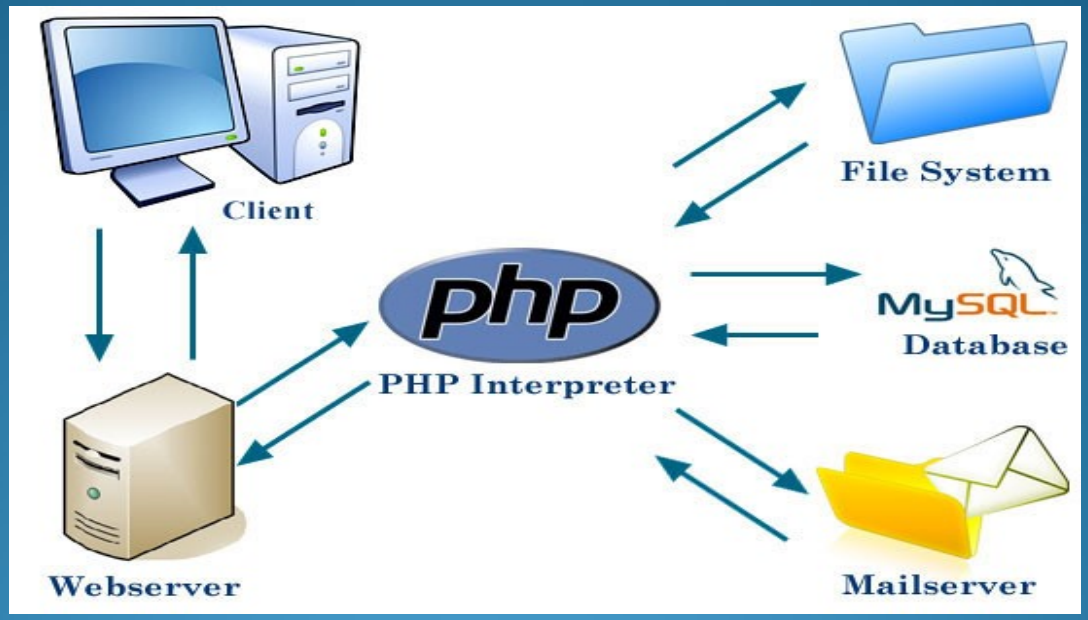
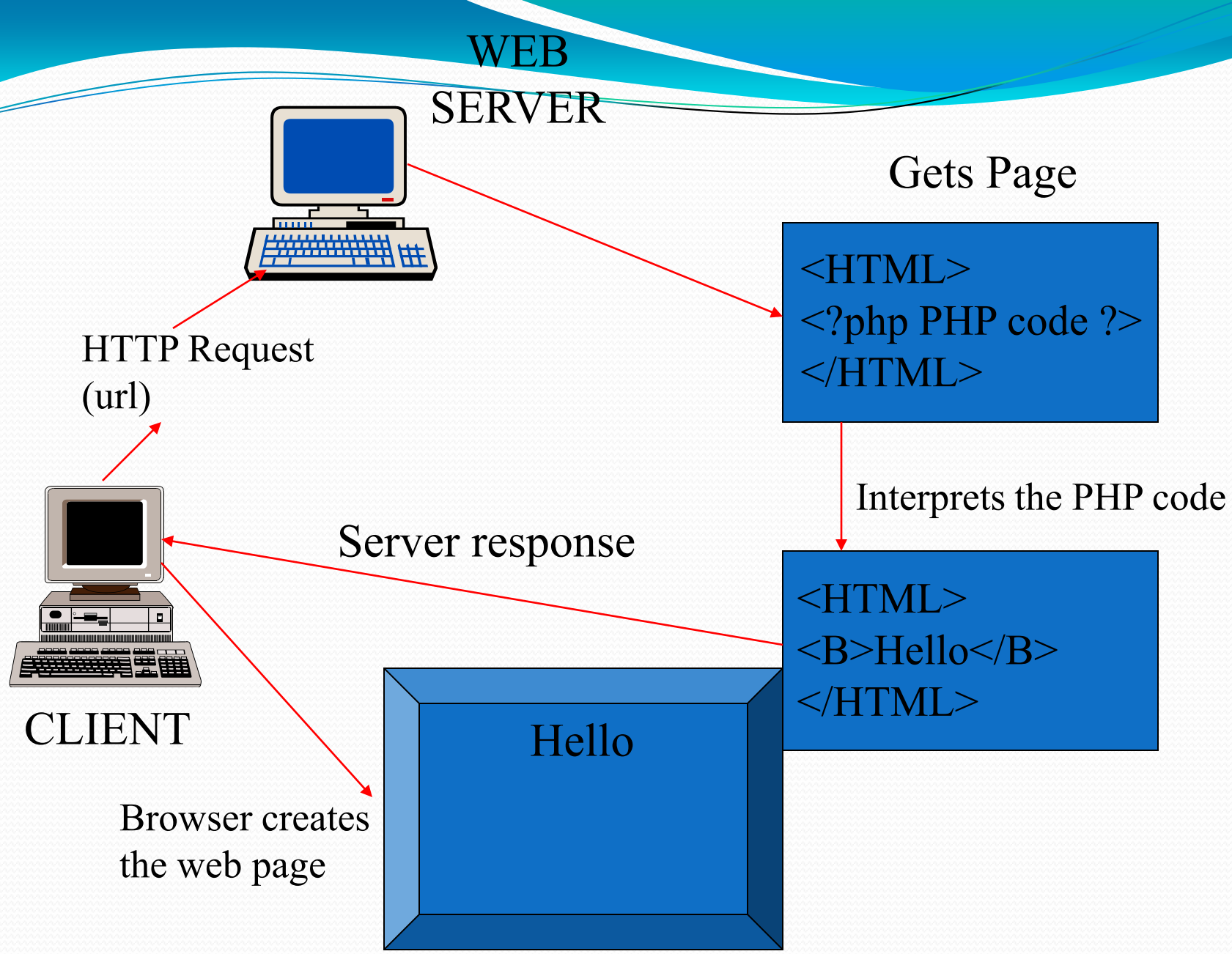
PHP lab 1



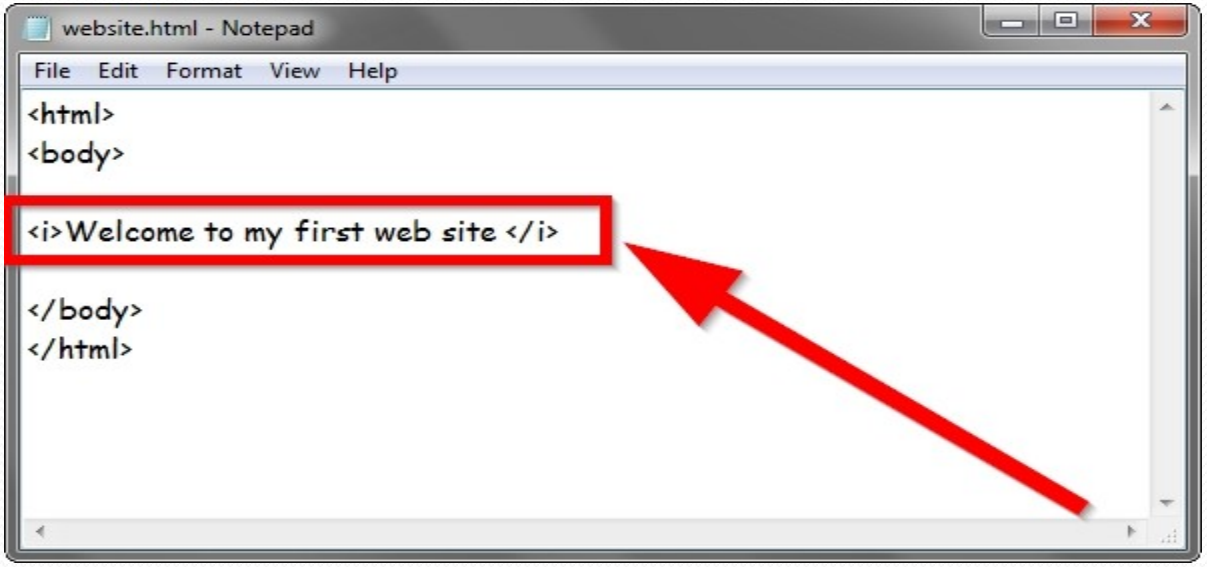
Samar Abdelghani **Demonstrator, CS Department,**


WEB
SERVER
Gets Page
<HTML>
<?php PHP code ?>
</HTML>
HTTP Request
(url)
Interprets the PHP code
Server response
<HTML>
<B>Hello</B>
</HTML>
CLIENT
Hello
Browser creates
the web page

Used Technologies

HTML

Hypertext Markup Language, a standardized system for tagging text files to achieve font, colour, graphic, and hyperlink effects on World Wide Web pages.



Used Technologies

CCS

1. Cascading Style Sheets (CSS): style sheet language used to describe the presentation of a html document.
2. Define colors, fonts, layout, and other aspects of document
3. Why CSS?
   - more flexibility
   - control the specification of presentational characteristics
   - reduce complexity and repetition in the structural content.

Used Technologies

Javascript

1. JavaScript is a scripting language most often used for client-side.
2. JS functions are embedded in HTML pages and interact with the Document Object Model (DOM) of the page
3. Respond to user actions quickly, making an application feel more responsive
4. Detect user actions which HTML alone cannot

Used Technologies

PHP

Personal Home Page (PHP)

1. PHP recursive acronym for "PHP:Hypertext Preprocessor“
2. Widely-used Open Source scripting language
3. Especially suited for Web development
4. Used for producing dynamic web pages
5. Can be embedded into HTML.

Used Technologies

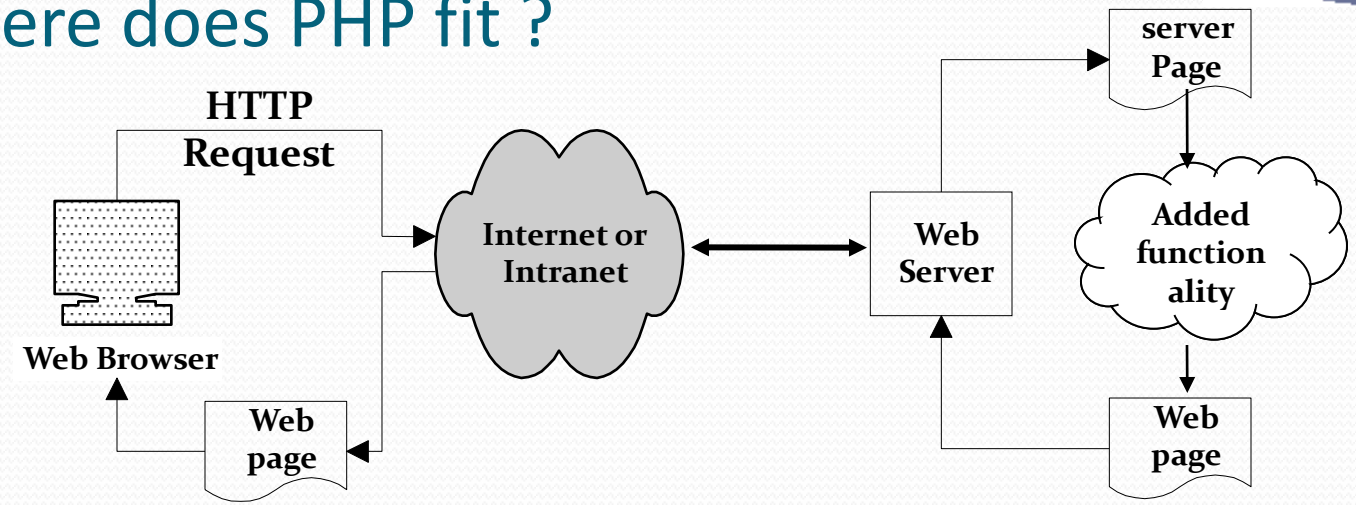
PHP

Hypertext Preprocessor

1. PHP includes a large number of free and open source libraries
2. Real Object Oriented Programming Language
3. Commonly install on Apache Server allow to interact with Database like Mysql
4. PHP is mainly focused on server-side scripting

Used Technologies

Where does PHP fit ?



Client-side
"Active pages"

JavaScript, VBScript,
Applet, ActiveX

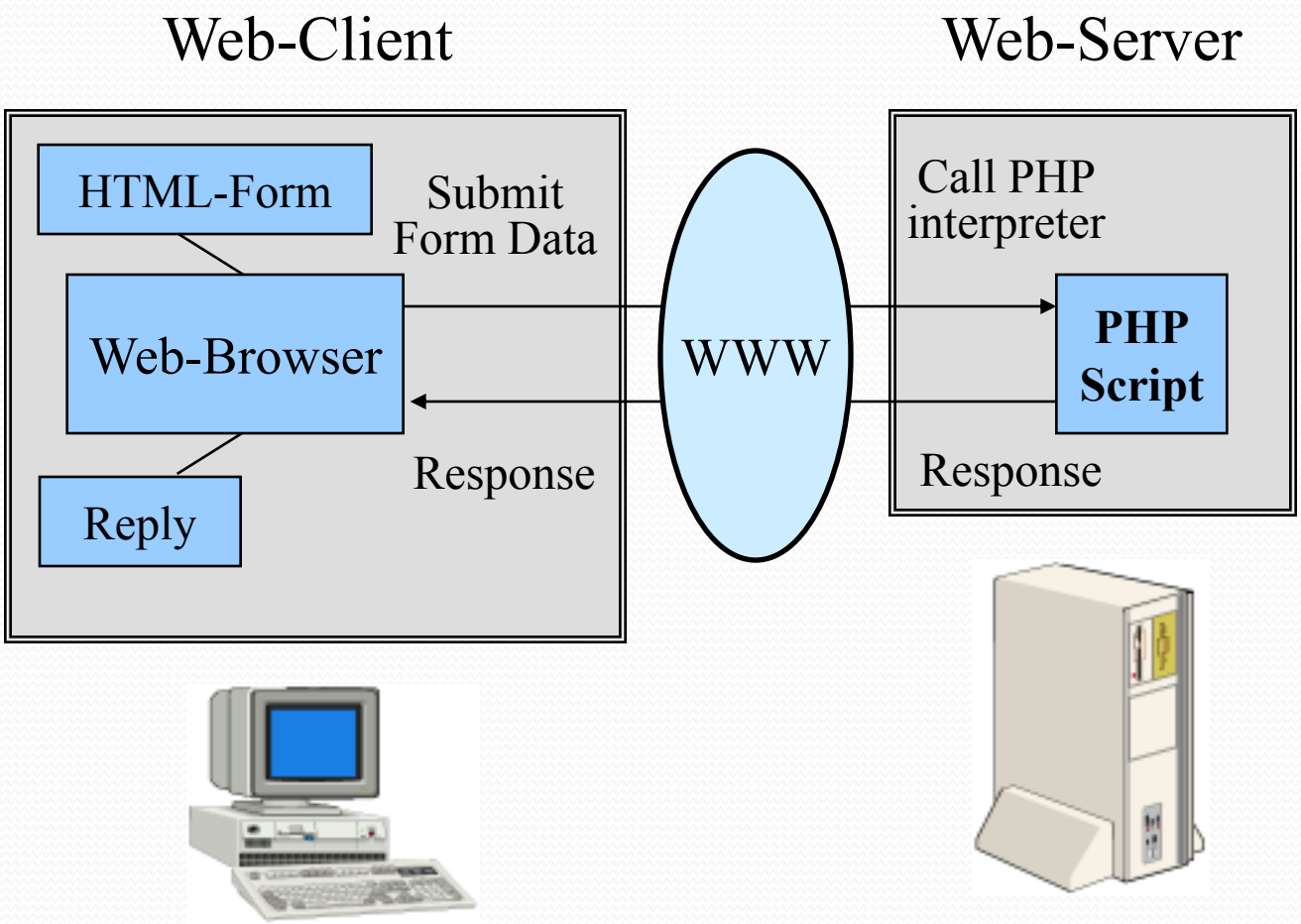
Server-side
"Dynamic pages"

CGI, SSI, Server
API, ASP, JSP, PHP,
COM/DCOM,
CORBA

Active and dynamic page technology can be used together – server-side program generates customized active pages.

Used Technologies

PHP - Communication client-server



PHP generally runs on a web server, taking PHP code as its input and creating Web pages as output

PHP vs. JSP

- PHP is faster in execution time
- A recent survey in ZDnet's *eWeek* online publication found that PHP is as much as 3.5 times faster than JSP
- Faster in development time – flatter learning curve
- PHP supports any 32-bit or better platform, whereas JSP supports only platforms that have a Java virtual machine available

PHP vs. ASP

- PHP is faster
- Superior Memory Management
- Closer to C Style of Programming
- Cross Platform Migration Strategy
- Dynamic generation of UI is more flexible

Used Technologies

MySQL

1. MySQL is a multithreaded, multi-user SQL database management system.
2. Popular for web applications.
3. Closely tied to PHP.
4. Allow all sort of queries.
5. PhpMyAdmin: friendly user interface to manage database develop on PHP.

What we'll cover

- PHP Introduction
- PHP Variables.
- PHP Comments.
- PHP Operators.
- PHP Conditional Statements.
- PHP Loops.
- PHP Arrays
- PHP Functions

PHP Introduction

- PHP == ‘Hypertext Preprocessor’
- Open-source, server-side scripting language
- Used to generate dynamic web-pages
- PHP scripts reside between reserved PHP tags
  - This allows the programmer to embed PHP scripts within HTML pages

PHP Introduction

- > PHP is a server-side scripting language
- > PHP scripts are executed on the server
- > PHP supports many databases (MySQL, Informix, Oracle, Sybase, Solid, PostgreSQL, Generic ODBC, etc.)
- > PHP is open source software
- > PHP is free to download and use

PHP Introduction

- > PHP runs on different platforms (Windows, Linux, Unix, etc.)
- > PHP is compatible with almost all servers used today (Apache, IIS, etc.)
- > PHP is FREE to download from the official PHP resource: www.php.net
- > PHP is easy to learn and runs efficiently on the server side

PHP Introduction

- Some info on MySQL which we will cover in the next workshop…

- > MySQL is a database server
- > MySQL is ideal for both small and large applications
- > MySQL supports standard SQL
- > MySQL compiles on a number of platforms
- > MySQL is free to download and use

PHP Introduction

What does PHP code look like?

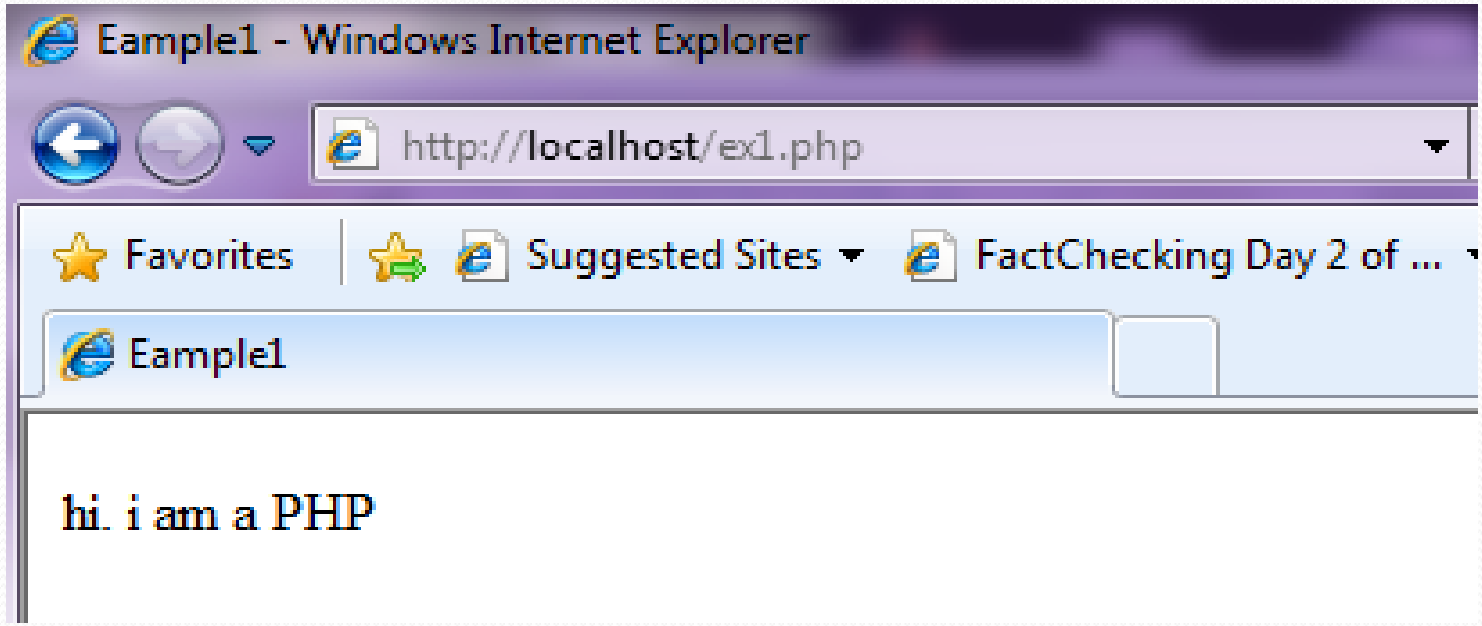
- Structurally similar to C/C++
- Supports procedural and object-oriented paradigm (to some degree)
- All PHP statements end with a semi-colon
- Each PHP script must be enclosed in the reserved PHP tag

```
<?php
    …
?>
```

PHP Introduction

```
1
2
3   <html>
4       <head>
5   <title> Eample1</title>
6       </head>
7       <body>
8       <?php
9   //this is acomment1
10  #this is acomment2
11
12  echo " hi. i am a PHP";
13      ?>
14       </body>
15  </html>
```

PHP Introduction

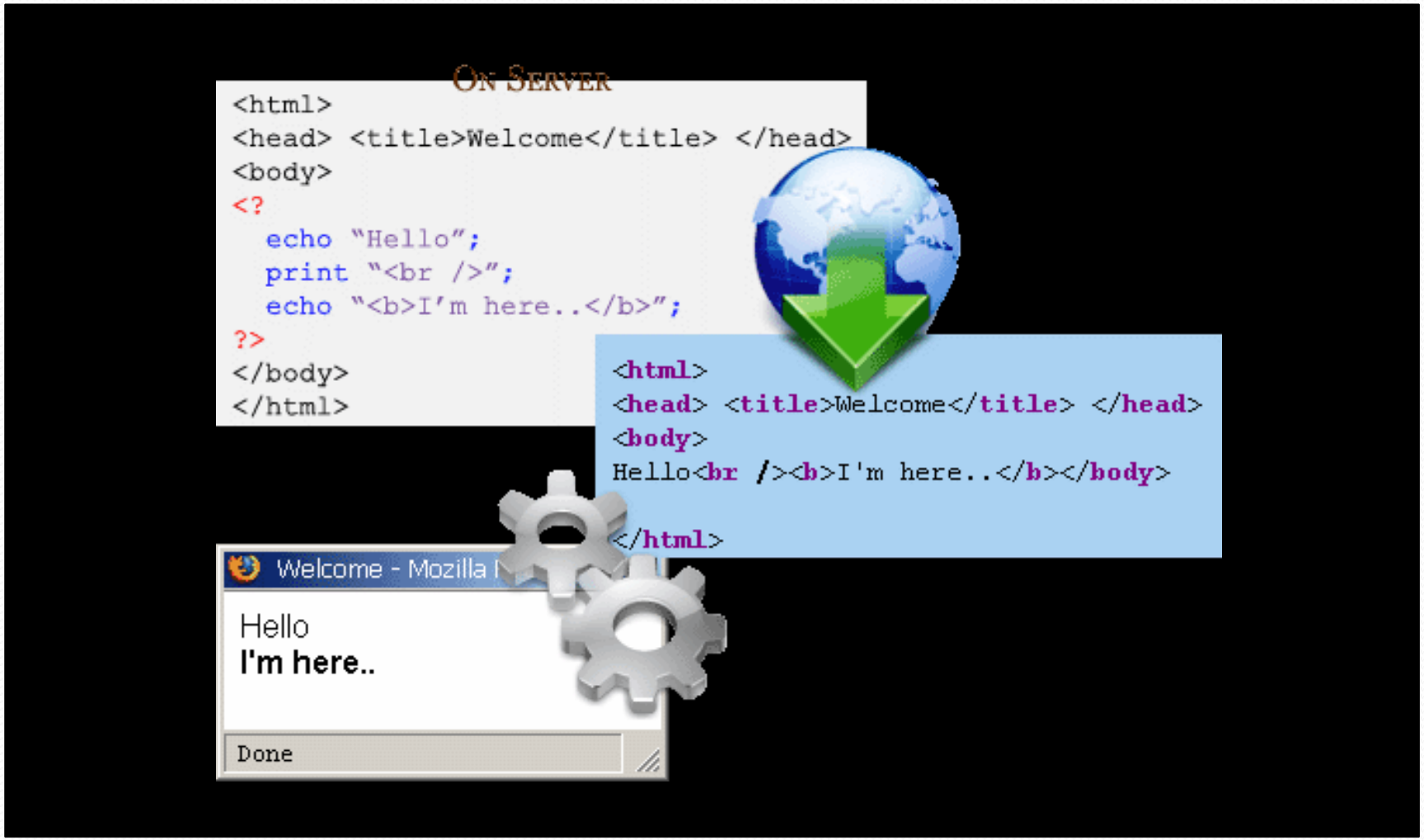


PHP Introduction

- PHP code is executed on the server, generating HTML which is then sent to the client. The client would receive the results of running that script, but would not know what the underlying code was.

- A visual, if you please…

PHP Introduction



PHP Getting Started

- On windows, you can download and install WAMP. With one installation and you get an Apache webserver, database server and php.
- http://www.wampserver.com

- On mac, you can download and install MAMP.
- http://www.mamp.info/en/index.html

PHP Hello World

```
<html>
 <head>
  <title>PHP Test</title>
 </head>
 <body>
 <?php echo '<p>Hello World</p>'; ?>
 </body>
</html>
```

- Above is the PHP source code.

PHP Hello World

- It renders as HTML that looks like this:

```
<html>
 <head>
  <title>PHP Test</title>
 </head>
 <body>
 <p>Hello World</p>
 </body>
</html>
```

PHP Hello World

- This program is extremely simple and you really did not need to use PHP to create a page like this. All it does is display: Hello World using the PHP **echo()** statement.

- Think of this as a normal HTML file which happens to have a set of special tags available to you that do a lot of interesting things.

PHP Comments

- In PHP, we use // to make a single-line comment or /* and */ to make a large comment block.

```
<html>
<body>

<?php
//This is a comment

/*
This is
a comment
block
*/
?>

</body>
</html>
```

PHP Variables

- Variables are used for storing values, like text strings, numbers or arrays.
- When a variable is declared, it can be used over and over again in your script.
- All variables in PHP start with a “$” sign symbol.
- Case-sensitive ($Foo != $foo != $fOo)
- The correct way of declaring a variable in PHP:

```
$var_name = value;
```

PHP Variables

```
<?php
$txt="Hello World!";
$x=16;
?>
```

- In PHP, a variable does not need to be declared before adding a value to it.
- In the example above, you see that you do not have to tell PHP which data type the variable is.
- PHP automatically converts the variable to the correct data type, depending on its value.

PHP Variables

- > A variable name must start with a letter or an underscore "_" -- not a number
- > A variable name can only contain alpha-numeric characters, underscores (a-z, A-Z, 0-9, and _)
- > A variable name should not contain spaces. If a variable name is more than one word, it should be separated with an underscore ($my_string) or with capitalization ($myString)

PHP Concatenation

- > The concatenation operator (.) is used to put two string values together.
- > To concatenate two string variables together, use the concatenation operator:

```
<?php
$txt1="Hello World!";
$txt2="What a nice day!";
echo $txt1 . " " . $txt2;
?>
```

PHP Concatenation

- The output of the code on the last slide will be:

```
Hello World! What a nice day!
```

- If we look at the code you see that we used the concatenation operator two times. This is because we had to insert a third string (a space character), to separate the two strings.

Echo

- The PHP command ‘**echo**’ is used to output the parameters passed to it
  - The typical usage for this is to send data to the client’s web-browser

Echo example

```
<?php
$foo = 25;              // Numerical variable
$bar = "Hello";         // String variable

echo $bar;              // Outputs Hello
echo $foo,$bar;         // Outputs 25Hello
echo "5x5=",$foo;       // Outputs 5x5=25
echo "5x5=$foo";        // Outputs 5x5=25
echo '5x5=$foo';        // Outputs 5x5=$foo
?>
```

- **Notice** how echo '5x5=$foo' outputs $foo rather than replacing it with 25
- Strings in single quotes (' ') are not interpreted or evaluated by PHP
- This is true for both variables and character escape-sequences (such as "\n" or "\\")

PHP Variables

```
1
2   <!-- data.php                    -->
3   <!-- Demonstration of PHP data types -->
4
5   <html>
6      <head>
7         <title>PHP data types</title>
8     </head>
9
10    <body>
11
12       <?php
13
14          // declare a string, double and integer
15          $testString = "3.5 seconds"
16          $testDouble = 79.2;
17          $testInteger = 12;
```

Assign a string to variable `$testString`

Assign a double to

Assign an integer to variable `$testInteger`

PHP Variables

```
19          // print each variable's value -->
20          echo $testString . "is a string" . '<br>';
21         print $testDouble ." is a double" . '<br>';
22         print $testInteger. "is an integer" . '<br>'.'<br>' ;
23
24            echo "Now, converting to other types:".
25
26            // call function settype to convert variable
27            // testString to different data types
28            print( "$testString" );
29            settype( $testString, "double" );
30            print( " as a double is $testString <br>" );
31            print( "$testString" );
32            settype( $testString, "integer" );
33            print( " as an integer is $testString <br />" );
34            settype( $testString, "string" );
35
36
37            $data = "98.6 degrees";
38
```

Print each variable's value

Call function settype to

Convert variable $testString back to a string

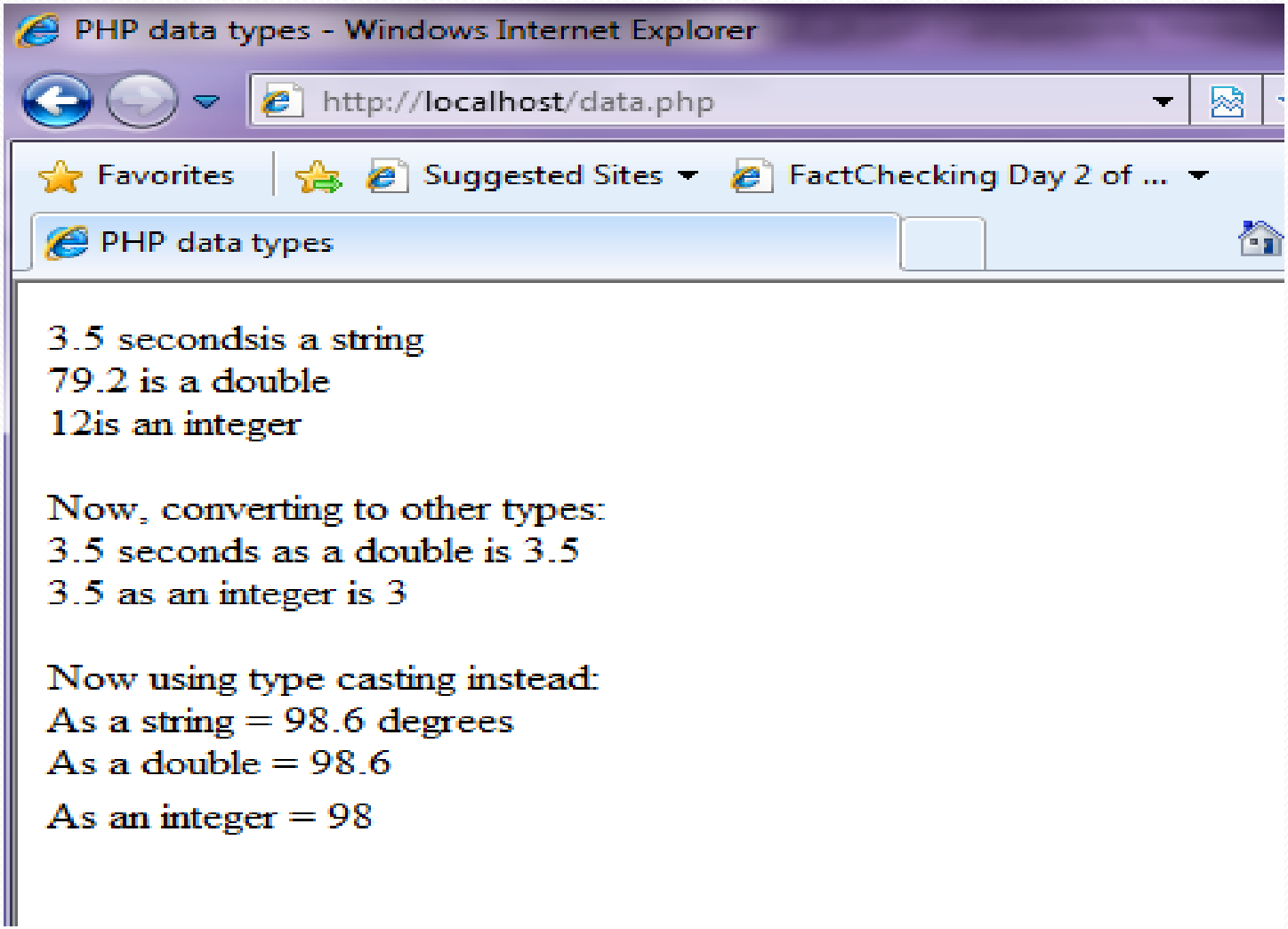
variable $testString to an integer

PHP Variables

```
// use type casting to cast variables to a different type

print( " <br > Now using type casting instead: <br />
    As a string = " . (string) $data .
    "<br />As a double = " . (double) $data .
    "<br />As an integer = " . (integer) $data );
?>
</body>
</html>
```

Use type casting to cast variable `$data` to different types

PHP Variables



PHP Operators

- Operators are used to operate on values. There are four classifications of operators:

- > Arithmetic
- > Assignment
- > Comparison
- > Logical

PHP Operators

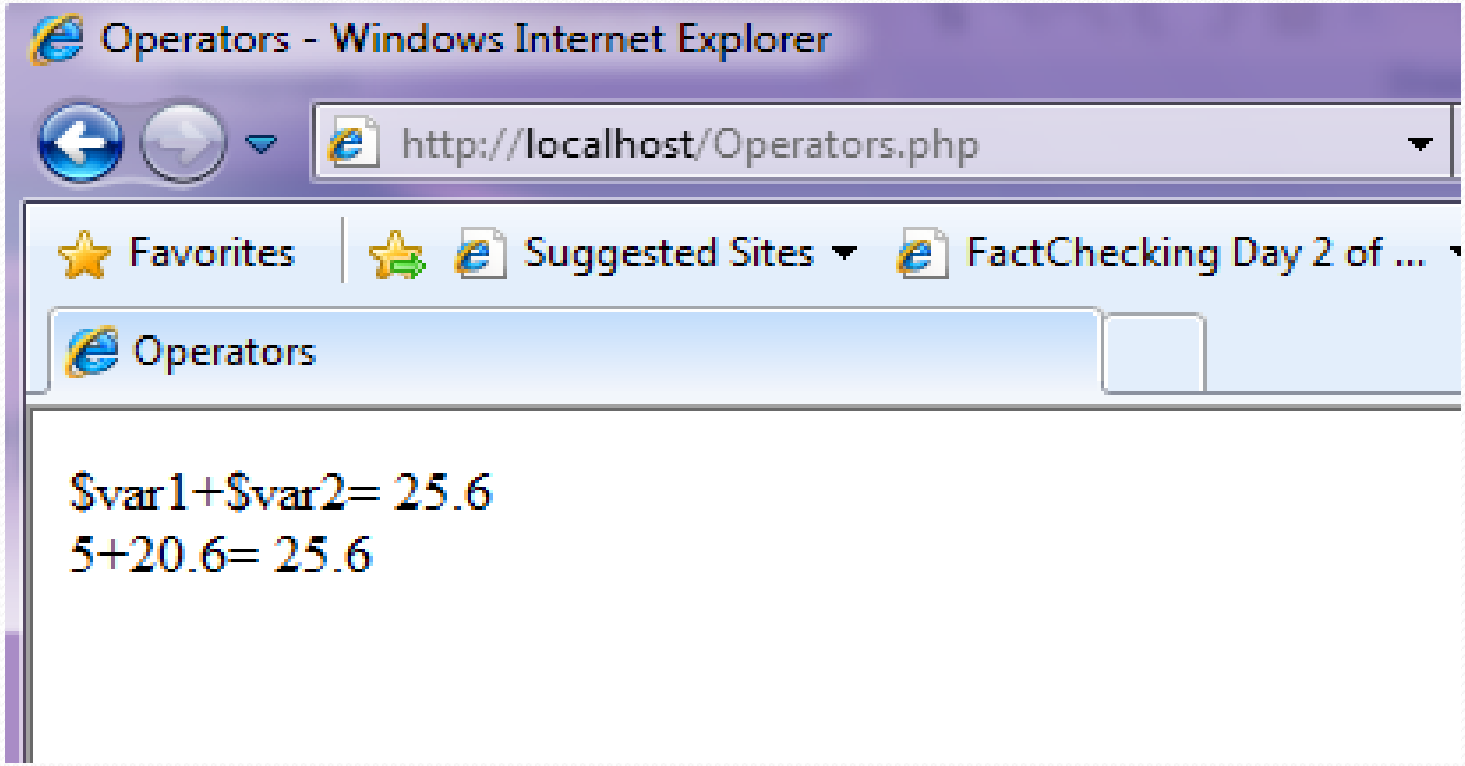
Arithmetic Operators

Operator	Description	Example	Result
+	Addition	x=2 x+2	4
-	Subtraction	x=2 5-x	3
*	Multiplication	x=4 x*5	20
/	Division	15/5 5/2	3 2.5
%	Modulus (division remainder)	5%2 10%8 10%2	1 2 0
++	Increment	x=5 x++	x=6
--	Decrement	x=5 x--	x=4

PHP Operators

```
1
2
3  <html>
4      <head>
5  <title> Operators</title>
6      </head>
7      <body>
8      <?php
9  $var1=5;
10 $var2=20.6;
11 $var3=$var1*$var2;
12 $var4=$var1/$var2;
13 $var5=$var1%$var2;
14
15 echo '$var1+$var2= ' . ($var1+$var2). '<br>';
16 echo "$var1+$var2= " . ($var1+$var2). '<br>';
17
18     ?>
19      </body>
20 </html>
```

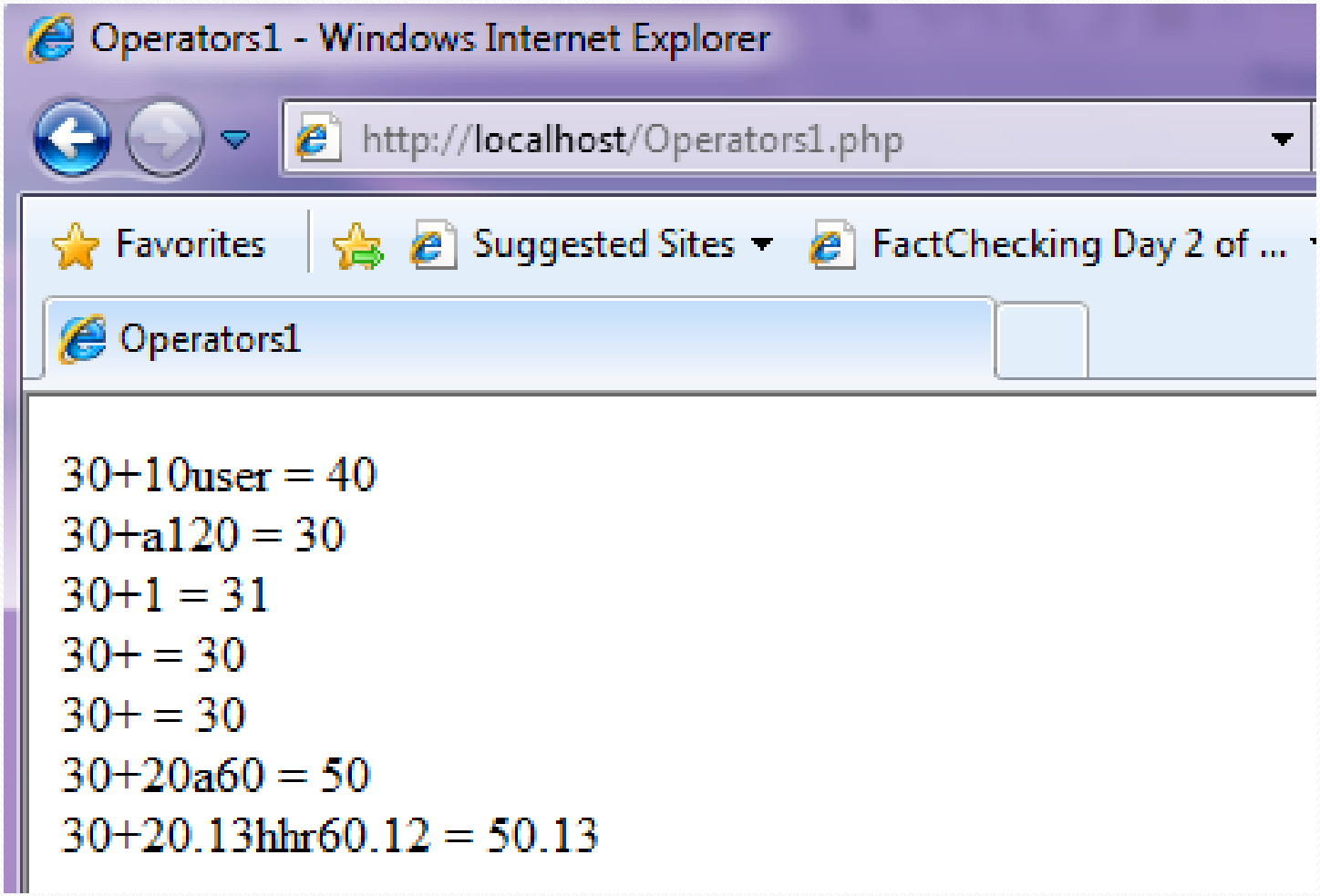
PHP Operators



PHP Operators

```
1
2   <html>
3       <head>
4   <title> Operators1</title>
5       </head>
6       <body>
7       <?php
8
9       $var1 = 30;
10  $var2 = '10user ';
11  $var3 = 'a120';
12
13  $var4 = true;
14  $var5 = false;
15  $var6 = null;
16  $var7 = '20a60';
17  $var8 = '20.13hhr60.12';
18  echo "$var1+$var2 = ".($var1+$var2).'<br>';
19  echo "$var1+$var3 = ".($var1+$var3).'<br>';
20  echo "$var1+$var4 = ".($var1+$var4).'<br>';
21  echo "$var1+$var5 = ".($var1+$var5).'<br>';
22  echo "$var1+$var6 = ".($var1+$var6).'<br>';
23  echo "$var1+$var7 = ".($var1+$var7).'<br>';
24  echo "$var1+$var8 = ".($var1+$var8).'<br>';
25
26      ?>
27       </body>
28  </html>
```

PHP Operators



PHP Operators

Assignment Operators

Operator	Example	Is The Same As
=	x=y	x=y
+=	x+=y	x=x+y
-=	x-=y	x=x-y
*=	x*=y	x=x*y
/=	x/=y	x=x/y
.=	x.=y	x=x.y
%=	x%=y	x=x%y

PHP Operators

Comparison Operators

Operator	Description	Example
==	is equal to	5==8 returns false
!=	is not equal	5!=8 returns true
<>	is not equal	5<>8 returns true
>	is greater than	5>8 returns false
<	is less than	5<8 returns true
>=	is greater than or equal to	5>=8 returns false
<=	is less than or equal to	5<=8 returns true

PHP Operators

Logical Operators

Operator	Description	Example
&&	and	x=6 y=3 (x < 10 && y > 1) returns true
||	or	x=6 y=3 (x==5 || y==5) returns false
!	not	x=6 y=3 !(x==y) returns true

PHP Conditional Statements

- > Very often when you write code, you want to perform different actions for different decisions.
- > You can use conditional statements in your code to do this.
- > In PHP we have the following conditional statements…

PHP Conditional Statements

- > **if** statement - use this statement to execute some code only if a specified condition is true
- > **if...else** statement - use this statement to execute some code if a condition is true and another code if the condition is false
- > **if...elseif....else** statement - use this statement to select one of several blocks of code to be executed
- > **switch** statement - use this statement to select one of many blocks of code to be executed

PHP Conditional Statements

- Example if/else if/else statement:

```
if ($foo == 0) {
        echo 'The variable foo is equal to 0';
}
else if (($foo > 0) && ($foo <= 5)) {
        echo 'The variable foo is between 1 and 5';
}
else {
        echo 'The variable foo is equal to '.$foo;
}
```

If … Else…

- If (condition)
  {
  Statements;
  }
  Else
  {
  Statement;
  }

```
<?php
If($user=="John")
{
        Print "Hello John.";
}
Else
{
        Print "You are not John.";
}
?>
```

No THEN in PHP

PHP Conditional Statements

```
2   <html>
3       <head>
4   <title> Operators2</title>
5       </head>
6       <body>
7       <?php
8
9           $a = 5;
10          print( "The value of variable a is $a <br />" );
11
12          // define constant VALUE
13          define( "VALUE", 5 );
14
15          // add constant VALUE to variable $a
16          $a = $a + VALUE;
17          print( "Variable a after adding constant VALUE is $a <br />" );
18
19          // multiply variable $a by 2
20          $a *= 2;
21          print( "Multiplying variable a b
22
```

Define constant VALUE.

Add constant VALUE to variable $a.

Multiply variable $a by two using the multiplication assignment operator *=.

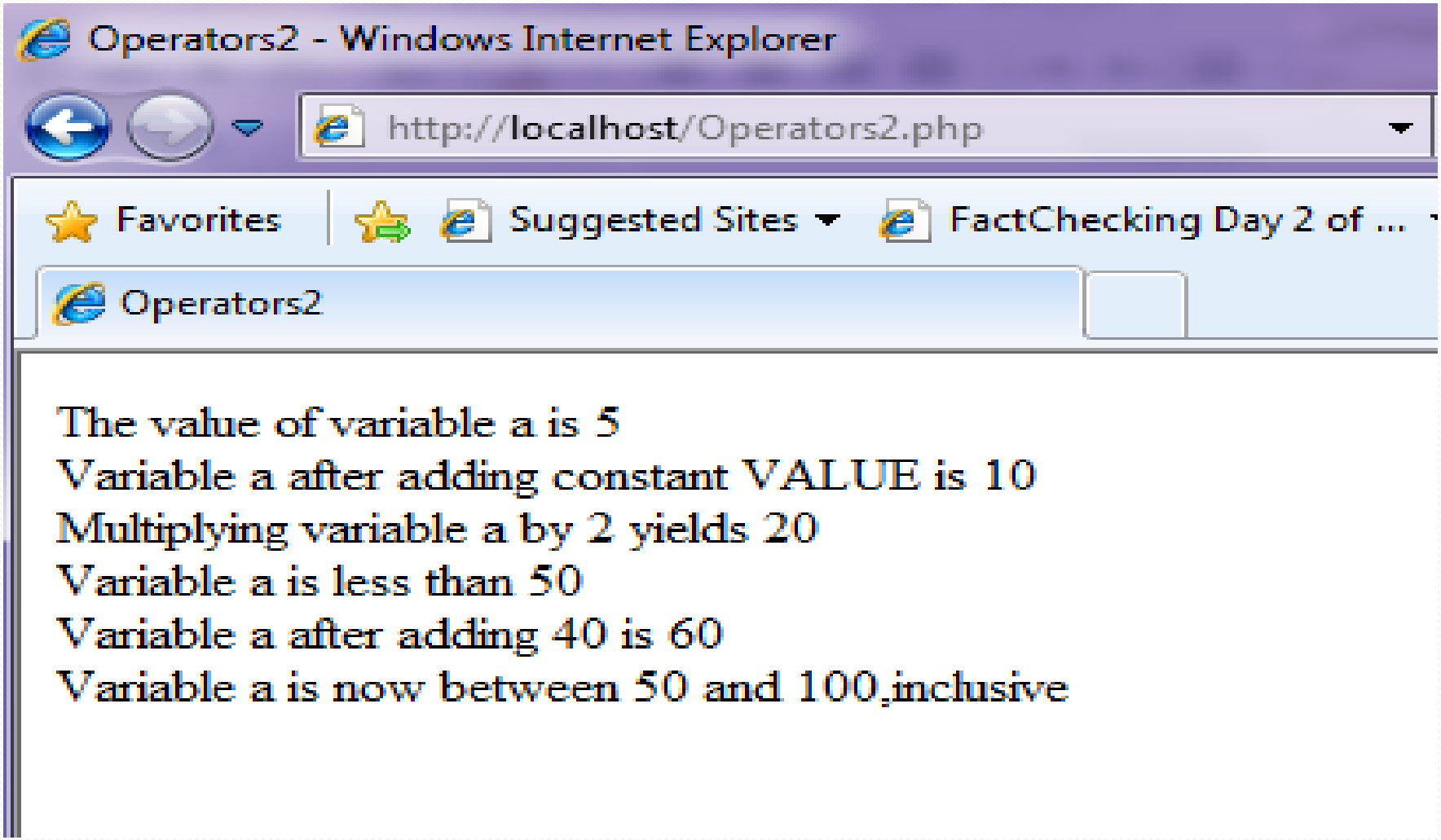
PHP Conditional Statements

```
// test if variable $a is less than 50
if ( $a < 50 )
   print( "Variable a is less than 50 <br />" );

// add 40 to variable $a
$a += 40;
print( "Variable a after adding 40 is $a <br />" );

// test if variable $a is 50 or less
if ( $a < 51 )
   print( "Variable a is still 50 or less<br />" );

// test if variable $a is between 50 and 100, inclusive
elseif ( $a < 101 )
   print( "Variable a is now between 50 and 100,inclusive<br />" );
else
   print( "Variable a is now greater than 100 <br />" );
```

Print if variable $a is less than 50.

Add 40 to variable $a using the addition assignment operator **+=**.

PHP Conditional Statements



PHP Conditional Statements

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

```
switch (n)
{
case label1:
  code to be executed if n=label1;
  break;
case label2:
  code to be executed if n=label2;
  break;
default:
  code to be executed if n is different from both label1 and label2;
}
```

PHP Conditional Statements

- For switches, first we have a single expression n (most often a variable), that is evaluated once.

- The value of the expression is then compared with the values for each case in the structure. If there is a match, the block of code associated with that case is executed.

- Use break to prevent the code from running into the next case automatically. The default statement is used if no match is found.

PHP Conditional Statements

```
<html>
<body>

<?php
switch ($x)
{
case 1:
  echo "Number 1";
  break;
case 2:
  echo "Number 2";
  break;
case 3:
  echo "Number 3";
  break;
default:
  echo "No number between 1 and 3";
}
?>

</body>
</html>
```

PHP Loops

- > Often when you write code, you want the same block of code to run over and over again in a row. Instead of adding several almost equal lines in a script we can use loops to perform a task like this.
- > In PHP, we have the following looping statements:

PHP Loops

- > **while** - loops through a block of code while a specified condition is true
- > **do...while** - loops through a block of code once, and then repeats the loop as long as a specified condition is true
- > **for** - loops through a block of code a specified number of times
- > **foreach** - loops through a block of code for each element in an array

PHP Loops - While

- The while loop executes a block of code while a condition is true. The example below defines a loop that starts with
- i=1. The loop will
- continue to run as
- long as i is less
- than, or equal to 5.
- i will increase by 1
- each time the loop
- runs:

```
<html>
<body>

<?php
$i=1;
while($i<=5)
  {
  echo "The number is " . $i . "<br />";
  $i++;
  }
?>

</body>
</html>
```

PHP Loops - While

Output:

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

PHP Loops – Do … While

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

- The next example defines a loop that starts with i=1. It will then increment i with 1, and write some output. Then the condition is checked, and the loop will continue to run as long as i is less than, or equal to 5:

PHP Loops – Do … While

```
<html>
<body>

<?php
$i=1;
do
  {
  $i++;
  echo "The number is " . $i . "<br />";
  }
while ($i<=5);
?>

</body>
</html>
```

PHP Loops – Do … While

Output:

```
The number is 2
The number is 3
The number is 4
The number is 5
The number is 6
```

PHP Loops - For

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

Syntax

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

PHP Loops - For

- Parameters:
- > **init**: Mostly used to set a counter (but can be any code to be executed once at the beginning of the loop)
- > **condition**: Evaluated for each loop iteration. If it evaluates to TRUE, the loop continues. If it evaluates to FALSE, the loop ends.
- > **increment**: Mostly used to increment a counter (but can be any code to be executed at the end of the loop)

PHP Loops - For

- The example below defines a loop that starts with i=1. The loop will continue to run as long as i is less than, or equal to 5. i will increase by 1 each time the loop runs:

```
<html>
<body>

<?php
for ($i=1; $i<=5; $i++)
  {
  echo "The number is " . $i . "<br />";
  }
?>

</body>
</html>
```

PHP Loops - For

Output:

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

PHP Loops - Foreach

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

- For every loop iteration, the value of the current array element is assigned to $value (and the array pointer is moved by one) - so on the next loop iteration, you'll be looking at the next array value.

PHP Loops - Foreach

- The following example demonstrates a loop that will print the values of the given array:

```
<html>
<body>

<?php
$x=array("one","two","three");
foreach ($x as $value)
  {
  echo $value . "<br />";
  }
?>

</body>
</html>
```

PHP Loops - Foreach

- Winner of the most impressive slide award

Output:

```
one
two
three
```

PHP Arrays

- > An array variable is a storage area holding a number or text. The problem is, a variable will hold only one value.
- > An array is a special variable, which can store multiple values in one single variable.

PHP Arrays

- If you have a list of items (a list of car names, for example), storing the cars in single variables could look like this:

```
$cars1="Saab";
$cars2="Volvo";
$cars3="BMW";
```

PHP Arrays

- > However, what if you want to loop through the cars and find a specific one? And what if you had not 3 cars, but 300?
- > The best solution here is to use an array.
- > An array can hold all your variable values under a single name. And you can access the values by referring to the array name.
- > Each element in the array has its own index so that it can be easily accessed.

PHP Arrays

- In PHP, there are three kind of arrays:
- > **Numeric array** - An array with a numeric index
- > **Associative array** - An array where each ID key is associated with a value
- > **Multidimensional array** - An array containing one or more arrays

PHP Numeric Arrays

- > A numeric array stores each array element with a numeric index.
- > There are two methods to create a numeric array.

PHP Numeric Arrays

- In the following example the index is automatically assigned (the index starts at 0):

```
$cars=array("Saab","Volvo","BMW","Toyota");
```

- In the following example we assign the index manually:

```
$cars[0]="Saab";
$cars[1]="Volvo";
$cars[2]="BMW";
$cars[3]="Toyota";
```

PHP Numeric Arrays

- In the following example you access the variable values by referring to the array name and index:

```
<?php
$cars[0]="Saab";
$cars[1]="Volvo";
$cars[2]="BMW";
$cars[3]="Toyota";
echo $cars[0] . " and " . $cars[1] . " are Swedish cars.";
?>
```

- The code above will output:

```
Saab and Volvo are Swedish cars.
```

PHP Numeric Arrays

```
<?php
    // create array first Numeric Arrays
    print( "<strong>Creating the first array Numeric Arrays</strong>
      <br />" );
    $first[ 0 ] = "zero";
    $first[ 1 ] = "one";
    $first[ 2 ] = "two";
    $first[] = "three";

    // print each element's i
    for ( $i = 0; $i < count( $first ); $i++ )
      print( "Element $i is $first[$i] <br />" );


    print( "<br /><strong>Creating

    // call function array to creat
    $second = array( "zero", "one", "two", "three" );
    for ( $i = 0; $i < count( $second ); $i++ )
      print( "Element $i is $second[$i] <br />" );
?>
```

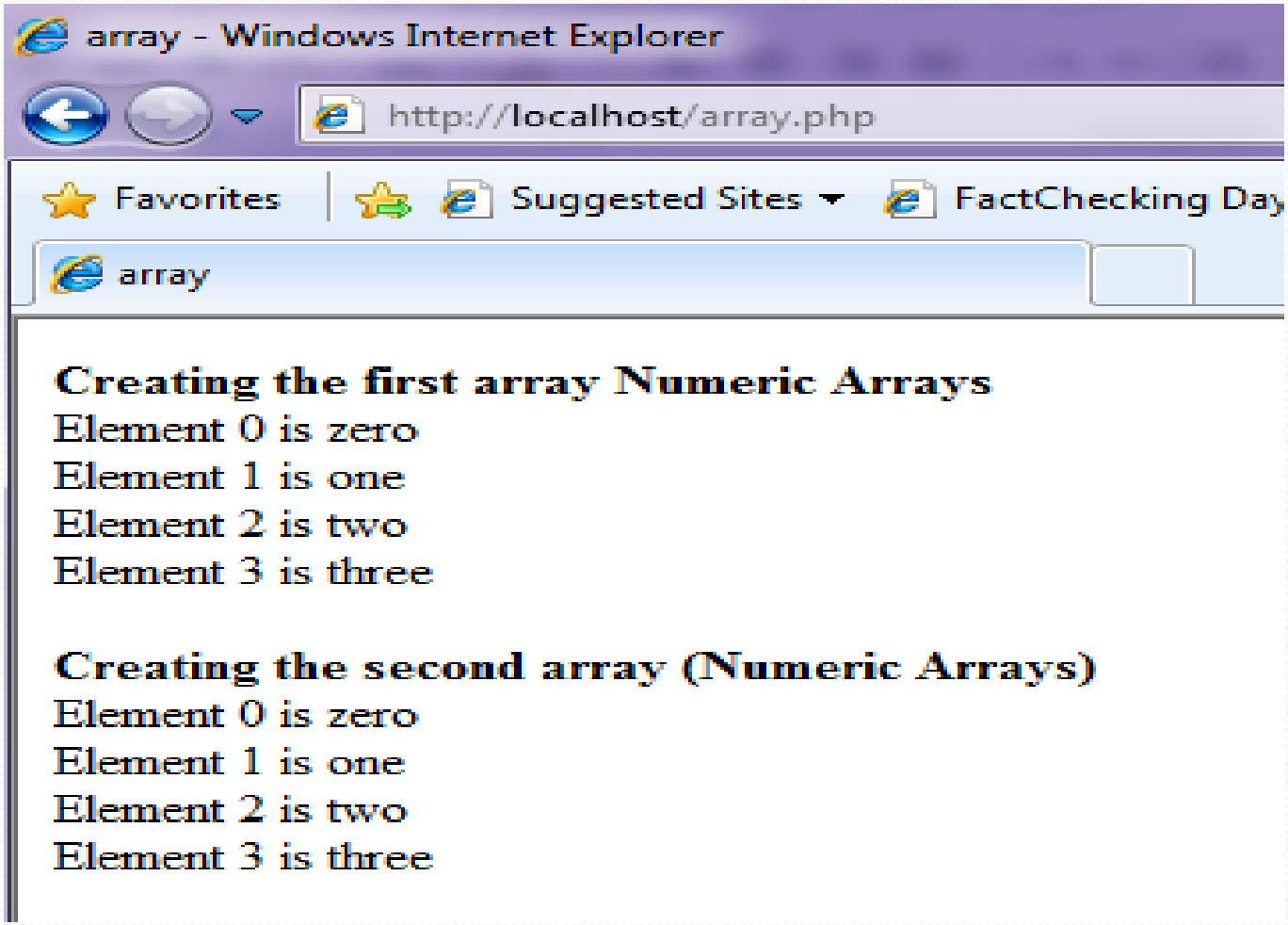
Create the array `$first` by assigning a value to an array element.

Assign a value to the array, omitting the index.

Use a `for` loop to print out each element's index and value. Function `count` returns the total number of elements in the array.

Call function `array` to create an array that contains the arguments passed to it. Store the array in variable `$second`.

PHP Numeric Arrays



PHP Associative Arrays

- > With an associative array, each ID key is associated with a value.
- > When storing data about specific named values, a numerical array is not always the best way to do it.
- > With associative arrays we can use the values as keys and assign values to them.

PHP Associative Arrays

- In this example we use an array to assign ages to the different persons:

```
$ages = array("Peter"=>32, "Quagmire"=>30, "Joe"=>34);
```

- This example is the same as the one above, but shows a different way of creating the array:

```
$ages['Peter'] = "32";
$ages['Quagmire'] = "30";
$ages['Joe'] = "34";
```

PHP Associative Arrays

The ID keys can be used in a script:

```
<?php
$ages['Peter'] = "32";
$ages['Quagmire'] = "30";
$ages['Joe'] = "34";

echo "Peter is " . $ages['Peter'] . " years old.";
?>
```

The code above will output:

```
Peter is 32 years old.
```

PHP Associative Arrays

```
1  <html>
2  <head>
3  <title> array2</title>
4  </head>
5  <body>
6
7  <?php
8          print( "<br /><strong>Creating the third array Associative Arrays</strong><br />" );
9
10         // assign values to non-numerical indices
11         $third[ "ArtTic" ] = 21;
12         $third[ "LunaTic" ] = 18;
13         $third[ "GalAnt" ] = 23;
14
15         // iterate through the ar
16         // element's name and value
17         for ( reset( $third ); $element = key( $third );next( $third ) )
18             print( "$element is $third[$element] <br />" );
19
20
21 </html>
22
```

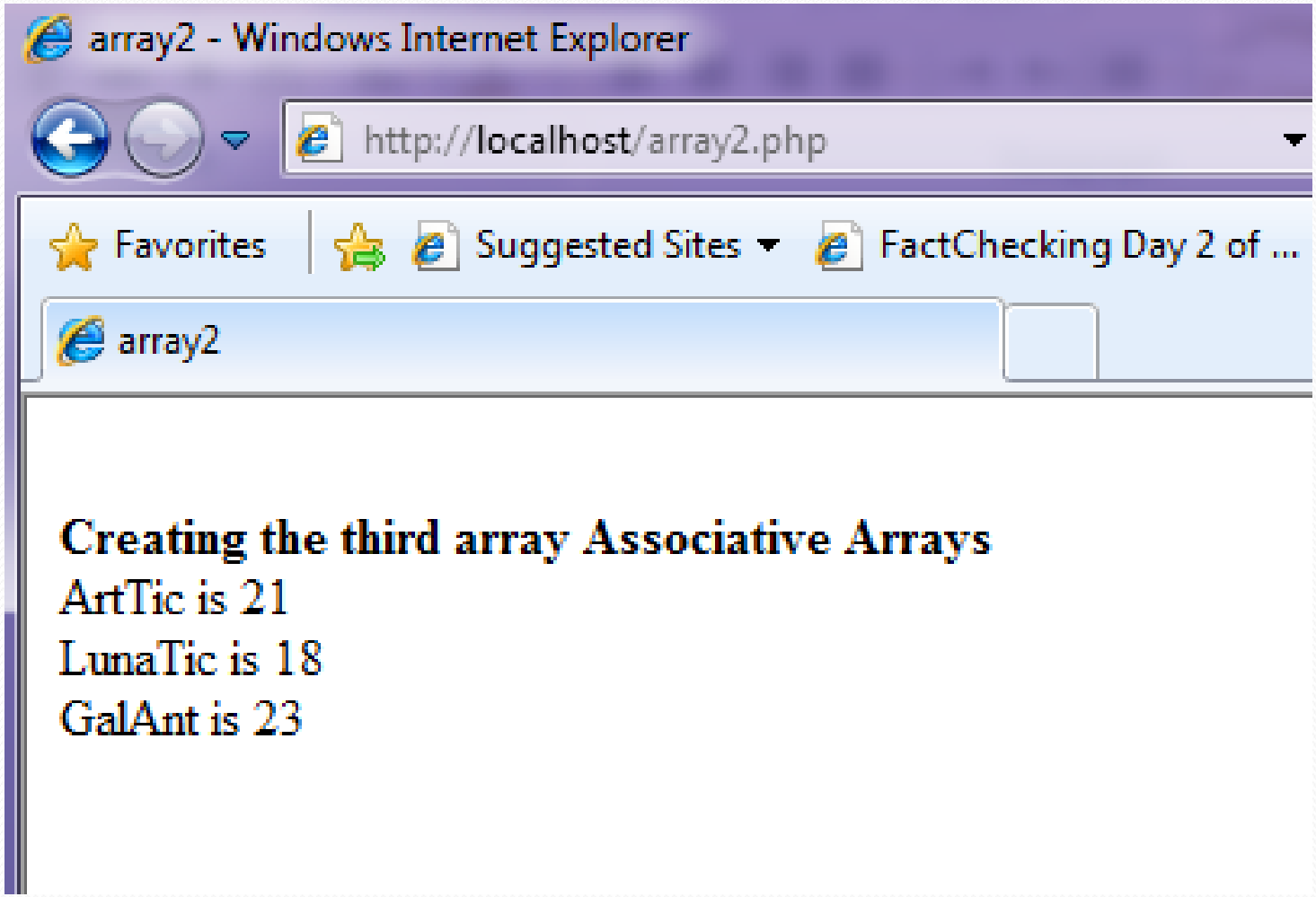
Assign values to non-numerical indices in array `$third`.

Function `reset` sets the internal pointer to the first element of the array.

Function `next` moves the internal pointer to the next element.

index of the element which the internal pointer references.

PHP Associative Arrays



PHP Multidimensional Arrays

- In a multidimensional array, each element in the main array can also be an array.
- And each element in the sub-array can be an array, and so on.

PHP Multidimensional Arrays

In this example we create a multidimensional array, with automatically assigned ID keys:

```
$families = array
  (
  "Griffin"=>array
  (
  "Peter",
  "Lois",
  "Megan"
  ),
  "Quagmire"=>array
  (
  "Glenn"
  ),
  "Brown"=>array
  (
  "Cleveland",
  "Loretta",
  "Junior"
  )
  );
```

PHP Multidimensional Arrays

The array above would look like this if written to the output:

```
Array
(
[Griffin] => Array
   (
   [0] => Peter
   [1] => Lois
   [2] => Megan
   )
[Quagmire] => Array
   (
   [0] => Glenn
   )
[Brown] => Array
   (
   [0] => Cleveland
   [1] => Loretta
   [2] => Junior
   )
)
```

PHP Multidimensional Arrays

Lets try displaying a single value from the array above:

```
echo "Is " . $families['Griffin'][2] .
" a part of the Griffin family?";
```

The code above will output:

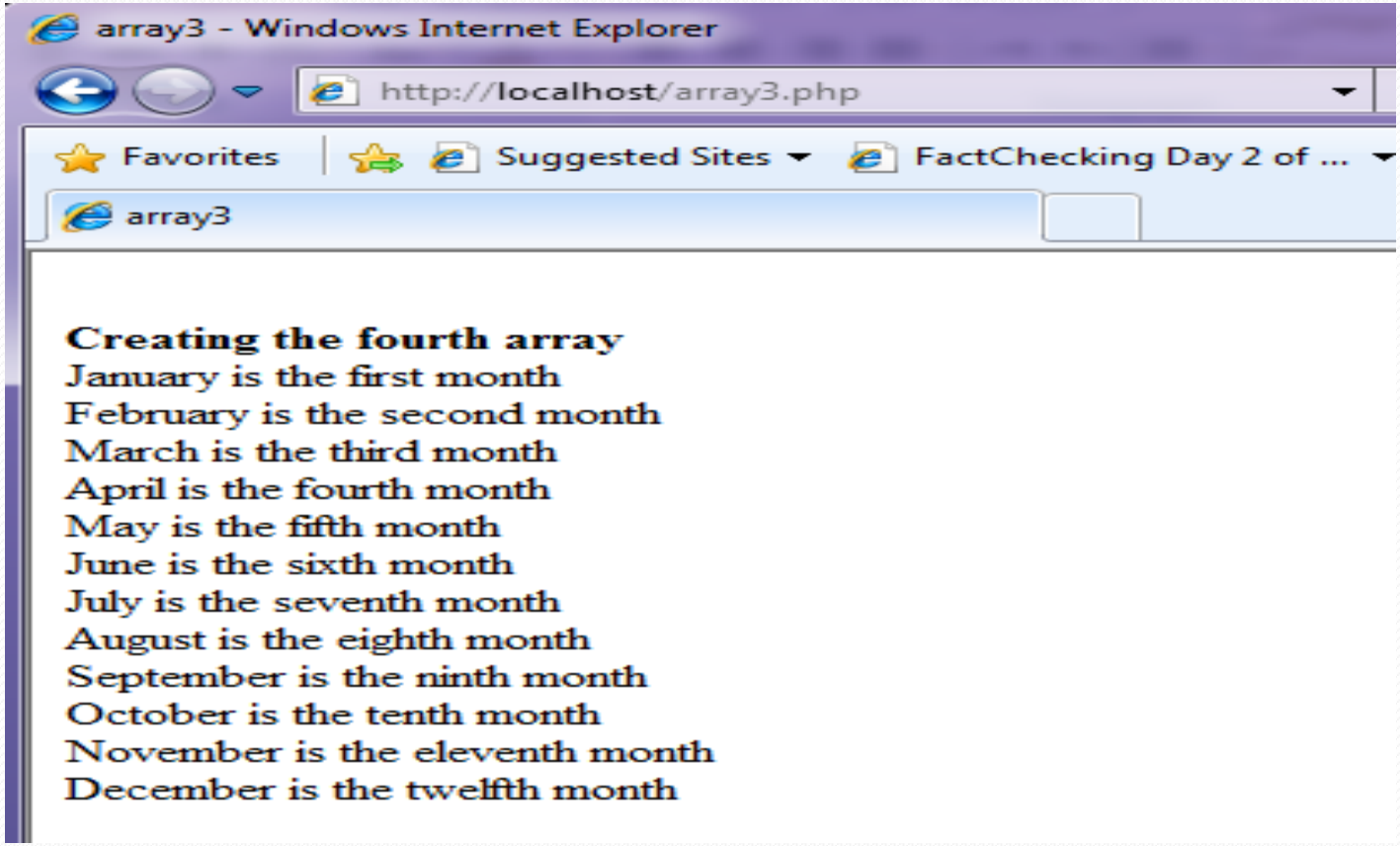
```
Is Megan a part of the Griffin family?
```

PHP Multidimensional Arrays

```
8  <?php
9      print( "<br /><strong>Creating the fourth array </strong><br />" );
10
11     // call function array to create array fourth using
12     // string indices
13     $fourth = array(
14        "January"   => "first",    "February" => "second",
15        "March"     => "third
16        "May"       => "fifth
17        "July"      => "seven
18        "September" => "ninth",    "October"  => "tenth",
19        "November"  => "eleventh","December" => "twelfth"
20        );
21
22     // print each element's name and value
23     foreach ( $fourth as $element => $value )
24        print( "$element is the $value month <br />" );
25  ?>
```

Operator **=>** is used in function `array` to assign each element a string index. The value to the left of the operator is the array index, and the value to the right is the element's value.

PHP Multidimensional Arrays



Arrays

```
<?php
$arr = array("foo" => "bar", 12 => true);
echo $arr["foo"]; // bar
echo $arr[12];    // 1
?>
```

```
<?php
array(5 => 43, 32, 56, "b" => 12);
array(5 => 43, 6 => 32, 7 => 56, "b" => 12);
?>
```

`array()` = creates arrays

key = either an integer or a string.

value = any PHP type.

if no key given (as in example), the PHP interpreter uses (maximum of the integer indices + 1).

if an existing key, its value will be overwritten.

PHP Functions

- > We will now explore how to create your own functions.
- > To keep the script from being executed when the page loads, you can put it into a function.
- > A function will be executed by a call to the function.
- > You may call a function from anywhere within a page.

PHP Functions

- A function will be executed by a call to the function.

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

- > Give the function a name that reflects what the function does
- > The function name can start with a letter or underscore (not a number)

PHP Functions

- A simple function that writes a name when it is called:

```
<html>
<body>

<?php
function writeName()
{
echo "Kai Jim Refsnes";
}

echo "My name is ";
writeName();
?>

</body>
</html>
```

PHP Functions - Parameters

- Adding parameters…
- > To add more functionality to a function, we can add parameters. A parameter is just like a variable.
- > Parameters are specified after the function name, inside the parentheses.

PHP Functions - Parameters

The following example will write different first names, but equal last name:

```
<html>
<body>

<?php
function writeName($fname)
{
echo $fname . " Refsnes.<br />";
}

echo "My name is ";
writeName("Kai Jim");
echo "My sister's name is ";
writeName("Hege");
echo "My brother's name is ";
writeName("Stale");
?>

</body>
</html>
```

PHP Functions - Parameters

Output:

```
My name is Kai Jim Refsnes.
My sister's name is Hege Refsnes.
My brother's name is Stale Refsnes.
```

PHP Functions - Parameters

```
<html>
<body>

<?php
function writeName($fname,$punctuation)
{
echo $fname . " Refsnes" . $punctuation . "<br />";
}

echo "My name is ";
writeName("Kai Jim",".");
echo "My sister's name is ";
writeName("Hege","!");
echo "My brother's name is ";
writeName("Ståle","?");
?>

</body>
</html>
```

This example adds different punctuation.

PHP Functions - Parameters

Output:

```
My name is Kai Jim Refsnes.
My sister's name is Hege Refsnes!
My brother's name is Ståle Refsnes?
```

Functions example

```
<?php
    // This is a function
     function foo($arg_1, $arg_2)
      {
        $arg_2 = $arg_1 * $arg_2;
        return $arg_2;
     }

     $result_1 = foo(12, 3);            // Store the function
     echo $result_1;                    // Outputs 36
     echo foo(12, 3);                   // Outputs 36
?>
```

Lab Activities:

- Write php script to draw engineering shapes.

➔ diamond, parallelogram, triangle