Introduction

PHP

- PHP is a server scripting language, and a powerful tool for making dynamic and interactive Web pages.

- PHP is a widely-used, free, and efficient alternative to competitors such as Microsoft's ASP.

PHP

While **PHP** originally stood for **Personal Home Page**, it now stands for **(PHP**: Hypertext Preprocessor), which is a recursive acronym.

P **-> PHP**
H -> **Hypertext**
P -> **Preprocessor**

- It is powerful enough to be at the core of the biggest blogging system on the web (WordPress)!
- It is deep enough to run the largest social network (Facebook)!
- It is also easy enough to be a beginner's first server side language!

What PHP can do

- PHP can generate dynamic page content
- PHP can create, open, read, write, delete, and close files on the server
- PHP can collect form data
- PHP can send and receive cookies

- PHP can add, delete, modify data in your database
- PHP can be used to control user-access
- PHP can encrypt data

With PHP you are not limited to output HTML.

You can output

- Images
- PDF files
- Flash movies
- XHTML
- XML.

Why PHP

- PHP runs on various platforms (Windows, Linux, Unix, Mac OS X, etc.)
- PHP is compatible with almost all servers used today (Apache, IIS, etc.)

- PHP supports a wide range of databases like MySQL, MS Access, MS SQL Server, Oracle, etc
- PHP is free.
- PHP is easy to learn and runs efficiently on the server side

FREE

- PHP is FREE
- It runs on Linux which is FREE
- It support MySQL Free
- It runs on Apache which is also free

That is why most of the time hosting of PHP is cheap.

Basic PHP Syntax

- A PHP script can be placed anywhere in the document.
- A PHP script starts with **<?php** and ends with **?>**:
-
```
<?php
// PHP code goes here
?>
```

PHP File Extension

- Extension of PHP page should be .php

End of Statement

PHP statements end with a semicolon (;)

Installation and Configuration of WAMP

To get the PHP running at your computer you will need to

- install a web server
- install PHP
- install a database, such as MySQL

The official PHP website (PHP.net) has installation instructions for PHP: http://php.net/manual/en/install.php

LAMP & WAMP

- Installation and configuration process of PHP can be easily done using Bundled Software know as LAMP and WAMP

- LAMP Stands for **L**inux, **A**pache, **M**ySQL and **P**HP.

- WAMP is Windows, **A**pache, **M**ySQL and **P**HP.

- We will be using a WAMP bundle

WAMP

- This is a open source platform.
- WAMP Server works on Windows Operating System only.
- WAMP is a combine package of Windows, **A**pache, **M**ySQL and **P**HP.
- .

VertrigoServ

VertrigoServ is easy to install package consisting of

- Apache (HTTP web server)
- PHP(Server Side Scripting)
- MySQL (SQL Database Management System)
- PhpMyAdmin (tool written in PHP intended to handle the administration of MySQL)

Hello World

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

<h1>My first PHP page</h1>

<?php
        echo "Hello World!";
?>

</body>
</html>
```

Echo & Print Statement

In PHP there are two basic ways to get output: echo and print.

The differences are small:

- echo has no return value while print has a return value of 1 so it can be used in expressions.
- echo can take multiple parameters (although such usage is rare) while print can take one argument.
- echo is marginally faster than print.

The PHP echo Statement

The echo statement can be used with or without parentheses: echo or echo().

Display Text

The following example shows how to output text with the echo command (notice that the text can contain HTML markup)

Example

```
<?php
	echo "<h2>PHP is Fun!</h2>";
	echo "Hello world!<br>";
	echo "I'm about to learn PHP!<br>";
	echo "This ", "string ", "was ", "made ",
	"with multiple parameters.";
?>
```

Display Variables

```
<?php
    $txt1 = "Learn PHP";
    $txt2 = "W3Schools.com";
    $x = 5;
    $y = 4;

    echo "<h2>$txt1</h2>";
    echo "Study PHP at $txt2<br>";
    echo $x + $y;
?>
```

The PHP print Statement

- The following example shows how to output text with the print command (notice that the text can contain HTML markup)

-
  ```
  <?php
  print "<h2>PHP is Fun!</h2>";
  print "Hello world!<br>";
  print "I'm about to learn PHP!";
  ?>
  ```

Comments in PHP

A comment in PHP code is a line that is not read/executed as part of the program. Its only purpose is to be read by someone who is looking at the code.

Usage of Comments

Comments can be used to:

- Let others understand what you are doing
- Remind yourself of what you did – Most programmers have experienced coming back to their own work a year or two later and having to re-figure out what they did. Comments can remind you of what you were thinking when you wrote the code.

Ways of commenting

PHP supports several ways of commenting:

- // (Single Line Comment)
- #   (Single Line Comment)
- /* */   (Multiple Line Comments)

Example

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

<?php
// This is a single-line comment

# This is also a single-line comment

/*
This is a multiple-lines comment block
that spans over multiple
lines
*/
?>

</body>
</html>
```

Commenting a Part of Statment

```
<!DOCTYPE html>
<html>
<body>
<?php>
// You can also use comments to leave out
parts of a code line
$x = 5 /* + 15 */ + 5;
echo $x;
?>
</body>
</html>
```

PHP Case Sensitivity

PHP Case Sensitivity

In PHP, all keywords (e.g. if, else, while, echo, etc.), classes, functions, and user-defined functions are NOT case-sensitive.

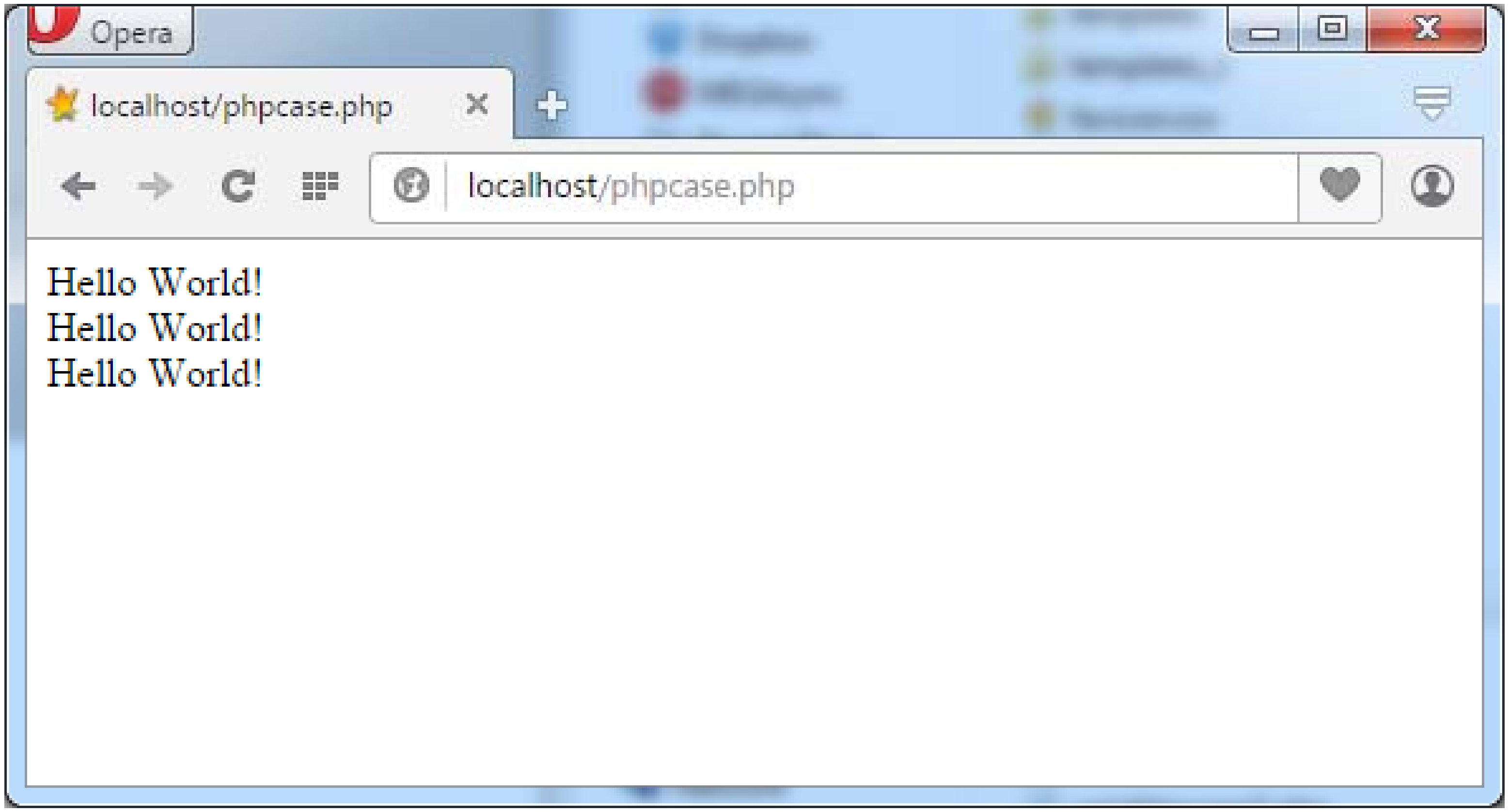
Example

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

<?php

        ECHO "Hello World!<br>";
        echo "Hello World!<br>";
        EcHo "Hello World!<br>";
?>

</body>
</html>
```



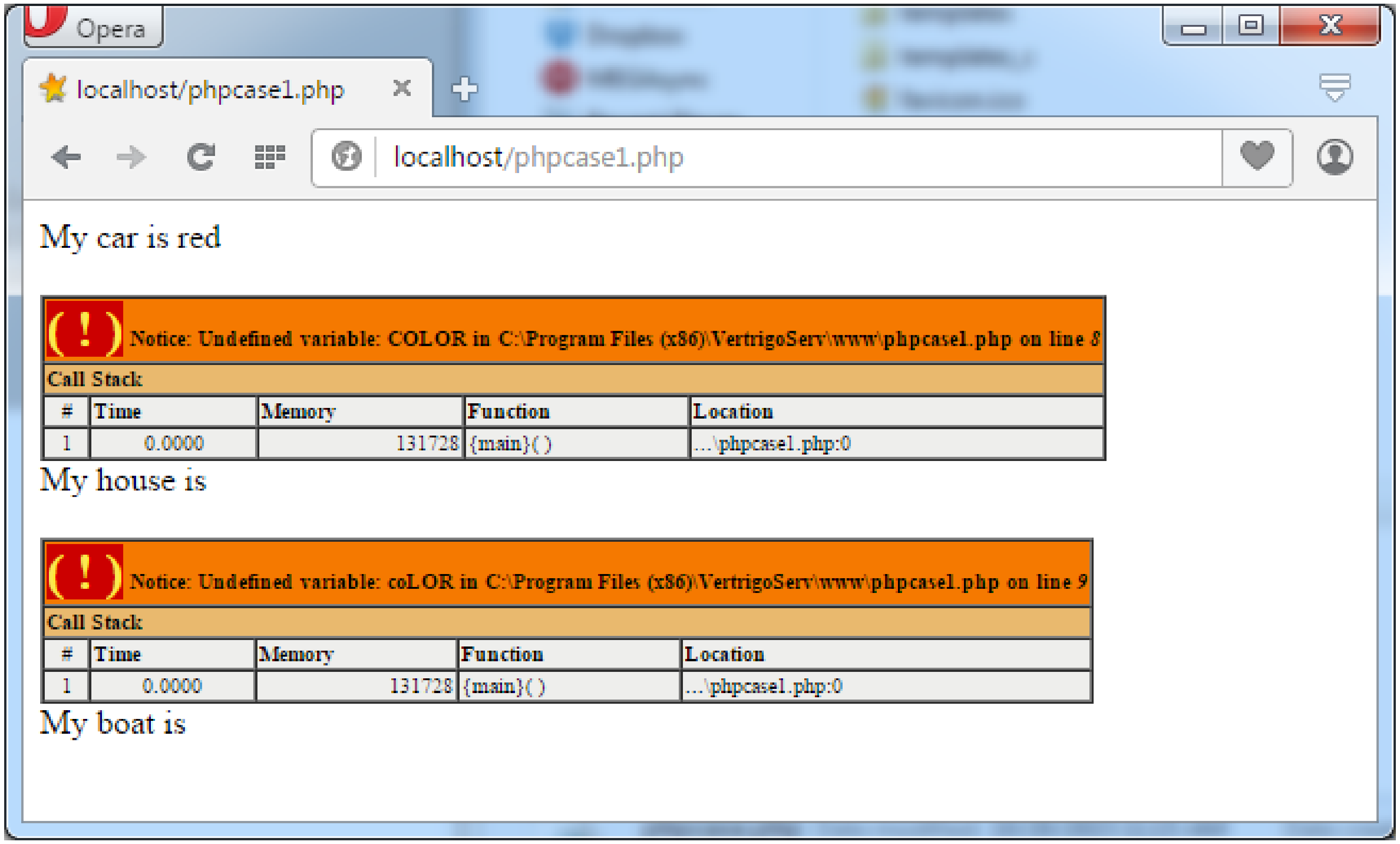
```
Hello World!
Hello World!
Hello World!
```

All variable names are case-sensitive

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

<?php
        $color = "red";
        echo "My car is " . $color . "<br>";
        echo "My house is " . $COLOR . "<br>";
        echo "My boat is " . $coLOR . "<br>";
?>

</body>
</html>
```



```
My car is red
```

(!)	Notice: Undefined variable: COLOR in C:\Program Files (x86)\VertrigoServ\www\phpcase1.php on line 8			
Call Stack				
#	Time	Memory	Function	Location
1	0.0000	131728	{main}( )	...\phpcase1.php:0

```
My house is
```

(!)	Notice: Undefined variable: coLOR in C:\Program Files (x86)\VertrigoServ\www\phpcase1.php on line 9			
Call Stack				
#	Time	Memory	Function	Location
1	0.0000	131728	{main}( )	...\phpcase1.php:0

```
My boat is
```

Variable Names

- A variable can have a short name (like x and y) or a more descriptive name (age, studentname, total_volume).

Rules for PHP variables:

- A variable starts with the $ sign, followed by the name of the variable
- A variable name must start with a letter or the underscore character
- A variable name cannot start with a number
- A variable name can only contain alpha-numeric characters and underscores (A-z, 0-9, and _)

Important

- Variable names are case-sensitive ($age and $AGE are two different variables)

Declaring PHP Variables

In PHP, a variable starts with the $ sign, followed by the name of the variable:

Example:

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

<?php
$txt = "Hello world!";
$x = 5;
$y = 10.5;

echo $txt;
echo "<br>";
echo $x;
echo "<br>";
echo $y;
?>

</body>
</html>
```

After the execution of the statements above, the variable **$txt** will hold the value **Hello world!**, the variable **$x** will hold the value **5**, and the variable **$y** will hold the value **10.5**.

Note: Unlike other programming languages, PHP has no command for declaring a variable. It is created the moment you first assign a value to it.

Data Types

Variables can store data of different types, and different data types can do different things.
PHP supports the following data types:

- String
- Integer
- Float (double)
- Boolean
- Array
- Object
- NULL

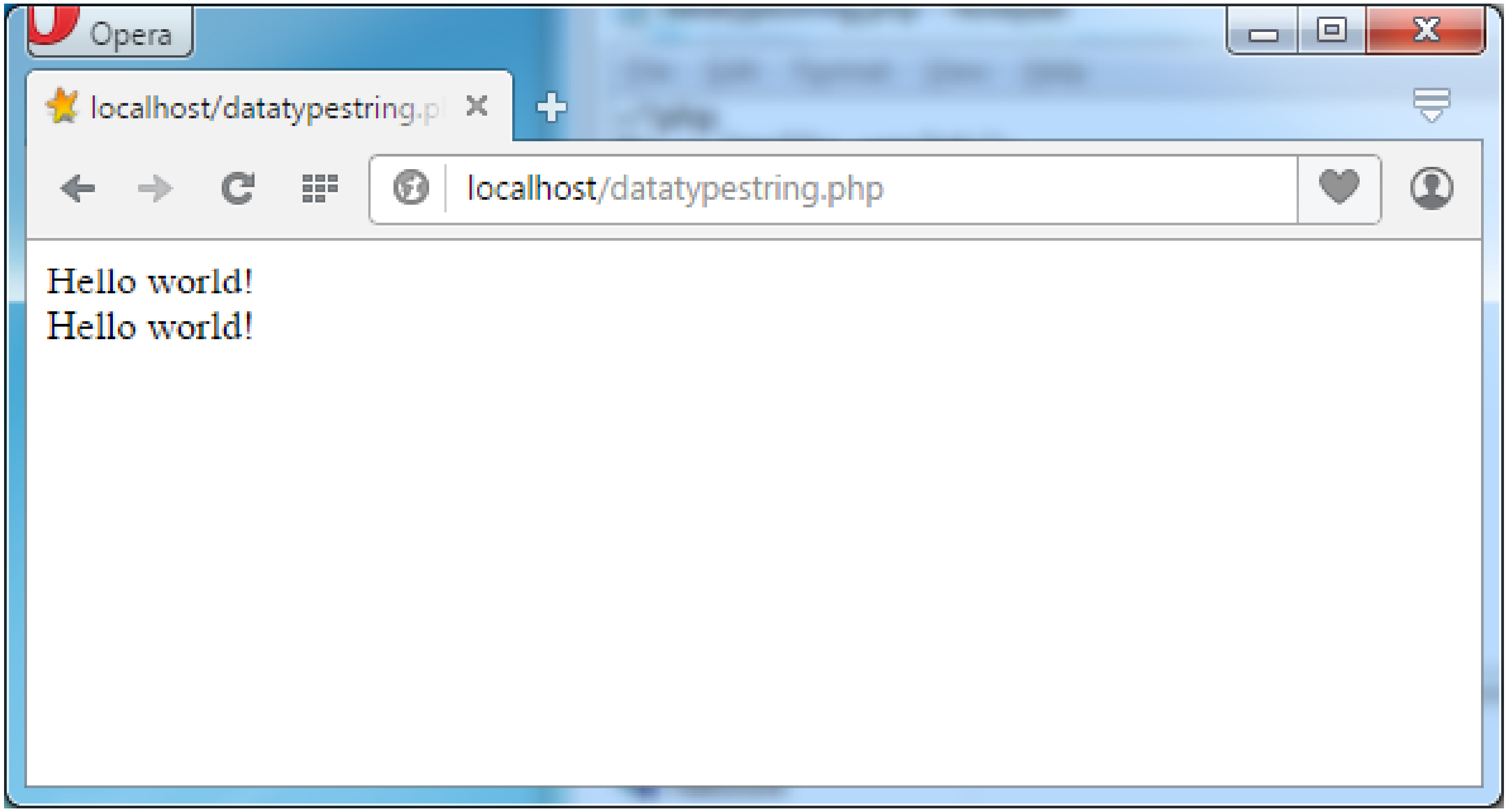
PHP String

- A string is a sequence of characters, like "Hello world!".
- A string can be any text inside quotes. You can use single or double quotes:

String Example

```
<?php
$x = "Hello world!";
$y = 'Hello world!';

echo $x;
echo "<br>";
echo $y;
?>
```



```
Hello world!
Hello world!
```

PHP Integer

- An integer is a whole number (without decimals). It is a number between -2,147,483,648 and +2,147,483,647.

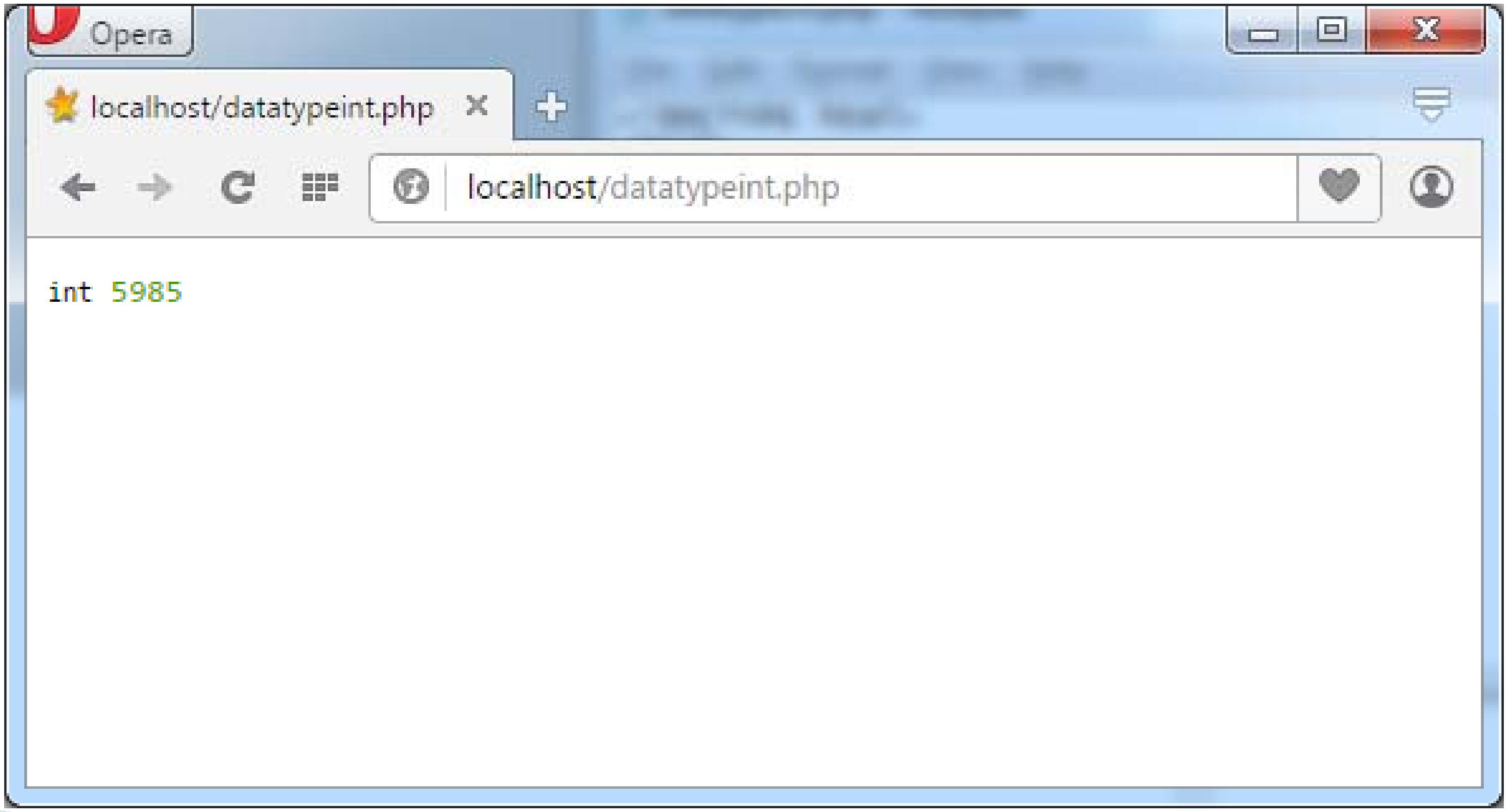
Rules for integers:

- An integer must have at least one digit (0–9)
- An integer cannot contain comma or blanks
- An integer must not have a decimal point
- An integer can be either positive or negative
- Integers can be specified in three formats: decimal (10–based), hexadecimal (16–based – prefixed with 0x) or octal (8–based – prefixed with 0)

Example

```
<?php
$x = 5985;
var_dump($x);
?>
```

var_dump($x) will return the data type variable.


Opera
localhost/datatypeint.php
localhost/datatypeint.php

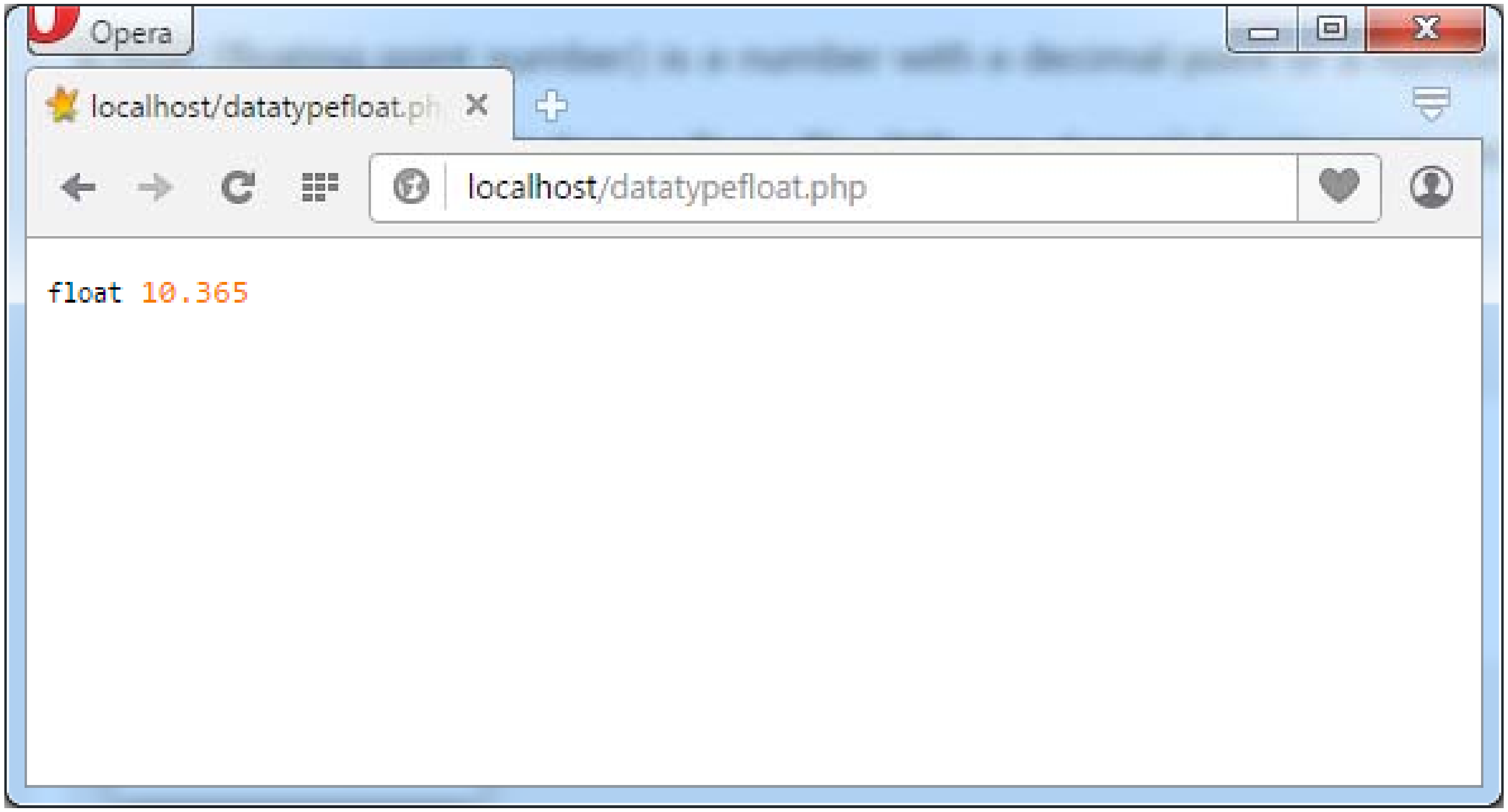
```
int 5985
```

PHP Float

A float (floating point number) is a number with a decimal point or a number in exponential form.

In the following example $x is a float.

```
<?php
$x = 10.365;
var_dump($x);
?>
```


Opera
localhost/datatypefloat.ph
localhost/datatypefloat.php

```
float 10.365
```

PHP Boolean

A Boolean represents two possible states: TRUE or FALSE.

```
$x = true;
$y = false;
```

Booleans are often used in conditional testing. You will learn more about conditional testing in a later chapter of this tutorial.

PHP Object

An object is a data type which stores data and information on how to process that data.

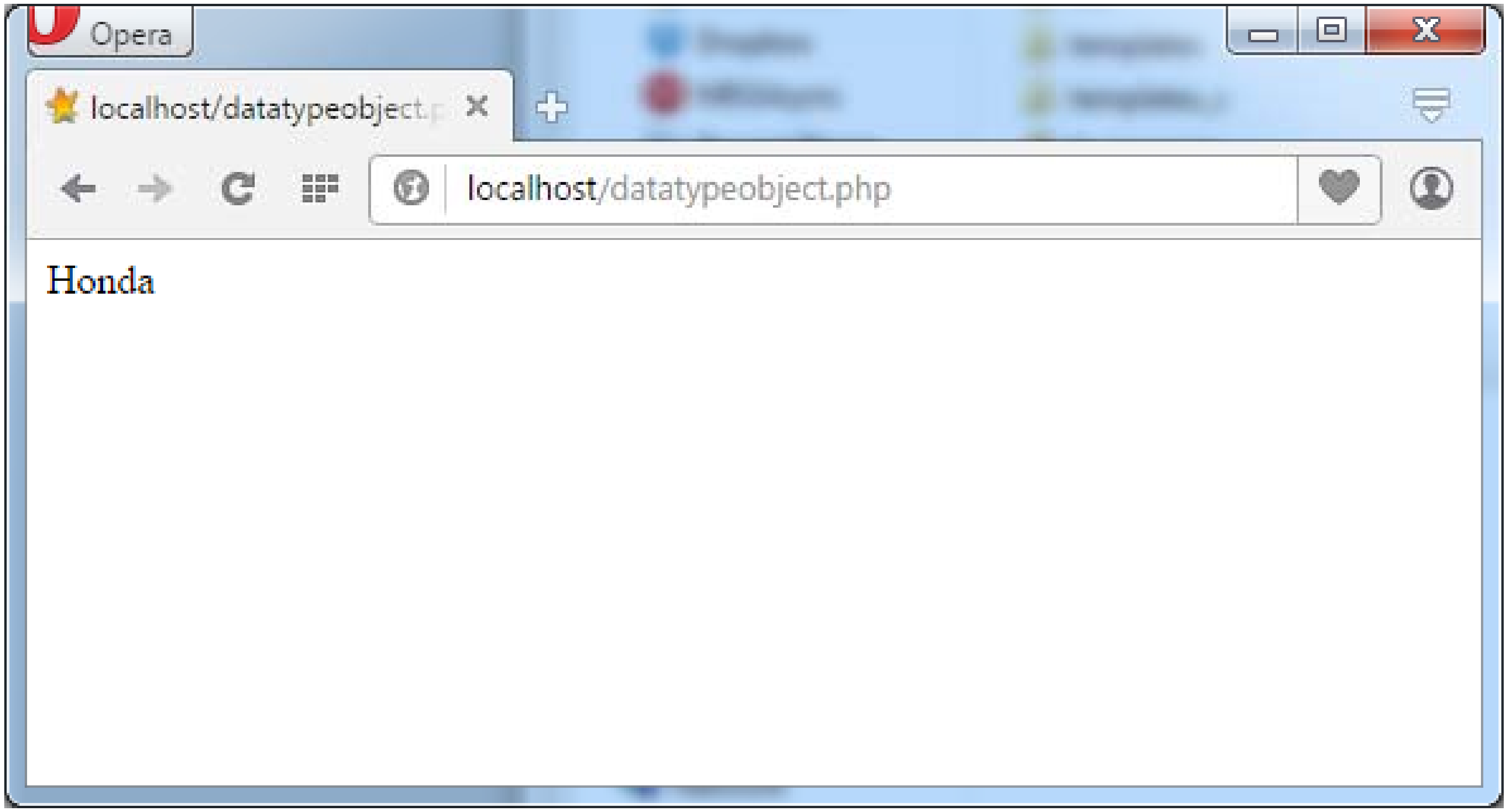
In PHP, an object must be explicitly declared.

First we must declare a class of object. For this, we use the class keyword. A class is a structure that can contain properties and methods:

Example

```
<?php
class Car {
    function Car() {
        $this->model = “Honda";
    }
}
// create an object
$fahad = new Car();

// show object properties
echo $fahad->model;
?>
```


Opera
localhost/datatypeobject.p
localhost/datatypeobject.php

```
Honda
```

PHP NULL Value

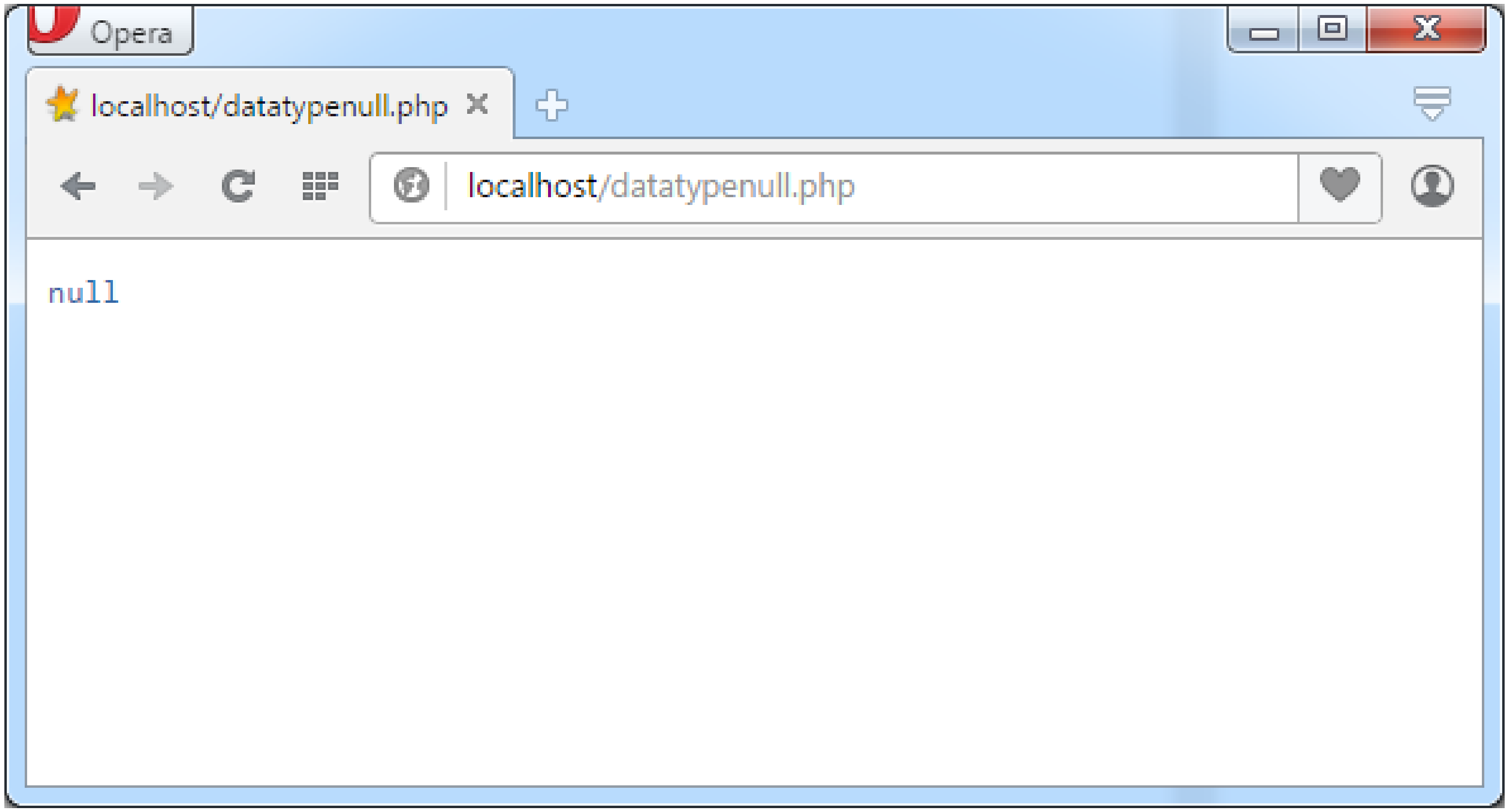
Null is a special data type which can have only one value: NULL.

A variable of data type NULL is a variable that has no value assigned to it.

- **Tip**: If a variable is created without a value, it is automatically assigned a value of NULL.
- Variables can also be emptied by setting the value to NULL:

Example

```
<?php
$x = "Hello world!";
$x = null;
var_dump($x);
?>
```


Opera
localhost/datatypenull.php
localhost/datatypenull.php

```
null
```

PHP is a Loosely Typed Language

- PHP automatically converts the variable to the correct data type, depending on its value.
- In other languages such as C, C++, and Java, the programmer must declare the name and type of the variable before using it.

Example

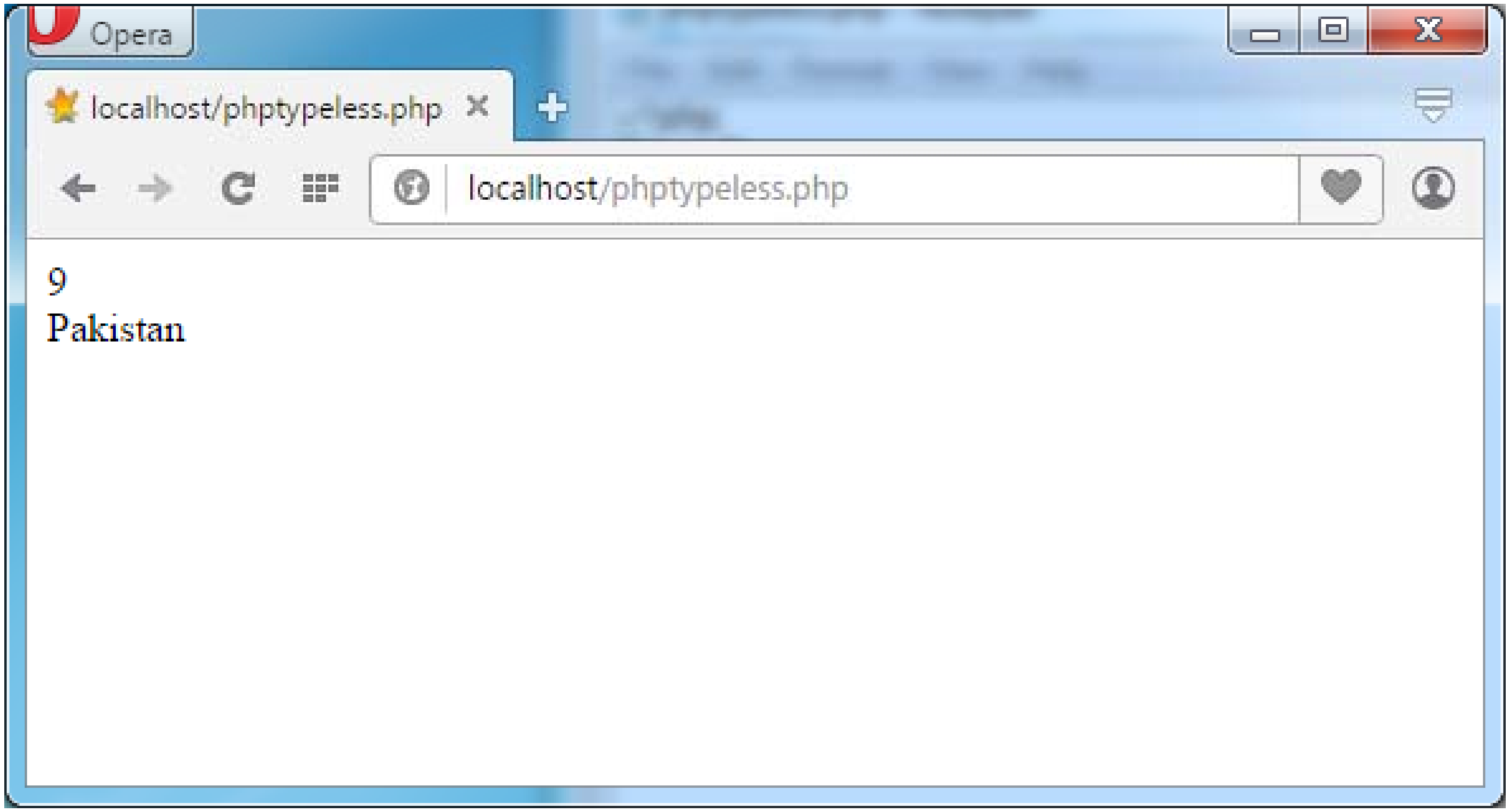
- <?php
  $x = 5;
  $y = 4;
  echo $x + $y;

- $x = “Pakistan”

- Echo $x

-

  ?>


Opera
localhost/phptypeless.php
localhost/phptypeless.php

```
9
Pakistan
```

PHP Variables Scope

In PHP, variables can be declared anywhere in the script.

The scope of a variable is the part of the script where the variable can be referenced/used.

PHP has three different variable scopes:

- local
- global
- static

Global and Local Scope

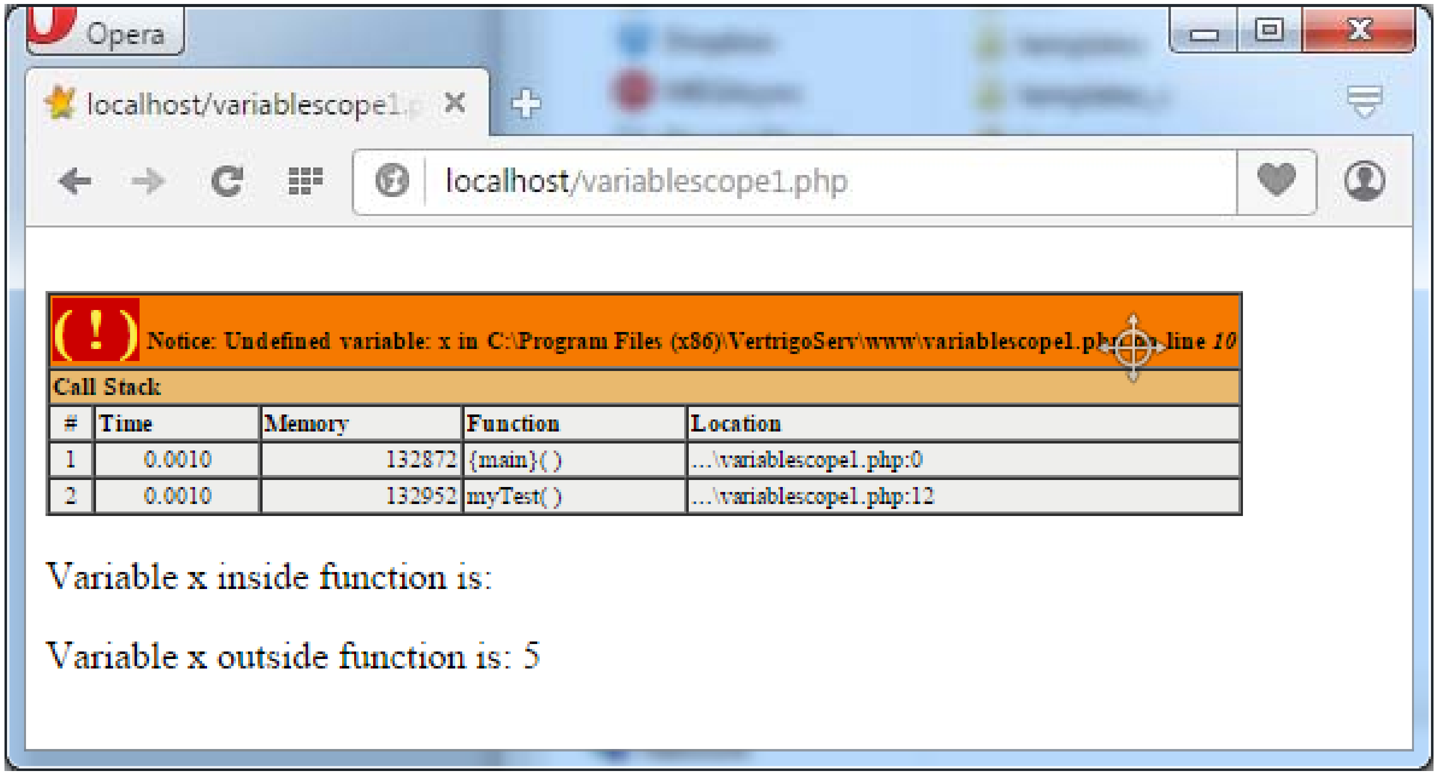
A variable declared **outside** a function has a GLOBAL SCOPE and can only be accessed outside a function:

Example

```
<?php
$x = 5; // global scope

function myTest() {
    // using x inside this function will generate an
error
    echo "<p>Variable x inside function is:
$x</p>";
}
myTest();

echo "<p>Variable x outside function is:
$x</p>";
?>
```



```
( ! ) Notice: Undefined variable: x in C:\Program Files (x86)\VertrigoServ\www\variablescope1.php on line 10
Call Stack
#  Time    Memory  Function   Location
1  0.0010  132872  {main}( )  ...\variablescope1.php:0
2  0.0010  132952  myTest( )  ...\variablescope1.php:12
```

Variable x inside function is:

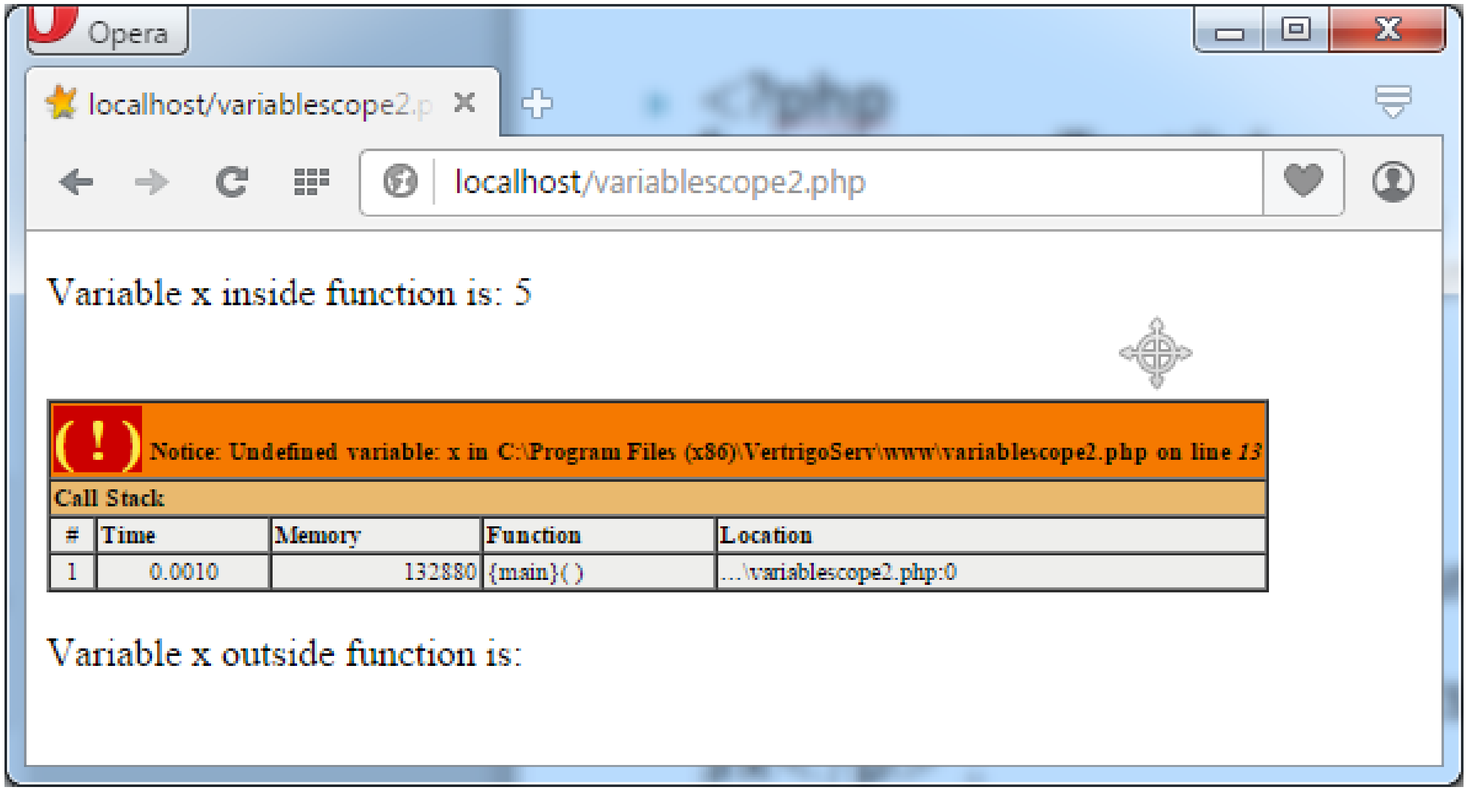
Variable x outside function is: 5

A variable declared **within** a function has a LOCAL SCOPE and can only be accessed within that function:

Example

```
<?php
function myTest() {
    $x = 5; // local scope
    echo "<p>Variable x inside function is:
$x</p>";
}
myTest();

// using x outside the function will generate an
error
echo "<p>Variable x outside function is:
$x</p>";
?>
```



```
Variable x inside function is: 5
```

(!) Notice: Undefined variable: x in C:\Program Files (x86)\VertrigoServ\www\variablescope2.php on line *13*				
Call Stack				
#	Time	Memory	Function	Location
1	0.0010	132880	{main}( )	...\variablescope2.php:0

```
Variable x outside function is:
```

PHP The global Keyword

The global keyword is used to access a global variable from within a function.

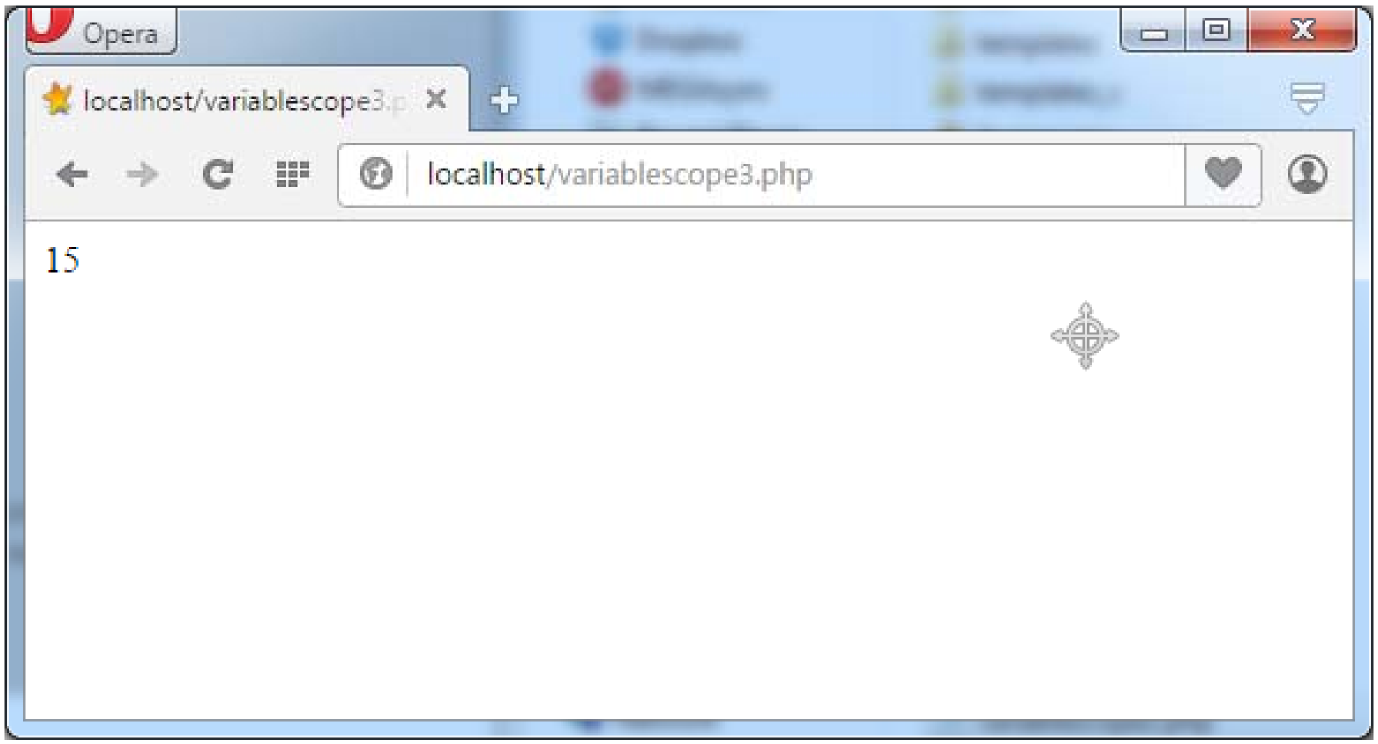
To do this, use the global keyword before the variables (inside the function):

Example

```
<?php
$x = 5;
$y = 10;

function myTest() {
    global $x, $y;
    $y = $x + $y;
}

myTest();
echo $y; // outputs 15
?>
```


Opera
localhost/variablescope3.p
localhost/variablescope3.php

```
15
```

PHP The static Keyword

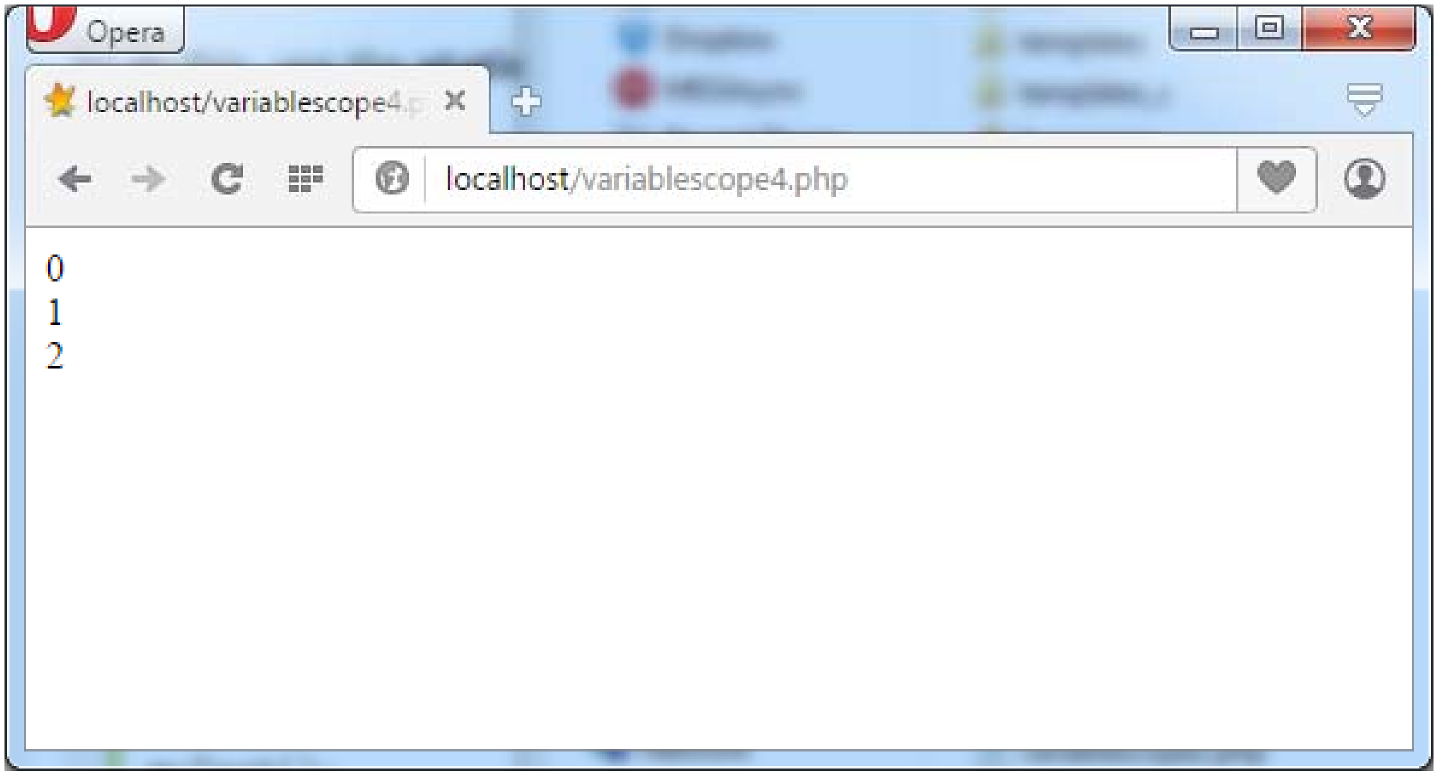
Normally, when a function is completed/executed, all of its variables are deleted. However, sometimes we want a local variable NOT to be deleted. We need it for a further job.

To do this, use the **static** keyword when you first declare the variable:

Example

```php
<?php
function myTest() {
    static $x = 0;
    echo $x;
    $x++;
}

myTest();
myTest();
myTest();
?>
```


Opera
localhost/variablescope4.p
localhost/variablescope4.php

```
0
1
2
```

PHP Strings

A string is a sequence of characters, like "Hello world!".

PHP String Functions

Get The Length of a String

The PHP strlen() function returns the length of a string.

The example below returns the length of the string "Hello world!":

```
<?php
echo strlen("Hello world!"); // outputs 12
?>
```

Count The Number of Words in a String

The PHP str_word_count() function counts the number of words in a string:

```
<?php
	echo str_word_count("Hello world!");
// outputs 2
?>
```

Reverse a String

The PHP strrev() function reverses a string:

```
<?php
echo strrev("Hello world!");
// outputs !dlrow olleH
?>
```

The PHP strpos() function searches for a specific text within a string.

If a match is found, the function returns the character position of the first match. If no match is found, it will return FALSE.

Example

The example below searches for the text "world" in the string "Hello world!":

```
<?php
echo strpos("Hello world!", "world");
// outputs 6
?>
```

Replace Text Within a String

- The PHP str_replace() function replaces some characters with some other characters in a string.
- The example below replaces the text "world" with "Dolly":

```
<?php
echo str_replace("world", "Dolly", "Hello
world!"); // outputs Hello Dolly!
?>
```

Complete PHP String Reference

- For a complete reference of all string functions, go to complete PHP String Reference.
- http://www.w3schools.com/php/php_ref_string.asp
- The PHP string reference contains description and example of use, for each function!

PHP Constants

A constant is an identifier (name) for a simple value. The value cannot be changed during the script.

A valid constant name starts with a letter or underscore (no $ sign before the constant name).

Note: Unlike variables, constants are automatically global across the entire script.

Create a PHP Constant

To create a constant, use the define() function.

Syntax

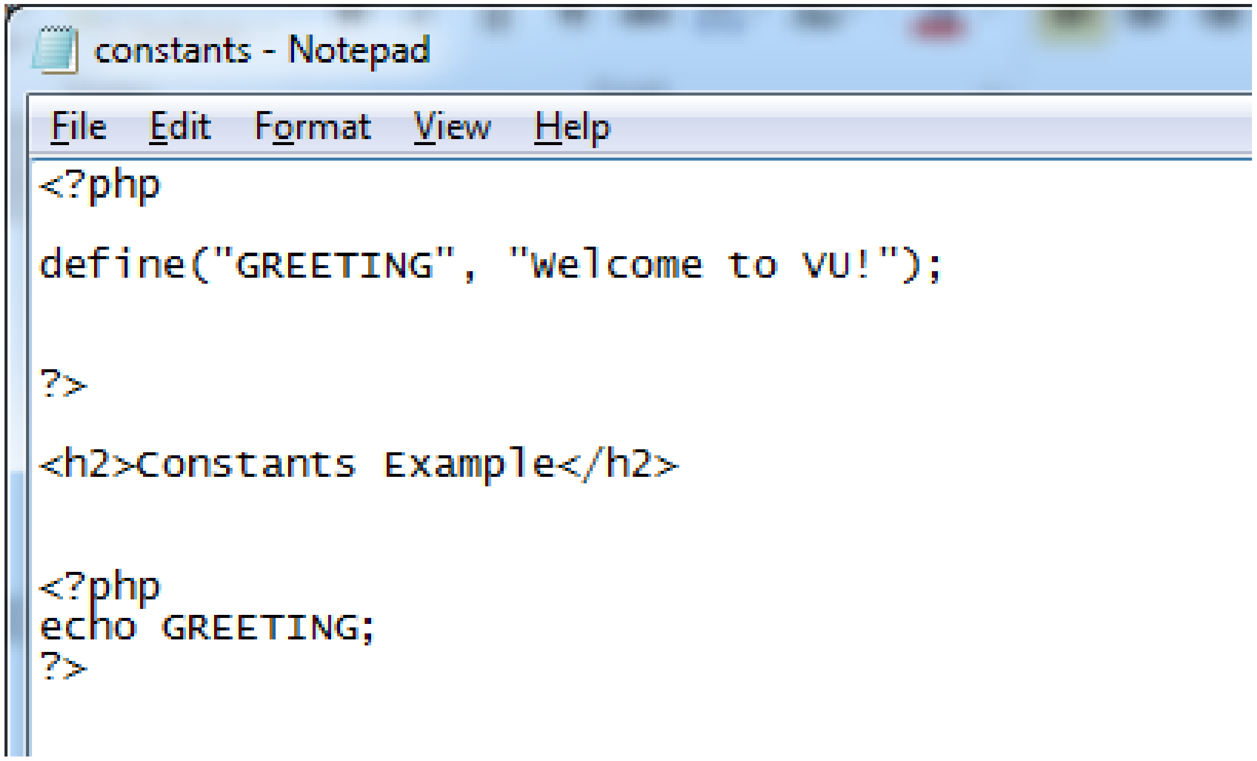
    define(*name*, *value*, *case-insensitive*)

Parameters:

- *name*: Specifies the name of the constant
- *value*: Specifies the value of the constant
- *case-insensitive*: Specifies whether the constant name should be case-insensitive. Default is false

Example

The example below creates a constant with a **case-sensitive** name:



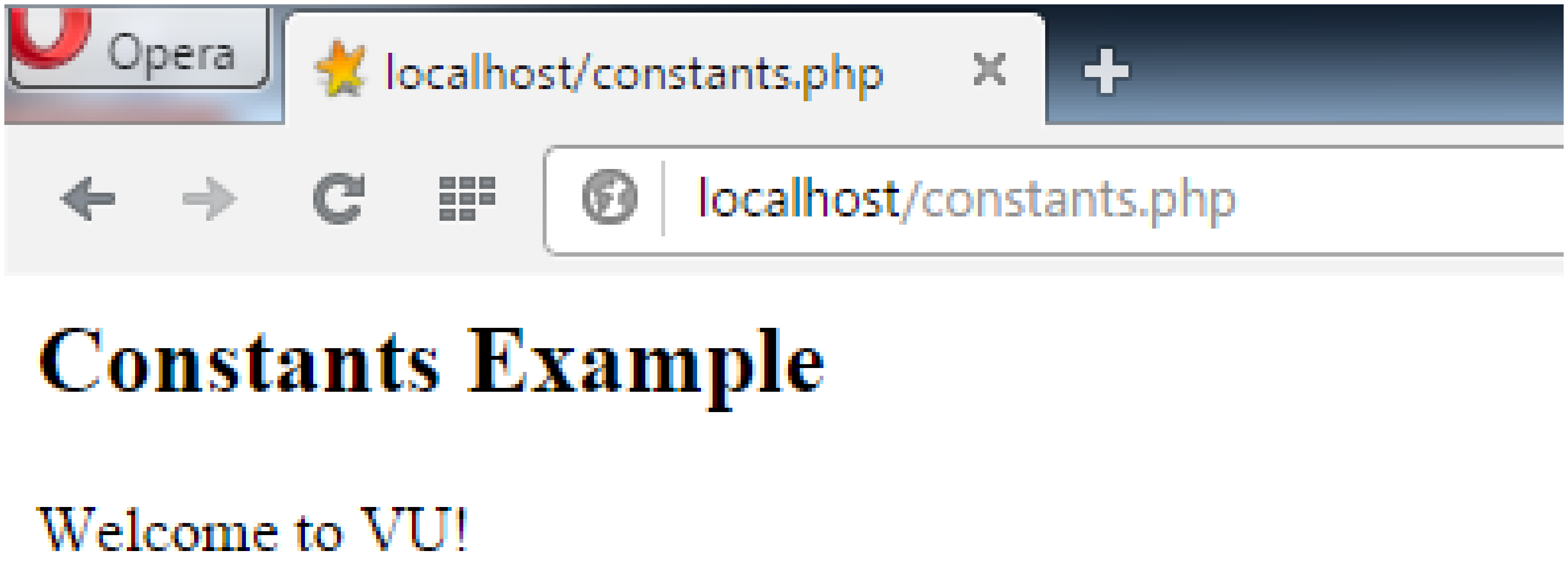
```
<?php

define("GREETING", "Welcome to VU!");


?>

<h2>Constants Example</h2>


<?php
echo GREETING;
?>
```



Constants Example

Welcome to VU!

PHP 5 Operators

Operators are used to perform operations on variables and values.

PHP divides the operators in the following groups:

- Arithmetic operators
- Assignment operators
- Comparison operators
- Increment/Decrement operators
- Logical operators
- String operators
- Array operators

PHP Arithmetic Operators

PHP Arithmetic Operators

Operator	Name	Example	Result
+	Addition	$x + $y	Sum of $x and $y
-	Subtraction	$x - $y	Difference of $x and $y
*	Multiplication	$x * $y	Product of $x and $y
/	Division	$x / $y	Quotient of $x and $y
%	Modulus	$x % $y	Remainder of $x divided by $y
**	Exponentiation	$x ** $y	Result of raising $x to the $y'th power (Introduced in PHP 5.6)

PHP Assignment Operators

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.

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

PHP Comparison Operators

PHP Comparison Operators

Operator	Name	Example	Result
==	Equal	$x == $y	Returns true if $x is equal to $y
===	Identical	$x === $y	Returns true if $x is equal to $y, and they are of the same type
!=	Not equal	$x != $y	Returns true if $x is not equal to $y
<>	Not equal	$x <> $y	Returns true if $x is not equal to $y

!==	Not identical	$x !== $y	Returns true if $x is not equal to $y, or they are not of the same type
>	Greater than	$x > $y	Returns true if $x is greater than $y
<	Less than	$x < $y	Returns true if $x is less than $y
>=	Greater than or equal to	$x >= $y	Returns true if $x is greater than or equal to $y
<=	Less than or equal to	$x <= $y	Returns true if $x is less than or equal to $y

PHP Increment & Decrement Operators

PHP Increment / Decrement Operators

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

PHP Logical Operators

Operator	Name	Example	Result
and	And	$x and $y	True if both $x and $y are true
or	Or	$x or $y	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

PHP String Operators

Operator	Name	Example	Result
.	Concatenation	$txt1 . $txt2	Concatenation of $txt1 and $txt2
.=	Concatenation assignment	$txt1 .= $txt2	Appends $txt2 to $txt1

if statement

if statement

if statement is used when we want to perform some action if the condition is true

if statement

<u>General PHP syntax</u>

Condition will be evaluated to true or false

```
if(condition)
        Code to be executed if condition is true
```

if statement (Cont…)

If there are multiple statements then statements should be enclosed within curly braces.

```
if(condition)
{
        Code to be executed if condition is true

}
```

Example 1

```
<?php
        $num1 = 10 ;
        $num2 = 5 ;
        if ($num1 > $num2)
            echo "Number 1 is greater than number 2" ;
?>
```

Example 1 (Cont..)

Out put of program:

```
Number 1 is greater than number 2 ;
```

Example 2

```
<?php
        $num1 = 5;
        $num2 = 10 ;
        if ($num1 > $num2)
            echo "Number 1 is greater than number 2" ;
?>
```

Nothing will be displayed on the screen as condition is false

if else statement

if else statement

This statement is used when we want to perform some action if the condition is true and another action if condition is not true or false

if else statement (Cont…)

General PHP syntax

```
if(condition)
        Code to be executed if condition is true
else
        Code to be executed if condition is false
```

Example 1

```
<?php

$num1 = 20 ;
$num2 = 10 ;
if ($num1 > $num2)
		echo “Number 1 is greater than number 2” ;
else
		echo “Number 2 is greater than number 1” ;

?>
```

Example 1(Cont..)

<u>Out put of program:</u>

```
Number 1 is greater than number 2
```

Example 2

```
<?php

$num1 = 10 ;
$num2 = 20 ;
if ($num1 > $num2)
		echo “Number 1 is greater than number 2” ;
else
		echo “Number 2 is greater than number 1” ;

?>
```

Example 2(Cont..)

<u>Out put of program:</u>

```
Number 2 is greater than number 1
```

if elseif else

if…elseif...else

We use this statement to execute some code if one of the several conditions is true.

if…elseif...else (Cont..)

General PHP syntax

```
if(condition)
        Code to be executed if condition is true
else if(condition)
        Code to be executed if condition is true

else if(condition)
        Code to be executed if condition is true
….
else
        Code to be executed if condition is false
```

Example 1

```
<?php
$num1 = 10 ;
$num2 = 20 ;

if ($num1 > $num2)
        echo “Number 1 is greater than number 2” ;
else if ($num1 < $num2)
        echo “Number 1 is less than  number 2” ;
else
        echo “Both numbers are equal” ;
 ?>
```

Example 1 (Cont..)

Out put of program:

```
Number 1 is less than number 2 ;
```

Example 2

```
 <?php
$num1 = 10 ;
$num2 = 10 ;

if ($num1 > $num2)
          echo “Number 1 is greater than number 2” ;
else if ($num1 < $num2)
          echo “Number 1 is less than  number 2” ;
else
          echo “Both numbers are equal” ;
 ?>
```

Example 2 (Cont..)

<u>Out put of program:</u>

```
Both numbers are equal
```

Multiple if statements

```
if(condition)
            Code to be executed if condition is true
if(condition)
            Code to be executed if condition is true
if(condition)
            Code to be executed if condition is true
if(condition)
            Code to be executed if condition is true
```

Multiple if statements vs. elseif statements

Multiple if statements	if … elseif… else
`if ($num1 > $num2)`	`if ($num1 > $num2)`
`        echo “Number 1 is greater than number 2” ;`	`        echo “Number 1 is greater than number 2” ;`
`if($num1 < $num2)`	`else if ($num1 < $num2)`
`        echo “Number 2 is greater than number 1” ;`	`        echo “Number 1 is less than  number 2” ;`
`if($num1 == $num2)`	`else`
`        echo “Both numbers are equal” ;`	`        echo “Both numbers are equal” ;`

Switch Statement

switch statement

We use this statement to select one of several blocks of code to be executed.

Used to avoid long blocks of *if elseif else* statements .

switch statement (Cont..)

```
switch(variable/expression)
{

case lable1:
                Code to be executed if variable/expression value matches label1
case label2:
                code to be executed if variable/expression value matches label2
....
default:
           code to be executed if variable/expression does not match any case
}
```

Example 1

```
<?php

$signal = 'Y' ;


switch($signal)
{

case 'R':
        echo "Red signal" ;
        break ;
case 'Y':
        echo "Yellow signal" ;
        break ;
```

Example 1 (Cont..)

```
case 'G':
        echo "Green signal" ;
        break ;

default:
        echo "Invalid signal" ;
}


?>
```

break statement is used at the end of each case to prevent the code from running into next case automatically

Example 1 (Cont..)

Output of program

```
Yellow signal
```

Example 2

```
<?php

$num = 2 ;

switch($num)
{

case '1':
          echo "Number 1" ;
          break ;
case '2':
          echo "Number 2" ;
          break ;
```

Example 2 (Cont..)

```
case '3':
          echo "Number 3" ;
          break ;

default:
          echo "Not a number from 1 to 3" ;

}


?>
```

Example 2 (Cont..)

<u>Output of program</u>

```
Number 2
```

for loop

for loop

Loops are repetition structures that are used to perform the same task again and again.

for loops are used to execute same block of code again and again for the specified number of times.

If we know the exact iteration of loop then for loop must be used.

for loop

General PHP syntax

```
for(initialization; condition; increment/decrement)
{
    // loop body
}
```

for loop

for (initialization; condition; increment/decrement)

In the first part, we set the value of loop counter and it is initialized only once at the start of loop

for loop

for (initialization; condition; increment/decrement)

In the second part, we set the condition for the continuation/termination of loop

In each iteration, if the condition evaluates to true then loop continues otherwise if it evaluates to false then loop terminates

for loop

for (initialization; condition; increment/decrement)

In third part, we increment/decrement the value of loop counter

At the end of each iteration, third part is evaluated

Example 1

```
for ($i = 0; $i<5;  $i++)
{
    echo “value of i is $i <br>” ;
}
```

Iterations	Condition ($i<5)	Loop body Output	Increment $i++
1	0<5 (True)	0	1
2	1<5 (True)	1	2
3	2<5 (True)	2	3
4	3<5 (True)	3	4
5	4<5 (True)	4	5
6	5<5 (False)	Loop terminates	

Example 2

The following example will output the square of numbers from 0 to 5.

```
<?php

$square = 0 ;
for($i = 0 ; $i<=5; $i++)
{
   $square = $i*$i ;
   echo “Square of $i is $square  <br>” ;
}

?>
```

Example 2 (Cont..)

Output of program

```
Square of 0 is 0
Square of 1 is 1
Square of 2 is 4
Square of 3 is 9
Square of 4 is 16
Square of 5 is 25
```

Example 3

The following example will output the square of numbers in descending order.

```
<?php
$square = 0 ;
for($i = 5 ; $i>= 0; $i--)
{
   $square = $i*$i ;
   echo “Square of $i is $square ” ;
}
?>
```

Example 3 (Cont..)

Output of program

```
Square of 5 is 25
Square of 4 is 16
Square of 3 is 9
Square of 2 is 4
Square of 1 is 1
Square of 0 is 0
```

while loop

while loop

while loops are repetition structures that are used to perform the same task again and again till the satisfaction of certain condition.

The loop continues to run as long as the condition is true and terminates if the condition is false.

If we want to repeat the process till the satisfaction of certain condition and the exact iteration of loop is not known then while loop must be used.

while loop

General PHP syntax:

```
while(condition)
{
    // code to be executed
}
```

Example 1

```
$i = 0 ;
while($i<5)
{
   echo "value of i is $i <br>" ;
   $i++ ;
}
```

Iterations	Condition ($i<5)	Loop body Output	Increment $i++
1	0<5 (True)	0	1
2	1<5 (True)	1	2
3	2<5 (True)	2	3
4	3<5 (True)	3	4
5	4<5 (True)	4	5
6	5<5 (False)	Loop terminates	

Example 2

The following example will output the square of numbers as long as the value of variable i is less than or equal to 5.

```
<?php
$square = 0 ;
$i = 0 ;
while($i<=5)
{
   $square = $i*$i ;
   echo “Square of $i is $square  <br>” ;
   $i++ ;
}
?>
```

Example 2 (Cont..)

Output of program

```
Square of 0 is 0
Square of 1 is 1
Square of 2 is 4
Square of 3 is 9
Square of 4 is 16
Square of 5 is 25
```

do while loop

do while loop

In do while loop, first the body of loop is executed and then loop is repeated till the satisfaction of certain condition.

In each iteration of loop, body of loop is executed and then condition is checked for continuation or termination of loop.

If there is a situation in which a loop must be executed at least once then a do-while loop must be used.

do while loop

General PHP syntax:

```
do
{
  // code to be executed

} while(condition) ;
```

Example 1

```
$i = 0 ;
do
{
   echo “value of i is $i <br>” ;
   $i++ ;
} while($i<5) ;
```

Iterations	Loop body Output	Increment $i++	Condition ($i<5)
1	0	1	1<5 (True)
2	1	2	2<5 (True)
3	2	3	3<5 (True)
4	3	4	4<5 (True)
5	4	5	5<5 (False)

Here loop terminates

Example 2

The following example will output the square of numbers as long as the value of variable i is less than or equal to 5.

```
<?php
$square = 0 ;
$i = 0;

do
{
   $square = $i*$i ;
   echo “Square of $i is $square <br> ” ;
   $i++ ;
} while($i<=5) ;
?>
```

Example 2 (Cont..)

Output of program

```
Square of 0 is 0
Square of 1 is 1
Square of 2 is 4
Square of 3 is 9
Square of 4 is 16
Square of 5 is 25
```

foreach loop

foreach loop

foreach loop is a looping structure that is used to loop through arrays.

For each iteration of loop, the value of current element of array is automatically assigned to a variable and then array pointer is moved by 1.

foreach loop

General PHP syntax:

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

Example 1

The following example will print the values of each element of array using foreach loop.

```
<?php

$a = array(1, 2, 3, 4, 5) ;

foreach ($a as $value)
{
	echo $value . “<br/>” ;
}
?>
```

Example 2

The following example will calculate the square of each element of array using foreach loop.

```
<?php

$a = array(1, 2, 3, 4, 5) ;

foreach ($a as $value)
{
        $square = $value * $value ;
        echo $square. "<br>" ;
}
?>
```

PHP Include Statement

PHP include statement

- The include statement takes all the text/code/markup that exists in the specified file and copies it into the file that uses the include statement.

- Including files is very useful when you want to include the same PHP, HTML, or text on multiple pages of a website.

Syntax

```
include 'filename';
```

PHP include Example

Assume we have a standard footer file called "footer.php", that looks like this:

```
<?php

echo "<p>Copyright; 1990-" . date("Y") . "
ABC.com</p>";

?>
```

PHP include Example (Continued)

To include the footer file in a page, use the include statement:

```
<html>
 <body>

<h1>Welcome to my home page!</h1>
 <p>Some text.</p>
<p>Some more text.</p>

<?php include 'footer.php';?>

</body>
 </html>
```

PHP Date and Time

PHP Date and Time

- The PHP date() function is used to format a date and/or a time.

The PHP Date() Function

- The PHP date() function formats a timestamp to a more readable date and time.

Syntax

date(format,timestamp)

Parameter	Description
format	Required. Specifies the format of the timestamp
timestamp	Optional. Specifies a timestamp. Default is the current date and time

Note that the PHP date() function will return the current date/time of the server!

Get a Simple Date

- The required format parameter of the date() function specifies how to format the date (or time).

- Here are some characters that are commonly used for dates:
  - d - Represents the day of the month (01 to 31)
  - m - Represents a month (01 to 12)
  - Y - Represents a year (in four digits)
  - l (lowercase 'L') - Represents the day of the week

- Other characters, like"/", ".", or "-" can also be inserted between the characters to add additional formatting.

Example

- The example below formats today's date in three different ways:

```
<?php
echo "Today is " . date("Y/m/d") . "<br>";
echo "Today is " . date("Y.m.d") . "<br>";
echo "Today is " . date("Y-m-d") . "<br>";
 echo "Today is " . date("l");
?>
```

Automatic Copyright Year

- Use the date() function to automatically update the copyright year on your website:

Example

```
&copy; 2010-<?php echo date("Y")?>
```

Get a Simple Time

- Here are some characters that are commonly used for times:
  - h - 12-hour format of an hour with leading zeros (01 to 12)
  - i - Minutes with leading zeros (00 to 59)
  - s - Seconds with leading zeros (00 to 59)
  - a - Lowercase Ante meridiem and Post meridiem (am or pm)

Example

- The example below outputs the current time in the specified format:

```
<?php
echo "The time is " . date("h:i:sa");
?>
```

Get Your Time Zone

- If the time you got back from the code is not the right time, it's probably because your server is in another country or set up for a different timezone.
- So, if you need the time to be correct according to a specific location, you can set a timezone to use.

Get Your Time Zone (Example)

- The example below sets the timezone to "Asia/Karachi", then outputs the current time in the specified format:

Example

```
<?php
date_default_timezone_set("Asia/Karachi");
echo "The time is " . date("h:i:sa");
?>
```

PHP Cookies

What is a Cookie?

- A cookie is often used to identify a user. A cookie is a small file that the server embeds on the user's computer.
- Each time the same computer requests a page with a browser, it will send the cookie too.
- With PHP, you can both create and retrieve cookie values.

Create Cookies With PHP

A cookie is created with the setcookie() function.

Syntax

```
setcookie(name, value, expire, path, domain, secure, httponly);
```

Only the name parameter is required. All other parameters are optional.

Note: The setcookie() function must appear BEFORE the <html> tag.

Example

```
<?php
$cookie_name = "user";
$cookie_value = "Ahmed";
setcookie($cookie_name, $cookie_value, time() + (86400
* 30), "/"); // 86400 = 1 day
?>
<html><body>
<?php
if(!isset($_COOKIE[$cookie_name])) {
    echo "Cookie named '" . $cookie_name . "' is not set!";
} else {
    echo "Cookie '" . $cookie_name . "' is set!<br>";
    echo "Value is: " . $_COOKIE[$cookie_name];}
?>
</body></html>
```

The value of the cookie is automatically URL encoded when sending the cookie, and automatically decoded when received (to prevent URL encoding, use setrawcookie() instead).

Delete a Cookie

To delete a cookie, use the setcookie() function with an expiration date in the past:

Example

```
<?php
// set the expiration date to one hour ago
setcookie("user", "", time() - 3600,"/");
?>
<html><body>
<?php
echo "Cookie 'user' is deleted.";
?>
</body></html>
```

PHP Sessions

What is a Session?

- A session is a way to store information (in variables) to be used across several pages.
- Unlike a cookie, the information is not stored on the users computer.
- When you work with an application, you open it, do some changes, and then you close it. This is much like a Session.
- On the internet there is one problem: the web server does not know who you are, because the HTTP address doesn't maintain state.
- Session variables solve this problem by storing user information to be used across multiple pages (e.g. username etc.). By default, session variables last until the user closes the browser.
- Session variables hold information about one single user, and are available to all pages in one application.

Start a PHP Session

- A session is started with the session_start() function.
- Session variables are set with the PHP global variable: $_SESSION.
- Now, let's create a new page called "session_demo1.php". In this page, we start a new PHP session and set some session variables:

Example

```
<?php
// Start the session
session_start();
?>
<!DOCTYPE html>
<html><body>
<?php
// Set session variables
$_SESSION["favcolor"] = "green";
$_SESSION["favanimal"] = "cat";
echo "Session variables are set.";
?>
</body></html>
```

Note: The session_start() function must be the very first thing in your document. Before any HTML tags.

Get PHP Session Variable Values

- Next, we create another page called "session_demo2.php". From this page, we will access the session information we set on the first page ("session_demo1.php").
- Session variables are not passed individually to each new page, instead they are retrieved from the session we open at the beginning of each page (session_start()).
- All session variable values are stored in the global $_SESSION variable

Example

```
<?php
session_start();
?>
<!DOCTYPE html>
<html>
 <body>

<?php
// Echo session variables that were set on previous page
echo "Favorite color is " . $_SESSION["favcolor"] . ".<br>";
echo "Favorite animal is " . $_SESSION["favanimal"] . ".";
?>

 </body>
</html>
```

Example 2

- Another way to show all the session variable values for a user session is to run the following code:

```
<?php
session_start();
?>
<!DOCTYPE html>
<html>
<body>

<?php
 print_r($_SESSION);
?>

</body>
</html>
```

How does it work?

- Most sessions set a user-key on the user's computer that looks something like this: 765487cf34ert8dede5a562e4f3a7e12.
- Then, when a session is opened on another page, it scans the computer for a user-key.
- If there is a match, it accesses that session, if not, it starts a new session.

How does it work?

- Most sessions set a user-key on the user's computer that looks something like this: 765487cf34ert8dede5a562e4f3a7e12.
- Then, when a session is opened on another page, it scans the computer for a user-key.
- If there is a match, it accesses that session, if not, it starts a new session.

Destroy a PHP Session

To remove all global session variables and destroy the session, use session_unset() and session_destroy():

Example

```
<?php
session_start();
?>
<!DOCTYPE html>
<html><body>

<?php
// remove all session variables
session_unset();

// destroy the session
session_destroy();
?>

</body></html>
```

PHP Filters

Why Use Filters?

Filters are used for

- Validating data = Determine if the data is in proper form.
- Sanitizing data = Remove any illegal character from the data.
- Many web applications receive external input. External input/data can be:
  - User input from a form
  - Cookies
  - Web services data
  - Server variables
  - Database query results

- You should always validate external data!
- Invalid submitted data can lead to security problems and break your webpage!
- By using PHP filters you can be sure your application gets the correct input!

PHP filter_var() Function

- The filter_var() function both validate and sanitize data.

- The filter_var() function filters a single variable with a specified filter. It takes two pieces of data:
  - The variable you want to check
  - The type of check to use

Sanitize a String

- The following example uses the filter_var() function to remove all HTML tags from a string:

Example

```
<?php
$str = "<h1>Hello World!</h1>";
$newstr = filter_var($str,
FILTER_SANITIZE_STRING);
echo $newstr;
?>
```

Sanitize and Validate a URL

- The following example uses the filter_var() function to first remove all illegal characters from a URL, then check if $url is a valid URL:

Example

```
<?php
$url = "http://www.vu.edu.pk";

// Remove all illegal characters from a url
$url = filter_var($url, FILTER_SANITIZE_URL);

// Validate url
if (!filter_var($url, FILTER_VALIDATE_URL) === false) {
    echo("$url is a valid URL");
} else {
    echo("$url is not a valid URL");
}
?>
```

Sanitize and Validate an Email Address

- The following example uses the filter_var() function to first remove all illegal characters from the $email variable, then check if it is a valid email address:

Example

```
<?php
$email = "john.doe@example.com";

// Remove all illegal characters from email
$email = filter_var($email, FILTER_SANITIZE_EMAIL);

// Validate e-mail
if (!filter_var($email, FILTER_VALIDATE_EMAIL) === false) {
    echo("$email is a valid email address");
} else {
    echo("$email is not a valid email address");
}
?>
```

Validate an Integer

- The following example uses the filter_var() function to check if the variable $int is an integer. If $int is an integer, the output of the code above will be: "Integer is valid". If $int is not an integer, the output will be: "Integer is not valid":

Example

```
<?php
$int = 100;

if (!filter_var($int, FILTER_VALIDATE_INT) === false) {
    echo("Integer is valid");
} else {
    echo("Integer is not valid");
}
?>
```

Tip: filter_var() and Problem With 0

- In the previous example, if $int was set to 0, the function above will return "Integer is not valid". To solve this problem, use the code below:

Example

```
<?php
$int = 0;

if (filter_var($int, FILTER_VALIDATE_INT) === 0 || !filter_var($int,
FILTER_VALIDATE_INT) === false) {
    echo("Integer is valid");
} else {
    echo("Integer is not valid");
}
?>
```

Validate an IP Address

- The following example uses the filter_var() function to check if the variable $ip is a valid IP address:

Example

```
<?php
$ip = "127.0.0.1";

if (!filter_var($ip, FILTER_VALIDATE_IP) === false) {
    echo("$ip is a valid IP address");
} else {
    echo("$ip is not a valid IP address");
}
?>
```

Introduction to MySQL Database

Introduction to Databases

- **Data base**
  - A database is an organized collection of data.
- **Database management system (DBMS)**
  - It is a computer software application, designed to allow the definition, creation, querying, update, and administration of databases.
  - **Example**

    Some examples of DBMS are :
    - MySQL
    - PostgreSQL
    - Microsoft SQL Server
    - Oracle
    - Sybase

MySQL Database

- A free, fast, reliable, easy-to-use, multi-user multi-threaded relational database system.
- SQL (Structured Query Language) is use in MYSQL database system.
- It is freely available and released under GPL (GNU General Public License).
- Officially pronounced “my Ess Que Ell” (not my sequel).

Use of Database tools

Database tools

- There are many tools available for handling MY SQL databases such as:

- **XAMPP** :consisting mainly of the Apache HTTP Server, MySQL database
- **WAMP**:consisting of the Apache web server, OpenSSL for SSL support, MySQL database

- For our course we will use **WAMP**

PhpMyAdmin

- Is a FREE software tool written in PHP intended to handle the administration of MySQL over the Internet.
- PhpMyAdmin supports a wide range of operations with MySQL, the most frequently used being the managing of databases, tables, fields, relations, indexes, users, permissions.
- You still have the ability to execute SQL statements directly as well.

How to use MY SQL with XAMPP server

Privileges and Security

Privileges and Security

- These mysql database tables contain grant information:
- **user:** User accounts, global privileges, and other non-privilege columns.
- **db:** Database-level privileges.
- **host:** Obsolete.
- **tables_priv:** Table-level privileges.
- **columns_priv:** Column-level privileges.
- **procs_priv**: Stored procedure and function privileges.

Privileges and Security

- Used in PHP code,
-
```
<?php
$servername = "localhost";
$username = "username";
$password = "password";

// Create connection
$conn = mysqli_connect($servername, $username,
$password);

// Check connection
if (!$conn) {
    die("Connection failed: " . mysqli_connect_error());
}
echo "Connected successfully";
?>
```

MySQL Connect

MySQL Connect

- Earlier versions of PHP used the MySQL extension. PHP version 5 and after that work with a MySQL database using:
- **MySQLi extension** (the "i" stands for improved)
- **PDO (PHP Data Objects)**

MySQL Connect

- PDO will work on 12 different database systems, where as MySQLi will only work with MySQL databases.
- Three ways of working with PHP and MySQL:
  - i- MySQLi (object-oriented)
  - ii- MySQLi (procedural)
  - iii- PDO

Note: We will use Procedural method in our examples.

MySQL Create DB

MySQL Create DB

The CREATE DATABASE statement is used to create a database in MySQL.

Examples

```
<?PHP   $servername = "localhost";

$username = “root";

$password = “";
```

MySQL Create DB (example continued)

```
// Create connection
$conn = mysqli_connect($servername, $username,
$password);

// Check connection
if (!$conn) {
    die("Connection failed: " . mysqli_connect_error());}


        // Create database
$sql = "CREATE DATABASE mystudent";
if (mysqli_query($conn, $sql)) {
    echo "Database created successfully";
} else {
    echo "Error creating database: " . mysqli_error($conn);
}   ?>
```

MySQL Create Table

MySQL Create Table

- The CREATE TABLE statement is used to create a table in MySQL.

- We will create a table named “Student", with five columns: “s_id", "firstname", "lastname", "email" and "reg_date":

SQL Statement:-

```
CREATE TABLE Student (
s_id INT(6) UNSIGNED AUTO_INCREMENT PRIMARY KEY,
firstname VARCHAR(30) NOT NULL,
lastname VARCHAR(30) NOT NULL,
email VARCHAR(50),
reg_date TIMESTAMP
)
```

MYSQL Create Table

Example:-

```
<?php
$servername = "localhost";
$username = “root";
$password = “";
$dbname = "mystudent";

// Create connection
$conn = mysqli_connect($servername, $username,
$password, $dbname);
// Check connection
if (!$conn) {
    die("Connection failed: " . mysqli_connect_error());
}
```

MySQL Create Table (Example Continued)

```
// sql to create table
$sql = "CREATE TABLE student (
id INT(6) UNSIGNED AUTO_INCREMENT PRIMARY KEY,
firstname VARCHAR(30) NOT NULL,
lastname VARCHAR(30) NOT NULL,
email VARCHAR(50),
reg_date TIMESTAMP
)";

if (mysqli_query($conn, $sql)) {
    echo "Table student created successfully";
} else {
    echo "Error creating table: " . mysqli_error($conn);
}

mysqli_close($conn);
?>
```

MySQL Insert Data

MySQL Insert Data

- When a database and a table have been created, we can start adding data in them.

Here are some syntax rules to follow:

- The SQL query must be quoted in PHP
- String values inside the SQL query must be quoted
- Numeric values must not be quoted
- The word NULL must not be quoted
- The INSERT INTO statement is used to add new records to a MySQL table

MySQL Insert Data

- The INSERT INTO statement is used to add new records to a MySQL table:

SQL Statement: INSERT INTO table_name
(column1, column2, column3,...)
VALUES (value1, value2, value3,...)

Note: If a column is AUTO_INCREMENT (like the "id" column) or TIMESTAMP (like the "reg_date" column), it is no need to be specified in the SQL query; MySQL will automatically add the value.

MySQL Insert Data

Example

```
<?php
$servername = "localhost";
$username = “root";
$password = “";
$dbname = "mystudent";

// Create connection
$conn = mysqli_connect($servername, $username,
$password, $dbname);
// Check connection
if (!$conn) {
    die("Connection failed: " . mysqli_connect_error());
}
```

MySQL Insert Data

Example(continued)

```
$sql = "INSERT INTO student (firstname, lastname,
email)
VALUES (‘Ali', ‘Wali', ‘ali@example.com')";

if (mysqli_query($conn, $sql)) {
    echo "New record created successfully";
} else {
    echo "Error: " . $sql . "<br>" .
mysqli_error($conn);
}

mysqli_close($conn);
?>
```

MySQL Get Last ID

MySQL Get Last ID

- we perform an INSERT or UPDATE on a table with an AUTO_INCREMENT field, we can get the ID of the last inserted/updated record immediately.
- In the table “student", the "id" column is an AUTO_INCREMENT field:

Example

```
<?php
$servername = "localhost";
$username = “root";
$password = “";
$dbname = "mystudent";
```

MySQL Get Last ID

```
// Create connection
$conn = mysqli_connect($servername, $username, $password,
$dbname);
// Check connection
if (!$conn) {
    die("Connection failed: " . mysqli_connect_error());
}

$sql = "INSERT INTO student (firstname, lastname, email)
VALUES (‘Nadeem', ‘Muhammad', ‘Nadeem@example.com')";

if (mysqli_query($conn, $sql)) {
    $last_id = mysqli_insert_id($conn);
    echo "New record created successfully. Last inserted ID is: " .
$last_id;
} else {
    echo "Error: " . $sql . "<br>" . mysqli_error($conn);
}

mysqli_close($conn);
?>
```

MySQL Insert in Multiple.

MySQL Insert in Multiple.

- Multiple SQL statements must be executed with the mysqli_multi_query() function.
- The following examples add three new records to the “student" table:

```
<?php
$servername = "localhost";
$username = “root";
$password = “";
$dbname = "mystudent";
```

MySQL Insert in Multiple.

```
$conn = mysqli_connect($servername, $username,
$password, $dbname);
// Check connection
if (!$conn) {
    die("Connection failed: " . mysqli_connect_error());
}

$sql = "INSERT INTO student (firstname, lastname, email)
VALUES (‘Babar', ‘Ali', ‘Babar@example.com');";
$sql .= "INSERT INTO student (firstname, lastname, email)
VALUES (‘sara', ‘khan', ‘sara@example.com');";
$sql .= "INSERT INTO  student (firstname, lastname,
email)
VALUES (‘Mahmood', ‘Zakir', ‘Mahmood@example.com')";
```

MySQL Insert in Multiple.

```
if (mysqli_multi_query($conn, $sql)) {
    echo "New records created successfully";
} else {
    echo "Error: " . $sql . "<br>" .
mysqli_error($conn);
}

mysqli_close($conn);
?>
```

MySQL Prepared.

MySQL Prepared.

- A prepared statement is a feature used to execute the same (or similar) SQL statements repeatedly with high efficiency.
- Prepared statements basically work like this:
- Prepare: An SQL statement template is created and sent to the database. Certain values are left unspecified, called parameters (labeled "?"). Example: INSERT INTO student VALUES(?, ?, ?)
- The database parses, compiles, and performs query optimization on the SQL statement template, and stores the result without executing it
- Execute: At a later time, the application binds the values to the parameters, and the database executes the statement.

MySQL Prepared.

- The application may execute the statement as many times as it wants with different values
- Compared to executing SQL statements directly, prepared statements have two main advantages:

- Prepared statements reduces parsing time as the preparation on the query is done only once (although the statement is executed multiple times)
- Bound parameters minimize bandwidth to the server as you need send only the parameters each time, and not the whole query
- Prepared statements are very useful against SQL injections, because parameter values, which are transmitted later using a different protocol, need not be correctly escaped. If the original statement template is not derived from external input, SQL injection cannot occur.

MySQL Prepared.

Example

```
<?php
$servername = "localhost";
$username = “root";
$password = "";
$dbname = "mystudent";

// Create connection
$conn = new mysqli($servername, $username, $password,
$dbname);

// Check connection
if ($conn->connect_error) {
    die("Connection failed: " . $conn->connect_error);
}
```

MySQL Prepared.

```
// prepare and bind
$stmt = $conn->prepare("INSERT INTO student
(firstname, lastname, email) VALUES (?, ?, ?)");
$stmt->bind_param("sss", $firstname, $lastname,
$email);

// set parameters and execute
$firstname = “Khalid";
$lastname = “Zia";
$email = “khakid_zia@example.com";
$stmt->execute();
```

MySQL Prepared.

```
$firstname = “Amna";
$lastname = “khan";
$email = “amna@example.com";
$stmt->execute();

$firstname = “Zanub";
$lastname = “ali";
$email = “zanub@example.com";
$stmt->execute();

echo "New records created successfully";

$stmt->close();
$conn->close();
?>
```

MySQL Select Data

MySQL Select Data

- The SELECT statement is used to select data from one or more tables:

Example

SELECT column_name(s) FROM table_name

we can use the * character to select ALL columns from a table:

Example

SELECT * FROM table_name

MySQL Select Data

Example

```
<?php
$servername = "localhost";
$username = “root";
$password = "";
$dbname = "mystudent";

// Create connection
$conn = mysqli_connect($servername, $username,
$password, $dbname);
// Check connection
if (!$conn) {
    die("Connection failed: " . mysqli_connect_error());
}
```

MySQL Select Data

```
$sql = "SELECT id, firstname, lastname FROM student";
$result = mysqli_query($conn, $sql);

if (mysqli_num_rows($result) > 0) {
    // output data of each row
    while($row = mysqli_fetch_assoc($result)) {
        echo "id: " . $row["id"]. " - Name: " .
$row["firstname"]. " " . $row["lastname"]. "<br>";
    }
} else {
    echo "0 results";
}

mysqli_close($conn);
?>
```

MySQL Delete Data

MySQL Delete Data

The DELETE statement is used to delete records from a table.

SQL Statement:-DELETE FROM table_name WHERE some_column = some_value.

Example:-

```
<?php
$servername = "localhost";
$username = "username";
$password = "password";
$dbname = "myDB";
```

MySQL Delete Data

Example (continued)

```
// Create connection
$conn = mysqli_connect($servername, $username, $password,
$dbname);
// Check connection
if (!$conn) {
    die("Connection failed: " . mysqli_connect_error());
}

// sql to delete a record
$sql = "DELETE FROM student WHERE id=3";

if (mysqli_query($conn, $sql)) {
    echo "Record deleted successfully";
} else {
    echo "Error deleting record: " . mysqli_error($conn);
}
mysqli_close($conn);
?>
```

MySQL Update Data

MySQL Update Data

- The UPDATE statement is used to update existing records in a table:
- SQL UPDATE Statement:- UPDATE table_name SET column1=value, column2=value2,... WHERE some_column=some_value

Example:-

```
<?php
$servername = "localhost";
$username = “root";
$password = "";
$dbname = "mystudent";
```

MySQL Update Data

Example(continued)

```
// Create connection
$conn = mysqli_connect($servername, $username, $password,
$dbname);
// Check connection
if (!$conn) {
    die("Connection failed: " . mysqli_connect_error());
}

$sql = "UPDATE student SET lastname='Ismaile'  WHERE id=2";

if (mysqli_query($conn, $sql)) {
    echo "Record updated successfully";
} else {
    echo "Error updating record: " . mysqli_error($conn);
}

mysqli_close($conn);
?>
```

MySQL Limit Data

MySQL Limit Data

- MySQL provides a LIMIT clause that is used to specify the number of records to return

Example:-

$sql = "SELECT * FROM student Orders LIMIT 05";

When the SQL query above is run, it will return the first 05 records.

PHP File Handling

PHP File Handling

- File handling is an important part of any web application. You often need to open and process a file for different tasks.

PHP Manipulating Files

- PHP has several functions for creating, reading, uploading, and editing files.

- **Be careful when manipulating files!**

  When you are manipulating files you must be very careful. You can do a lot of damage if you do something wrong.

  - **Common errors are**:
    - Editing the wrong file
    - Filling a hard-drive with garbage data
    - And deleting the content of a file by accident

PHP readfile() Function

PHP File Handling

PHP readfile() Function

- The readfile() function reads a file and writes it to the output buffer.
- Assume we have a text file called "**webdictionary.txt**", stored on the server, that looks like this:

```
AJAX = Asynchronous JavaScript and XML
CSS = Cascading Style Sheets
HTML = Hyper Text Markup Language
PHP = PHP Hypertext Preprocessor
SQL = Structured Query Language
SVG = Scalable Vector Graphics
XML = EXtensible Markup Language
```

PHP readfile() Function (Cont.)

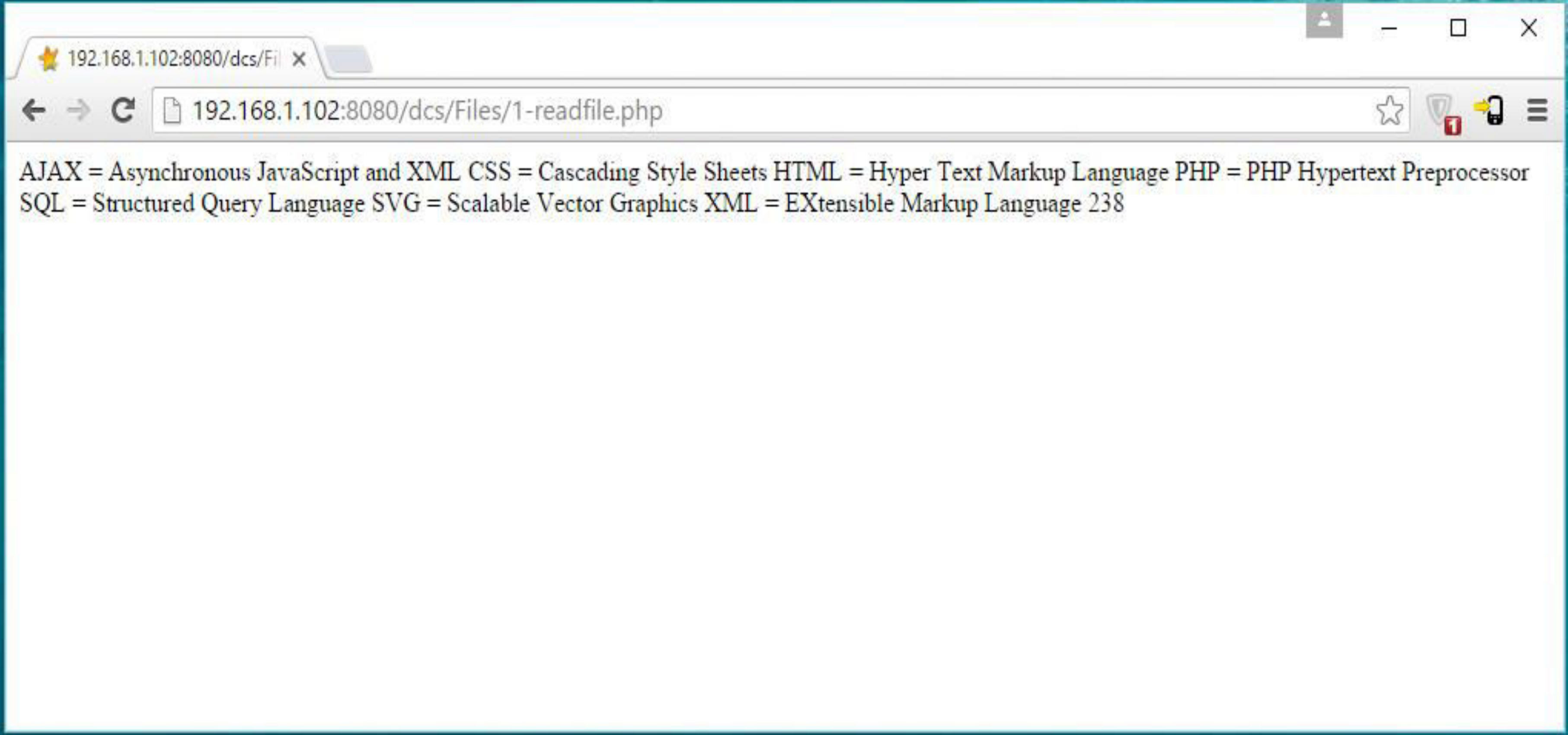
The PHP code to read the file and write it to the output buffer is as follows (the readfile() function returns the number of bytes read on success):

Example

```
<?php
echo readfile("webdictionary.txt");
?>
```

The readfile() function is useful if all you want to do is open up a file and read its contents.

Web Browser output


192.168.1.102:8080/dcs/Fi
192.168.1.102:8080/dcs/Files/1-readfile.php

```
AJAX = Asynchronous JavaScript and XML CSS = Cascading Style Sheets HTML = Hyper Text Markup Language PHP = PHP Hypertext Preprocessor
SQL = Structured Query Language SVG = Scalable Vector Graphics XML = EXtensible Markup Language 238
```

PHP Open File – fopen()

PHP File Handling

PHP Open File – fopen()

A better method to open files is with the fopen() function. This function gives you more options than the readfile() function.

We will use the text file, "**webdictionary.txt**", during the lessons:

AJAX = Asynchronous JavaScript and XML
CSS = Cascading Style Sheets
HTML = Hyper Text Markup Language
PHP = PHP Hypertext Preprocessor
SQL = Structured Query Language
SVG = Scalable Vector Graphics
XML = EXtensible Markup Language

fopen() function

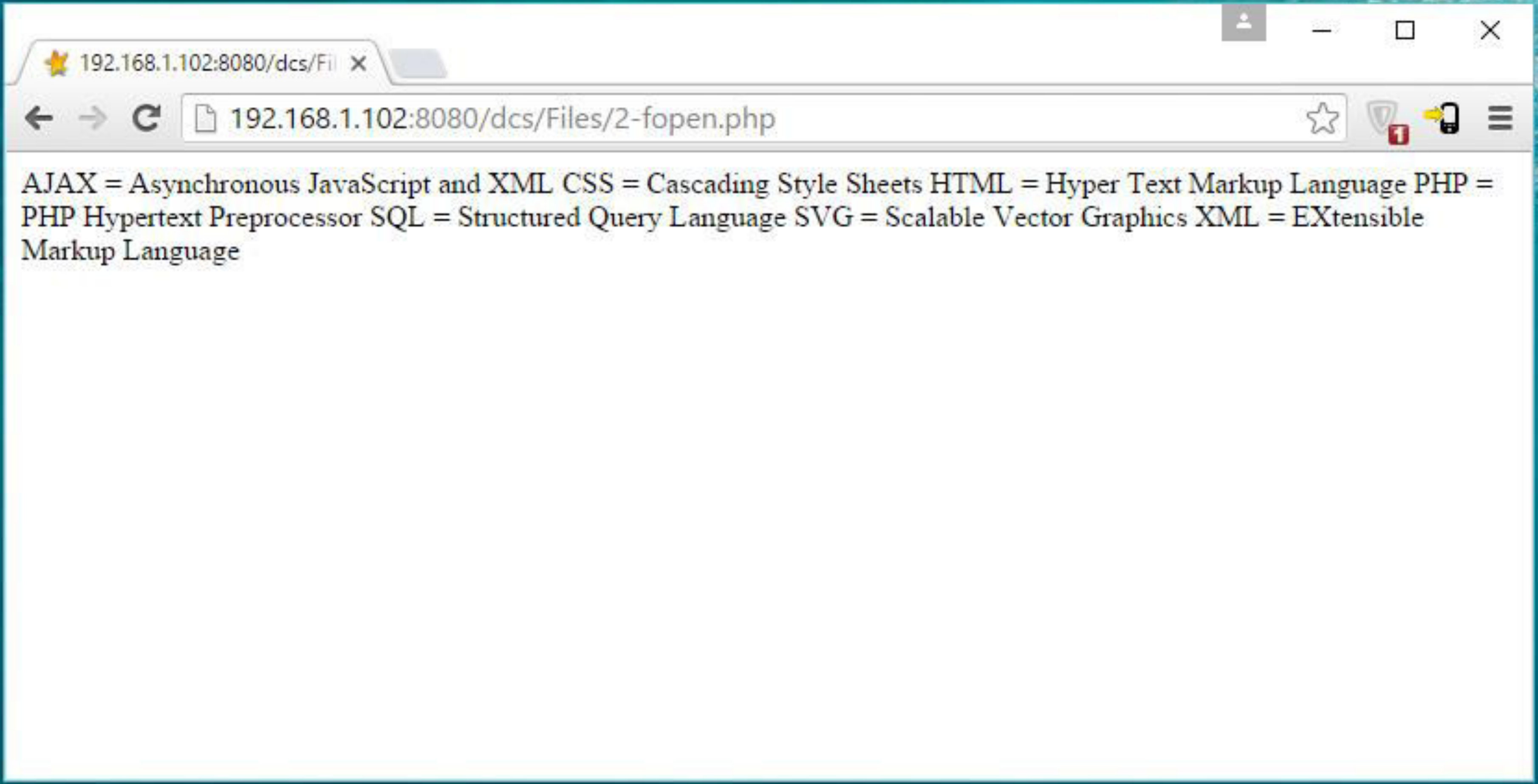
- fopen() function looks like this
  fopen("webdictionary.txt", "r")

- The first parameter of fopen() contains the name of the file to be opened and the second parameter specifies in which mode the file should be opened.

Example

```
<?php
$myfile = fopen("webdictionary.txt", "r") or
die("Unable to open file!");
echo
fread($myfile,filesize("webdictionary.txt"));
fclose($myfile);
?>
```

Web Browser Output


192.168.1.102:8080/dcs/Fi
192.168.1.102:8080/dcs/Files/2-fopen.php

```
AJAX = Asynchronous JavaScript and XML CSS = Cascading Style Sheets HTML = Hyper Text Markup Language PHP =
PHP Hypertext Preprocessor SQL = Structured Query Language SVG = Scalable Vector Graphics XML = EXtensible
Markup Language
```

Modes to open a file

The file may be opened in one of the following modes:

Modes	Description
r	**Open a file for read only.** File pointer starts at the beginning of the file
w	**Open a file for write only.** Erases the contents of the file or creates a new file if it doesn't exist. File pointer starts at the beginning of the file
a	**Open a file for write only.** The existing data in file is preserved. File pointer starts at the end of the file. Creates a new file if the file doesn't exist
x	**Creates a new file for write only.** Returns FALSE and an error if file already exists
r+	**Open a file for read/write.** File pointer starts at the beginning of the file
w+	**Open a file for read/write.** Erases the contents of the file or creates a new file if it doesn't exist. File pointer starts at the beginning of the file
a+	**Open a file for read/write.** The existing data in file is preserved. File pointer starts at the end of the file. Creates a new file if the file doesn't exist
x+	**Creates a new file for read/write.** Returns FALSE and an error if file already exists

PHP Read File – fread()

PHP File Handling

PHP Read File - fread()

The fread() function reads from an open file.

The following PHP code reads the "**webdictionary.txt**" file to the end:

```
fread($myfile,filesize("webdictionary.txt"));
```

The first parameter of fread() contains the name of the file to read from and the second parameter specifies the maximum number of bytes to read.

PHP Close File – fclose()

PHP File Handling

PHP Close File – fclose()

- The fclose() function is used to close an open file.
- It's a good programming practice to close all files after you have finished with them. You don't want an open file running around on your server taking up resources!
- The fclose() requires the name of the file (or a variable that holds the filename) we want to close:

Example

```
<?php
 $myfile = fopen("webdictionary.txt", "r");
// some code to be executed....
 fclose($myfile);
 ?>
```

PHP Read Single Line – fgets()

PHP File Handling

PHP Read Single Line – fgets()

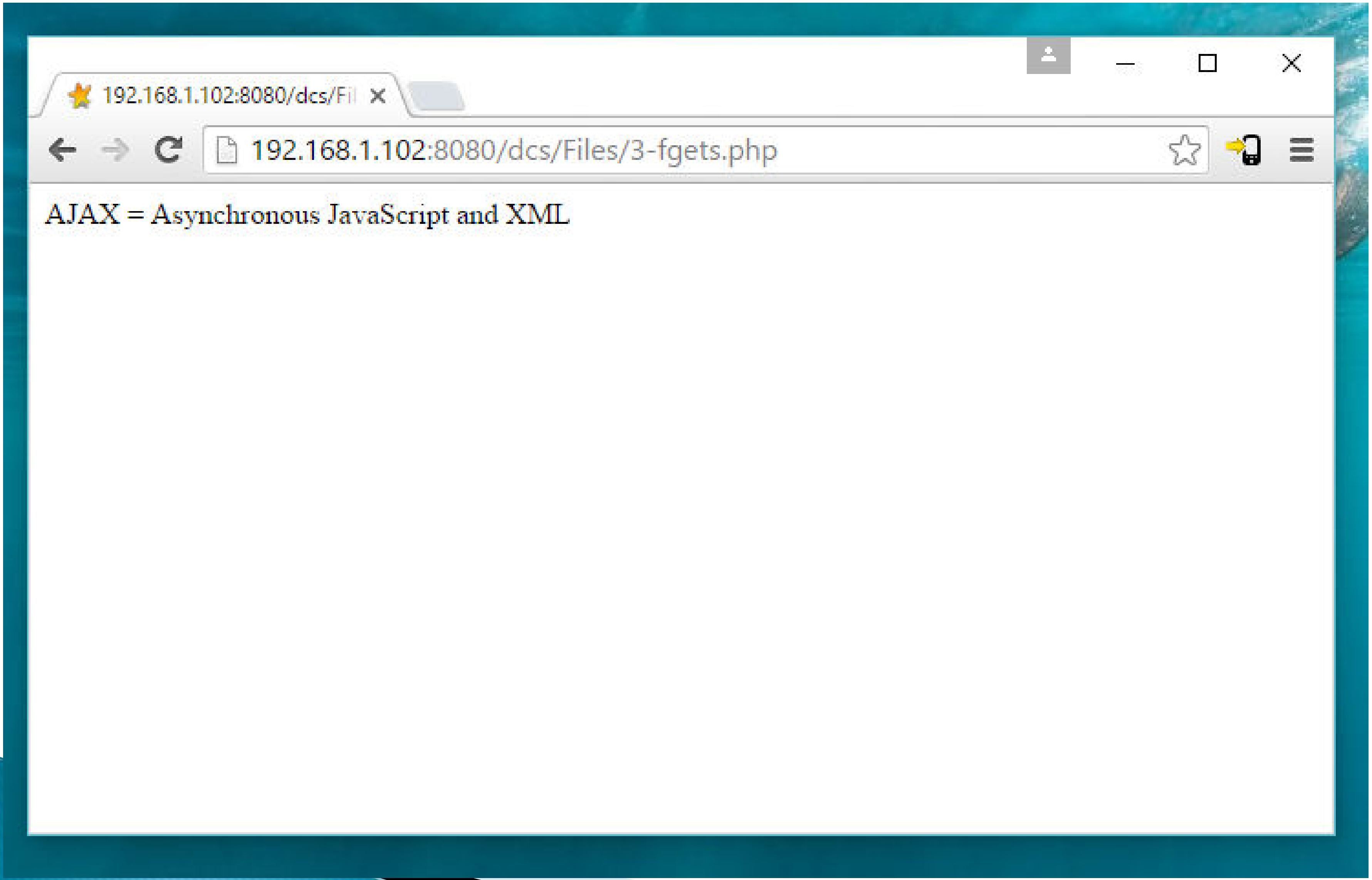
- The fgets() function is used to read a single line from a file.
- The example below outputs the first line of the "**webdictionary.txt**" file:

Example

```
<?php
$myfile = fopen("webdictionary.txt", "r") or die("Unable to
open file!");
echo fgets($myfile);
 fclose($myfile);
?>
```

Note: After a call to the fgets() function, the file pointer has moved to the next line.

Web Browser Output


192.168.1.102:8080/dcs/Fil
192.168.1.102:8080/dcs/Files/3-fgets.php

```
AJAX = Asynchronous JavaScript and XML
```

PHP Check End-Of-File - feof()

PHP File Handling

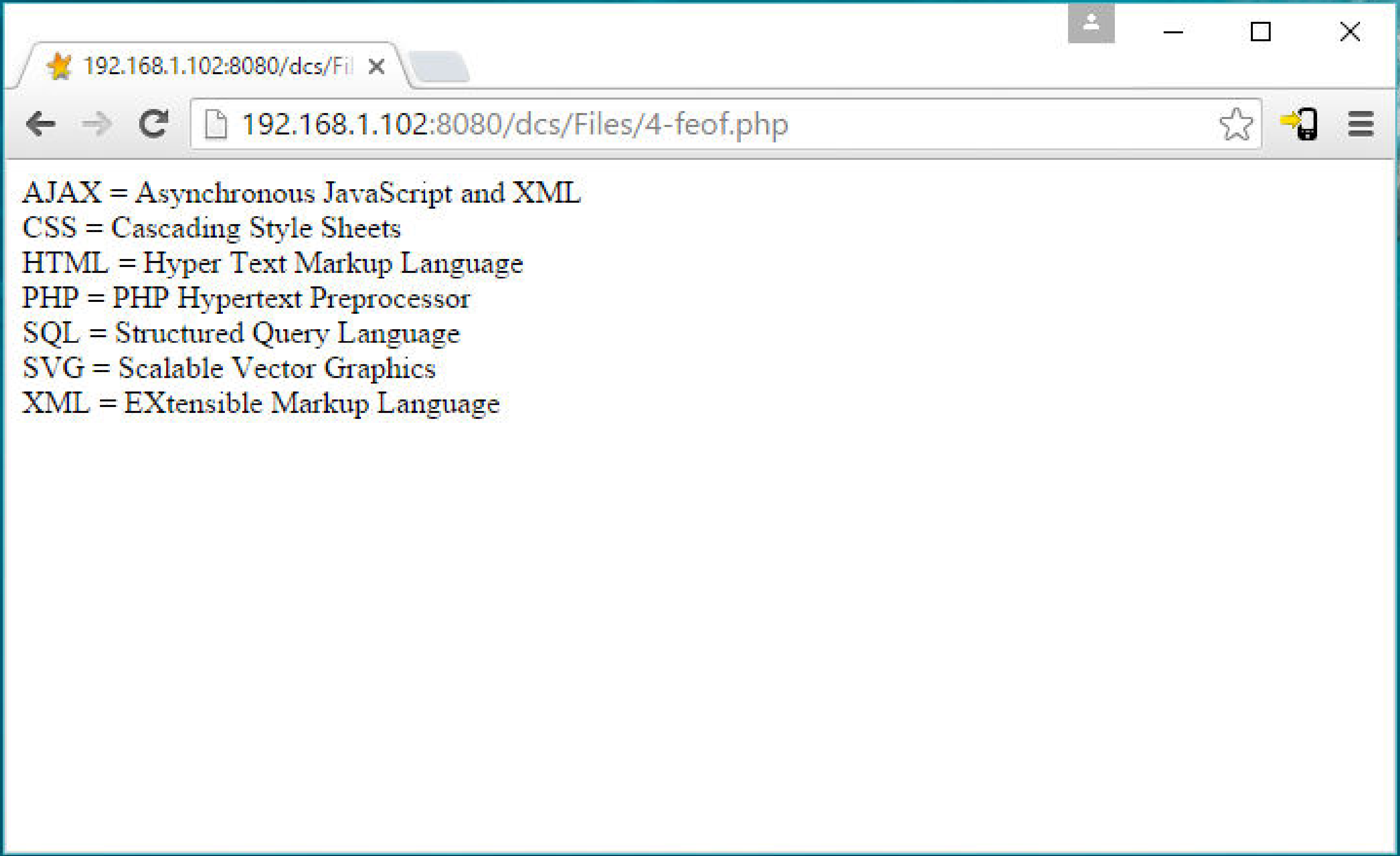
PHP Check End-Of-File - feof()

- The feof() function checks if the "end-of-file" (EOF) has been reached.
- The feof() function is useful for looping through data of unknown length.
- The example below reads the "**webdictionary.txt**" file line by line, until end-of-file is reached:

Example

```
<?php
$myfile = fopen("webdictionary.txt", "r") or die("Unable to open file!");
// Output one line until end-of-file
while(!feof($myfile)) {
  echo fgets($myfile) . "<br>";
}
fclose($myfile);
?>
```

Web Browser Output


192.168.1.102:8080/dcs/Fi
192.168.1.102:8080/dcs/Files/4-feof.php

```
AJAX = Asynchronous JavaScript and XML
CSS = Cascading Style Sheets
HTML = Hyper Text Markup Language
PHP = PHP Hypertext Preprocessor
SQL = Structured Query Language
SVG = Scalable Vector Graphics
XML = EXtensible Markup Language
```

PHP Read Single Character – fgetc()

PHP File Handling

PHP Read Single Character – fgetc()

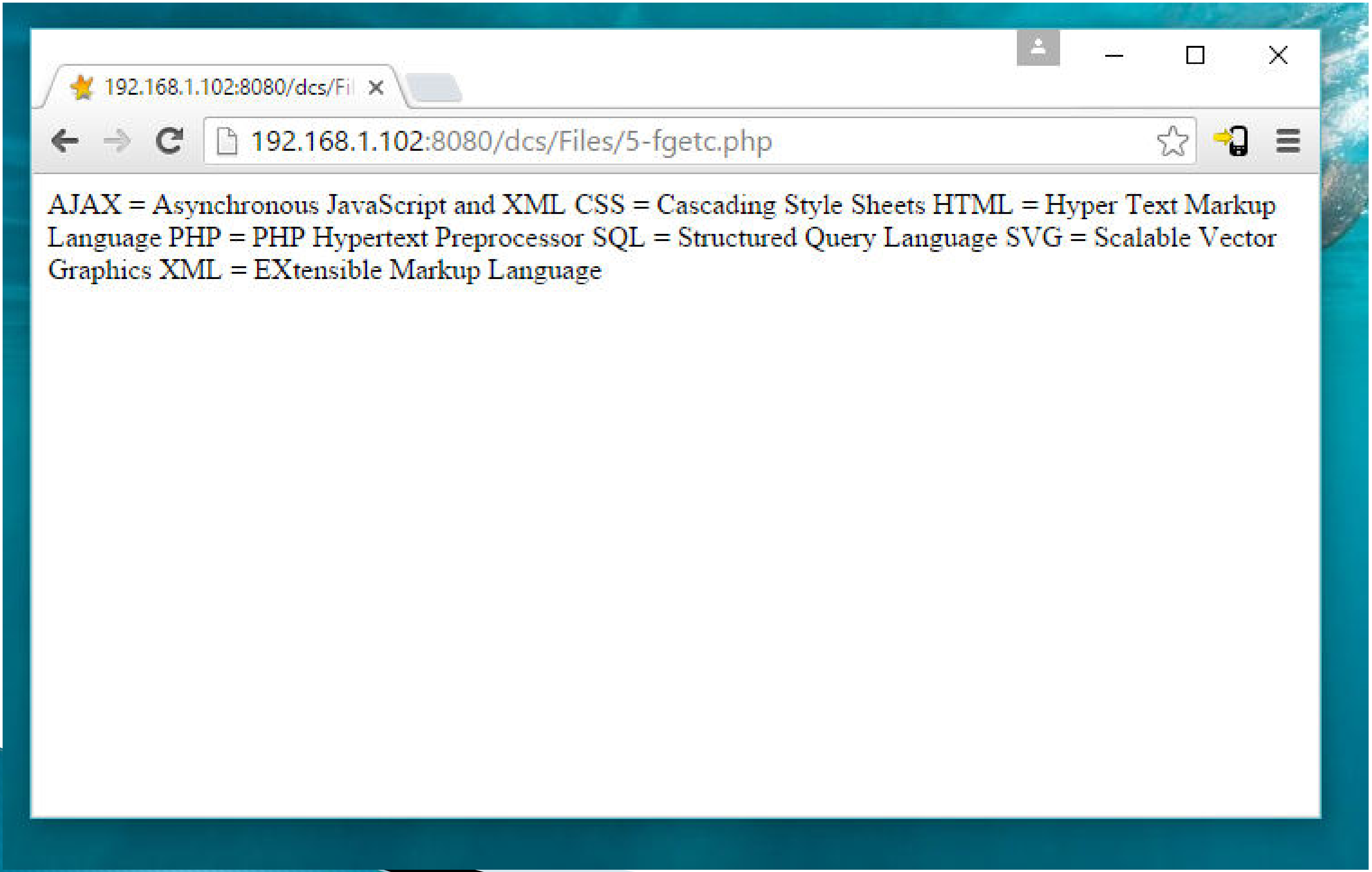
- The fgetc() function is used to read a single character from a file.
- The example below reads the "webdictionary.txt" file character by character, until end-of-file is reached:

Example

```
<?php
$myfile = fopen("webdictionary.txt", "r") or die("Unable to open file!");
// Output one character until end-of-file
while(!feof($myfile)) {
  echo fgetc($myfile);
}
fclose($myfile);
?>
```

Note: After a call to the fgetc() function, the file pointer moves to the next character.

Web Browser Output



192.168.1.102:8080/dcs/Fi
192.168.1.102:8080/dcs/Files/5-fgetc.php

```
AJAX = Asynchronous JavaScript and XML CSS = Cascading Style Sheets HTML = Hyper Text Markup
Language PHP = PHP Hypertext Preprocessor SQL = Structured Query Language SVG = Scalable Vector
Graphics XML = EXtensible Markup Language
```

Creating a Custom Exception Class

PHP Exception Handling

Creating a Custom Exception Class

- Creating a custom exception handler is quite simple.
- We simply create a special class with functions that can be called when an exception occurs in PHP.
- The class must be an extension of the exception class.
- The custom exception class inherits the properties from PHP's exception class and you can add custom functions to it.

Example

```
<?php
 class customException extends Exception {
   public function errorMessage() {
     //error message
     $errorMsg = 'Error on line '.$this->getLine().' in '.$this->getFile()
     .': <b>'.$this->getMessage().'</b> is not a valid E-Mail address';
     return $errorMsg;
   } }
$email = "someone@example...com";
 try {
   //check if
   if(filter_var($email, FILTER_VALIDATE_EMAIL) === FALSE) {
     //throw exception if email is not valid
     throw new customException($email);
   }
 }
catch (customException $e) {
   //display custom message
   echo $e->errorMessage(); } ?>
```

The new class is a copy of the old exception class with an addition of the errorMessage() function.
Since it is a copy of the old class, and it inherits the properties and methods from the old class, we can use the exception class methods like getLine() and getFile() and getMessage().

Example explained:

```
<?php
 class customException extends Exception {
  public function errorMessage() {
    //error message
    $errorMsg = 'Error on line '.$this->getLine().' in '.$this->getFile()
    .': <b>'.$this->getMessage().'</b> is not a valid E-Mail address';
    return $errorMsg;
  } }
$email = "someone@example...com";
 try {
  //check if
  if(filter_var($email, FILTER_VALIDATE_EMAIL) === FALSE) {
    //throw exception if email is not valid
    throw new customException($email);
  }
 }
catch (customException $e) {
  //display custom message
  echo $e->errorMessage(); } ?>
```

The **customException()** class is created as an extension of the old exception class. This way it inherits all methods and properties from the old exception class

Example explained:

```
<?php
 class customException extends Exception {
   public function errorMessage() {
     //error message
     $errorMsg = 'Error on line '.$this->getLine().' in '.$this->getFile()
     .': <b>'.$this->getMessage().'</b> is not a valid E-Mail address';
     return $errorMsg;
   } }
$email = "someone@example...com";
 try {
   //check if
   if(filter_var($email, FILTER_VALIDATE_EMAIL) === FALSE) {
     //throw exception if email is not valid
     throw new customException($email);
   }
 }
catch (customException $e) {
   //display custom message
   echo $e->errorMessage(); } ?>
```

The **errorMessage()** function is created. This function returns an error message if an e-mail address is invalid

Example explained:

```
<?php
 class customException extends Exception {
   public function errorMessage() {
     //error message
     $errorMsg = 'Error on line '.$this->getLine().' in '.$this->getFile()
     .': <b>'.$this->getMessage().'</b> is not a valid E-Mail address';
     return $errorMsg;
   } }
$email = "someone@example...com";
 try {
   //check if
   if(filter_var($email, FILTER_VALIDATE_EMAIL) === FALSE) {
     //throw exception if email is not valid
     throw new customException($email);
   }
 }
catch (customException $e) {
   //display custom message
   echo $e->errorMessage(); } ?>
```

The **$email** variable is set to a string that is not a valid e-mail address

Example explained:

```
<?php
 class customException extends Exception {
   public function errorMessage() {
     //error message
     $errorMsg = 'Error on line '.$this->getLine().' in '.$this->getFile()
     .': <b>'.$this->getMessage().'</b> is not a valid E-Mail address';
     return $errorMsg;
   } }
$email = "someone@example...com";
 try {
   //check if
   if(filter_var($email, FILTER_VALIDATE_EMAIL) === FALSE) {
     //throw exception if email is not valid
     throw new customException($email);
   }
 }
catch (customException $e) {
   //display custom message
   echo $e->errorMessage(); } ?>
```

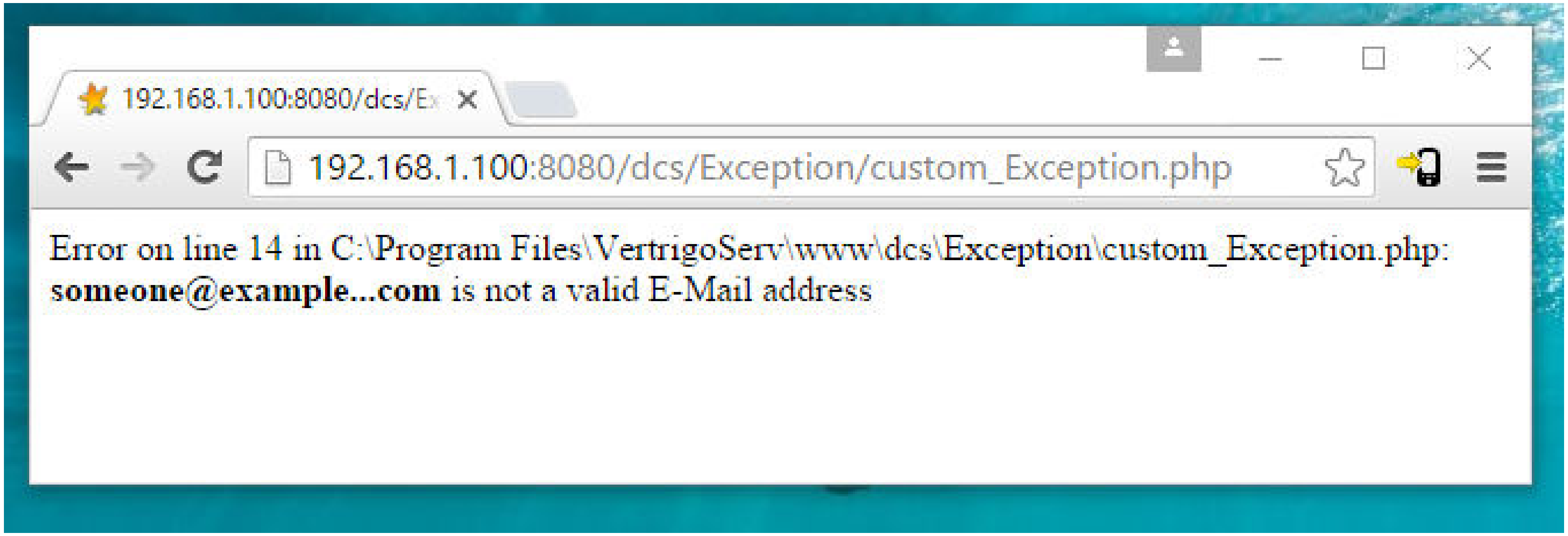
The "**try**" block is executed and an exception is thrown since the e-mail address is invalid

Example explained:

```
<?php
 class customException extends Exception {
   public function errorMessage() {
     //error message
     $errorMsg = 'Error on line '.$this->getLine().' in '.$this->getFile()
     .': <b>'.$this->getMessage().'</b> is not a valid E-Mail address';
     return $errorMsg;
   } }
$email = "someone@example...com";
 try {
   //check if
   if(filter_var($email, FILTER_VALIDATE_EMAIL) === FALSE) {
     //throw exception if email is not valid
     throw new customException($email);
   }
 }
catch (customException $e) {
   //display custom message
   echo $e->errorMessage(); } ?>
```

The "catch" block catches the exception and displays the error message

Web Browser Output


192.168.1.100:8080/dcs/Ex
192.168.1.100:8080/dcs/Exception/custom_Exception.php

```
Error on line 14 in C:\Program Files\VertrigoServ\www\dcs\Exception\custom_Exception.php:
someone@example...com is not a valid E-Mail address
```

PHP Create File – fopen()

PHP File Handling

PHP Create File - fopen()

- The fopen() function is also used to create a file. Maybe a little confusing, but in PHP, a file is created using the same function used to open files.
- If you use fopen() on a file that does not exist, it will create it, given that the file is opened for writing (w) or appending (a).
- The example below creates a new file called "testfile.txt". The file will be created in the same directory where the PHP code resides:

Example

```
$myfile = fopen("testfile.txt", "w")
```

PHP File Permissions
If you are having errors when trying to get this code to run, check that you have granted your PHP file access to write information to the hard drive.

PHP Write to File – fwrite()

PHP File Handling

PHP Write to File - fwrite()

- The fwrite() function is used to write to a file.
- The first parameter of fwrite() contains the name of the file to write to and the second parameter is the string to be written.
- The example below writes a couple of names into a new file called "newfile.txt":

Example

```
<?php
$myfile = fopen("newfile.txt", "w") or die("Unable to open
file!");
$txt = "John Doe ";
fwrite($myfile, $txt);
$txt = "Jane Doe ";
fwrite($myfile, $txt);
fclose($myfile);
?>
```

PHP Write to File - fwrite() (Cont.)

Example explained

- Notice that we wrote to the file "newfile.txt" twice. Each time we wrote to the file we sent the string $txt that first contained "John Doe" and second contained "Jane Doe". After we finished writing, we closed the file using the fclose() function.

If we open the "newfile.txt" file it would look like this:

John Doe Jane Doe

PHP Overwriting

- Now that "newfile.txt" contains some data we can show what happens when we open an existing file for writing. All the existing data will be ERASED and we start with an empty file.
- In the example below we open our existing file "newfile.txt", and write some new data into it:

Example

```
<?php
$myfile = fopen("newfile.txt", "w") or die("Unable to open
file!");
$txt = "Mickey Mouse ";
fwrite($myfile, $txt);
$txt = "Minnie Mouse ";
fwrite($myfile, $txt);
fclose($myfile);
?>
```

PHP Overwriting (Cont.)

If we now open the "newfile.txt" file, both John and Jane have vanished, and only the data we just wrote is present:

```
Mickey Mouse Minnie Mouse
```

PHP File Upload

PHP File Handling

Configure The "php.ini" File

- First, ensure that PHP is configured to allow file uploads.
- In your "php.ini" file, search for the file_uploads directive, and set it to On:

```
file_uploads = On
```

Create The HTML Form

- Next, create an HTML form that allow users to choose the image file they want to upload:

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

<form action="upload.php" method="post"
enctype="multipart/form-data">
Select image to upload:
<input type="file" name="fileToUpload" id="fileToUpload">
<input type="submit" value="Upload Image"
name="submit">
</form>

</body>
</html>
```

Create The HTML Form (Cont.)

Some rules to follow for the HTML form are:

- Make sure that the form uses method="post"
- The form also needs the following attribute: enctype="multipart/form-data". It specifies which content-type to use when submitting the form

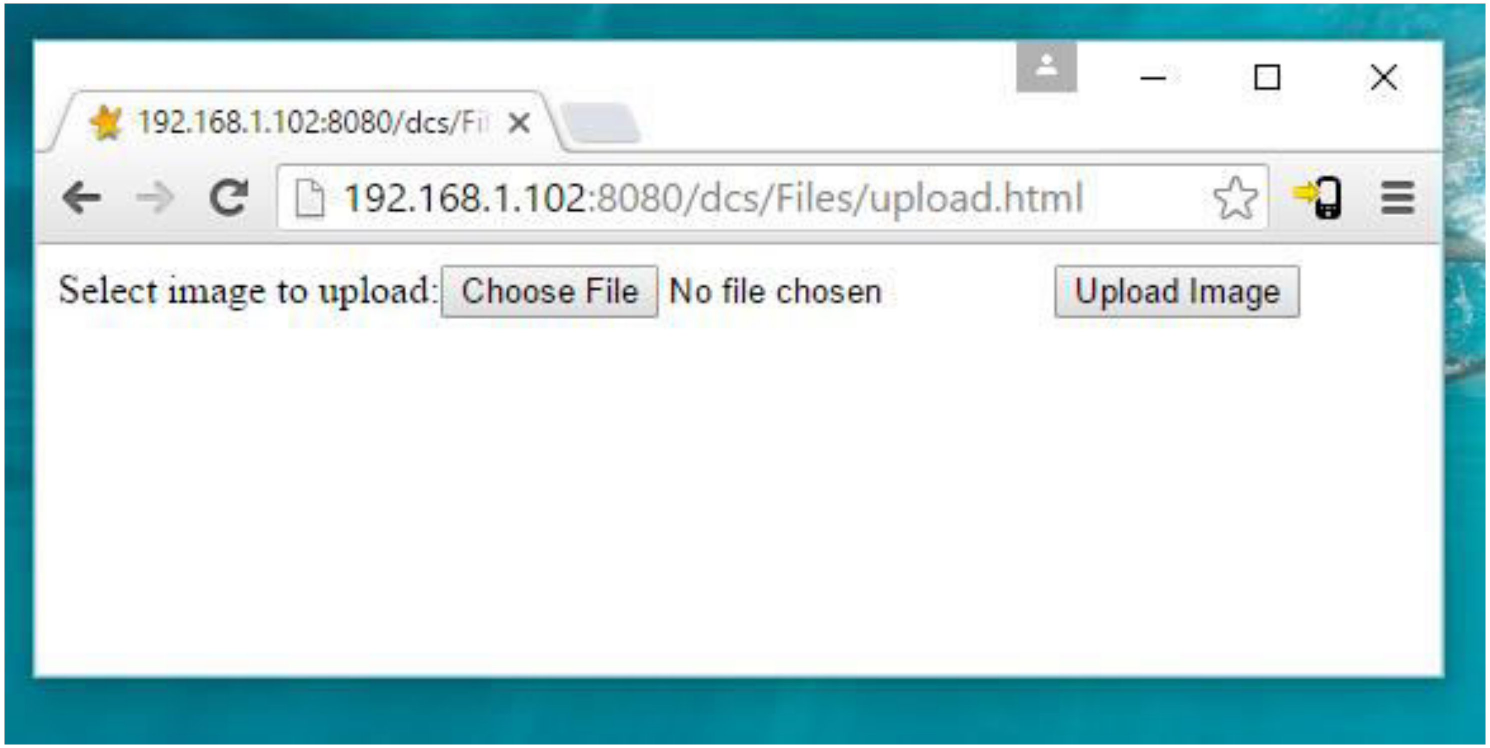
Without the requirements above, the file upload will not work.

Other things to notice:

- The type="file" attribute of the <input> tag shows the input field as a file-select control, with a "Browse" button next to the input control

- The form above sends data to a file called "upload.php", which we will create next.

Web Browser Output


192.168.1.102:8080/dcs/Fi
192.168.1.102:8080/dcs/Files/upload.html
Select image to upload:
Choose File
No file chosen
Upload Image

Create The Upload File PHP Script

The "upload.php" file contains the code for uploading a file:

```
<?php
$target_dir = "uploads/";
$target_file = $target_dir . basename($_FILES["fileToUpload"]["name"]);
$uploadOk = 1;
$imageFileType = pathinfo($target_file,PATHINFO_EXTENSION);
// Check if image file is a actual image or fake image
if(isset($_POST["submit"])) {
    $check = getimagesize($_FILES["fileToUpload"]["tmp_name"]);
    if($check !== false) {
        echo "File is an image - " . $check["mime"] . ".";
        $uploadOk = 1;
    } else {
        echo "File is not an image.";
        $uploadOk = 0;
    }
}?>
```

Upload File PHP Script explained

- $target_dir = "uploads/" - specifies the directory where the file is going to be placed
- $target_file specifies the path of the file to be uploaded
- $uploadOk=1 is not used yet (will be used later)
- $imageFileType holds the file extension of the file
- Next, check if the image file is an actual image or a fake image

Note: You will need to create a new directory called "uploads" in the directory where "upload.php" file resides. The uploaded files will be saved there.

Check if File Already Exists

- Now we can add some restrictions.
- First, we will check if the file already exists in the "uploads" folder. If it does, an error message is displayed, and $uploadOk is set to 0:

```
// Check if file already exists
if (file_exists($target_file)) {
    echo "Sorry, file already exists.";
    $uploadOk = 0;
 }
```

Limit File Size

- The file input field in our HTML form above is named "fileToUpload".
- Now, we want to check the size of the file. If the file is larger than 500kb, an error message is displayed, and $uploadOk is set to 0:

```
 // Check file size
if ($_FILES["fileToUpload"]["size"] > 500000) {
    echo "Sorry, your file is too large.";
    $uploadOk = 0;
 }
```

Limit File Type

- The code below only allows users to upload JPG, JPEG, PNG, and GIF files. All other file types gives an error message before setting $uploadOk to 0:

```
// Allow certain file formats
if($imageFileType != "jpg" && $imageFileType !=
"png" && $imageFileType != "jpeg"
&& $imageFileType != "gif" ) {
    echo "Sorry, only JPG, JPEG, PNG & GIF files are
allowed.";
    $uploadOk = 0;
}
```

Finally Upload File

Now if $uploadOk is set to 0 then file will not be uploaded otherwise file uploading code will be executed

```
// Check if $uploadOk is set to 0 by an error
if ($uploadOk == 0) {
    echo "Sorry, your file was not uploaded.";
// if everything is ok, try to upload file
} else {
    if (move_uploaded_file($_FILES["fileToUpload"]["tmp_name"],
   $target_file)) {
        echo "The file ". basename( $_FILES["fileToUpload"]["name"]). "
   has been uploaded.";
    } else {
        echo "Sorry, there was an error uploading your file.";
    }
}
```

PHP Exception Handling

What is an Exception?

- **Exceptions** are events that occur during the execution of programs that disturb the normal flow of instructions.
- We can also explicitly trigger an exception with throw keyword
- After exception is triggered it must be handled
- This is what normally happens when an exception is triggered:
  - The current code state is saved
  - The code execution will switch to a predefined (custom) exception handler function
  - Depending on the situation, the handler may then resume the execution from the saved code state, terminate the script execution or continue the script from a different location in the code

Note: Exceptions should only be used with error conditions, and should not be used to jump to another place in the code at a specified point.

Basic Use of Exceptions

PHP Exception Handling

Basic Use of Exceptions

- When an exception is thrown, the code following it will not be executed, and PHP will try to find the matching "catch" block.
- If an exception is not caught, a fatal error will be issued with an "Uncaught Exception" message.

Lets try to throw an exception without catching it:

```
<?php
 //create function with an exception
 function checkNum($number) {
   if($number>1) {
     throw new Exception("Value must be 1 or below");
   }
   return true;
 }
 //trigger exception
 checkNum(2);
 ?>
```

The code above will get an error like this:

```
Fatal error: Uncaught exception 'Exception'
with message 'Value must be 1 or below' in C:\webfolder\test.php:6
Stack trace: #0 C:\webfolder\test.php(12):
checkNum(28) #1 {main} thrown in C:\webfolder\test.php on line 6
```

Try, throw and catch

- To avoid the error from the example above, we need to create the proper code to handle an exception.

- Proper exception code should include:
  1. Try - A function using an exception should be in a "try" block. If the exception does not trigger, the code will continue as normal. However if the exception triggers, an exception is "thrown"
  2. Throw - This is how you trigger an exception. Each "throw" must have at least one "catch"
  3. Catch - A "catch" block retrieves an exception and creates an object containing the exception information

Try, throw and catch example

Lets try to trigger an exception with valid code:

```
<?php
 //create function with an exception
 function checkNum($number) {
   if($number>1) {
     throw new Exception("Value must be 1 or below");   }
   return true;
 }
//trigger exception in a "try" block
 try {
   checkNum(2);
   //If the exception is thrown, this text will not be shown
   echo 'If you see this, the number is 1 or below'; }

//catch exception
 catch(Exception $e) {
   echo 'Message: ' .$e->getMessage();
 }
 ?>
```

The code above will get an error like this:

```
Message: Value must be 1 or below
```

Example explained:

- The code throws an exception and catches it:
  1. The checkNum() function is created. It checks if a number is greater than 1. If it is, an exception is thrown
  2. The checkNum() function is called in a "try" block
  3. The exception within the checkNum() function is thrown
  4. The "catch" block retrieves the exception and creates an object ($e) containing the exception information
  5. The error message from the exception is echoed by calling $e->getMessage() from the exception object

WORDPRESS

WHAT IS WORDPRESS

WordPress is web software you can use to create a beautiful website, blog, or app. We like to say that WordPress is both free and priceless at the same time.

WORDPRESS EXAMPLES

- https://wordpress.org/showcase/

County Meath to Host First Facebook Data Center in Ireland
Recent News and Announcements
Contact Us
Upcoming Events

BENEFITS OF WORDPRESS

- Free
- Easy to use
- Written in PHP
- MySQL Database at back end
- Easy customization (Themes)
- Simple to understand

THINGS TO KNOW BEFORE INSTALLING WORDPRESS

- Access to your web server
- A text editor
- An FTP Client
- Your web browser of choice

REQUIREMENTS

- You must be able to execute PHP at your web server
- Access to an MySql Database
- In simple you web hosting company should provide PHP and MySQL support

INSTALLING WORDPRESS ON LOCALHOST

- Access to your web server www directory
- A text editor
- Your web browser of choice
- MySQL Database

INSTALLING ON REMOTE HOST

- Access to your web server www or public_html directory
- A text editor
- An FTP Client
- Your web browser of choice
- A MySQL Database

INSTALLATION PROCESS

ADMINISTRATOR LOGIN

WORDPRESS DASHBOARD

CUSTOMIZATION OF WEBSITE INFORMATION

ADDING PAGES TO YOUR WEBSITE

- Find the **Pages** menu in the WordPress Dashboard Navigation menu. Click **Add new.**

- Add the **title** of the page, like *About*. Note: If you have pretty permalinks set up, the title of your page will also be the URL slug.

- Next, add some **content**.

- To Publish the page click on save and publish

USING MEDIA IN WORDPRESS

- Media consists of the images, video, recordings, and files that you upload and use in your site/blog.
- Media is typically uploaded and inserted into the content when writing a Post or writing a Page.

MEDIA LIBRARY

The **Media Library Screen** allows you to edit, view, and delete Media previously uploaded to your blog. Multiple Media objects can be selected for deletion.

Search and filtering ability is also provided to allow you to find the desired Media.

MENUS IN WORDPRESS

DEFINING A MENU

You must define a menu before you can add items to it.

- Login to the Wordpress Dashboard.
- From the 'Appearance' menu on the left-hand side of the Dashboard, select the 'Menus' option to bring up the Menu Editor.
- Select **Create a new menu** at the top of the page
- Enter a name for your new menu in the Menu Name box
- Click the **Create Menu** button.

ADDING ITEMS TO A MENU

You can add different link types into your menu, these are split between panes left of the menu you're currently editing.

- Locate the pane entitled **Pages**.
- Within this pane, select the *View All* link to bring up a list of all the currently published Pages on your site.
- Select the Pages that you want to add by clicking the checkbox next to each Page's title.
- Click the **Add to Menu** button located at the bottom of this pane to add your selection(s) to the menu that you created in the previous step.
- Click the **Save Menu** button once you've added all the menu items you want.

DELETING A MENU ITEM

- Locate the menu item that you want to remove in the menu editor window
- Click on the arrow icon in the top right-hand corner of the menu item/box to expand it.
- Click on the *Remove* link. The menu item/box will be immediately removed.
- Click the **Save Menu** button to save your changes.

COMPLETE GUIDE

https://codex.wordpress.org/WordPress_Menu_User_Guide

WORDPRESS THEMES

WordPress themes are simply a group of files, called templates, which determine the look and basic function of your site

FREE VS PAID THEMES

FREE THEMES

https://wordpress.org/themes/

PAID THEMES

- http://www.templatemonster.com/wordpress-themes.php
- http://themeforest.net/
- http://www.themezilla.com/themes/

WORDPRESS PULGINS

Plugins tiny pieces of software which are used to extend and add to the functionality that already exists in **WordPress**.

The core of **WordPress** is designed to be lean and lightweight, to maximize flexibility and minimize code bloat. **Plugins** then offer custom functions and features so that each user can tailor their site to their specific needs.

https://wordpress.org/plugins/

EXAMPLE

We need to have employee list for our website which we have developed in Wordpress.

We can use plugins like

- Employee Spotlight
- Simple Staff List
- OS Our Team

WORDPRESS THEME CUSTOMIZATION

THEME CUSTOMIZER

The Theme Customizer allows you to preview changes to your site before publishing them. You can also navigate to different pages on your site to preview them.

- Site Title & Tagline
- Colors
- Header Image
- Background Image
- Navigation
- Widgets
- Static Front Page

FURTHER READING

https://codex.wordpress.org/Appearance_Customize_Screen