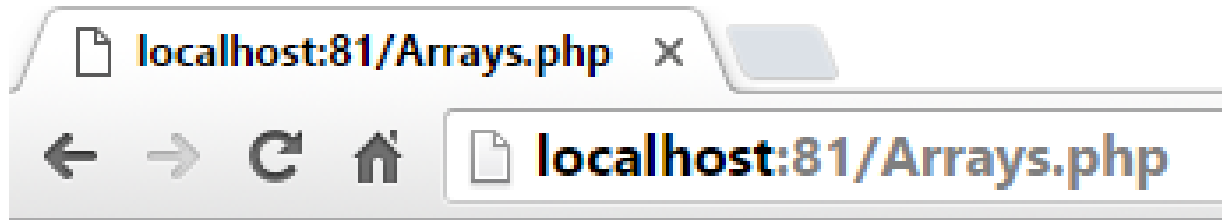
```
    echo "<br>";
```

```
}
```

ForEach Loop

```
?>
```

4. Arrays in PHP...



Arrays!

My Favourite Players

Shahid Khan Afridi

Muhammad Amir

AB de Villers

4. Arrays in PHP...

- **Associative arrays:** Arrays with named keys
 - `$array_name['element-name'] = value;`
 - `$players['shahid'] = "Shahid Khan Afridi";`
- **Adding element in an array:**
 - `$players['amir'] = "Muhammad Amir";`
- **Accessing element in an array:**
 - `echo $players['shahid'];`

4. Arrays in PHP...

- The **array()**; can also be used to create an array
 - `$array_name = array(item_1, item_2, ..., item_n);`
 - `$players = array("Shahid Khan Afridi", "Muhammad Amir");`
 - `$players = array("shahid"=>"Shahid Khan Afridi", "amir"=>"Muhammad Amir");`

4. Arrays in PHP...

Associative Array Declared Using array()

```
<?php
echo "<h1>Associative Array!</h1>";

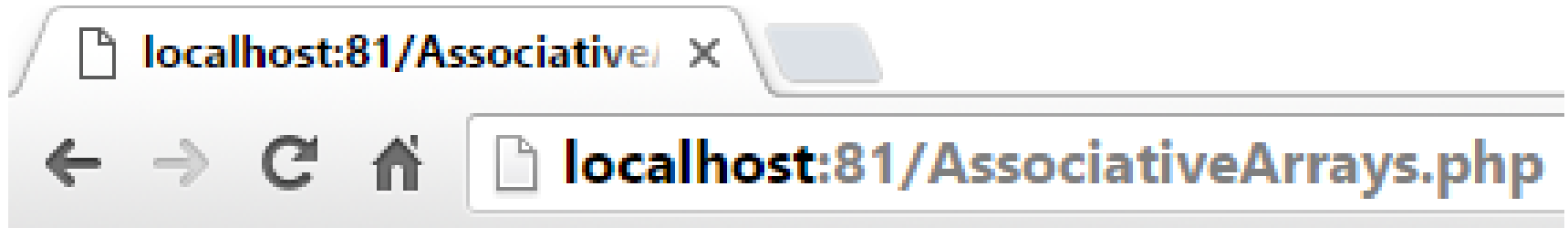
$Players = array('shahid' => "Shahid Khan Afridi", 'amir' => "Muhammad Amir");

echo "<h3 style=\"color:blue\">Players</h3>";

echo $Players['shahid']."<br>";
echo $Players['amir'];
```

Accessing Elements by Name

4. Arrays in PHP...



Associative Array!

Players

Shahid Khan Afridi

Muhammad Amir

4. Arrays in PHP...

- **Sorting arrays:**
 - **sort()**
 - Sorts the array in ascending order
 - **rsort()**
 - Sorts the array in descending order

4. Arrays in PHP...

Array Declaration

```
<?php
echo "<h1>Sort Function!</h1>";

$Players = array("Shahid Khan Afridi", "Muhammad Amir", "AB de Villers");
sort($Players);

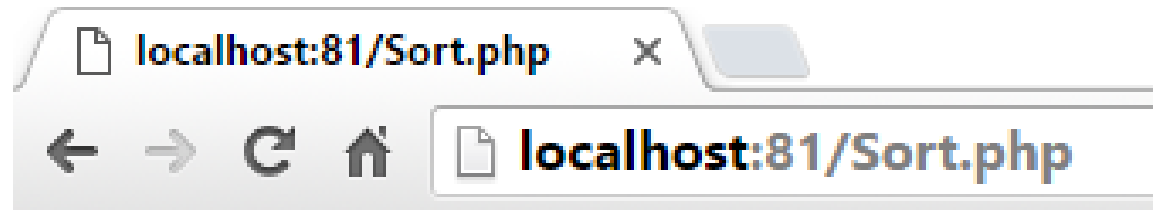
echo "<h3 style=\"color:blue\">My Favourite Players</h3>";

foreach($Players as $Player)
{
    echo $Player;
    echo "<br>";
}
?>
```

← Sorting Array

← ForEach Loop for Displaying Arrays

4. Arrays in PHP...



Sort Function!

My Favourite Players

AB de Villers

Muhammad Amir

Shahid Khan Afridi

4. Arrays in PHP...

```
<?php
echo "<h1>Reverse Sort Function!</h1>";

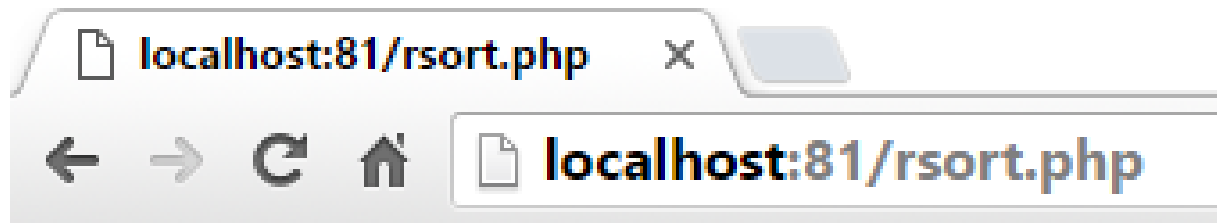
$Players = array("Shahid Khan Afridi", "Muhammad Amir", "AB de Villers");
rsort($Players); ← Reverse Sorting

echo "<h3 style=\"color:blue\">My Favourite Players</h3>";

foreach($Players as $Player)
{
    echo $Player;
    echo "<br>";
}
```

?>

4. Arrays in PHP...



R-Sort Function!

My Favourite Players

Shahid Khan Afridi

Muhammad Amir

AB de Villers

Summary of Today's Lecture

- **Operators in PHP**

- Arithmetic Operators: `+, -, *, /, %, **`
- Assignment Operators: `=`
- String Operators: `.`, `,`, `.=`
- Increment/decrement Operators: `++`, `--`
- Logical Operators: `AND`, `OR`, `NOT`, `XOR`, `&&`, `||`, `!`
- Comparison Operators: `>`, `<`, `<=`, `>=`
- Equality Operators: `==`, `!=`, `===`

- **Conditional statements**

- **if statement** - executes some code if one condition is true
- **if...else statement** - executes some code if a condition is true and another code if that condition is false
- **if...elseif....else statement** - executes different codes for more than two conditions
- **switch statement** - selects one of many blocks of code to be executed

Summary of Today's Lecture

- **Looping statements**
 - For Loop
 - While Loop
 - Do-While Loop
 - ForEach Loop
- **Arrays in PHP**
 - Associative arrays
 - Sorting arrays

THANK YOU
