

PHP (PHP:HYPertext Preprocessor)

PHP introduction

- **PHP is a server scripting language, and a powerful tool for making dynamic and interactive Web pages.**
 - PHP is a widely-used, open source scripting language
 - PHP scripts are executed on the server and the result is returned to the browser as plain HTML
 - PHP is free to download and use.
 - PHP files can contain text, HTML, CSS, JavaScript, and PHP code ; have extension ".php"

Installing PHP

- We need:
 - PHP interpreter/parser
 - o In order to process PHP script instructions a parser must be installed to generate HTML output that can be sent to the Web Browser.
 - Apache Web Server (Apache HPPTD)
 - MySQL for storing data
- You can download each of these and separately install
- Better option? XAMPP
 - An all-in-one solution (www.apachefriends.org/en/xampp.html),
 - It rolls Apache, MySQL, PHP, and a few other useful tools together into one easy installer.
 - XAMPP is free and available for Windows, Mac, and Linux.

PHP introduction

- **Installing PHP**
 - We need:
 - PHP interpreter/parser
 - In order to process PHP script instructions a parser must be installed to generate HTML output that can be sent to the Web Browser.
 - Apache Web Server (Apache HPPTD)
 - MySQL for storing data
 - You can download each of these and separately install
 - Better option? XAMPP
 - An all-in-one solution (www.apachefriends.org/en/xampp.html),
 - It rolls Apache, MySQL, PHP, and a few other useful tools together into one easy installer.
 - XAMPP is free and available for Windows, Mac, and Linux.

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  
?>
```

```
<!DOCTYPE html>  
<html>  
<body>  
<h1>My first PHP page</h1>  
<?php  
    echo "Hello World!";  
?>  
</body>  
</html>
```

Basic PHP Syntax

- A PHP script can be placed anywhere in the document.
- A PHP script starts with <?php and ends with ?>:

```
<?php  
echo "<h1>Hello from PHP</h1>";  
?>
```

A PHP file normally contains HTML tags, and some PHP scripting code.

- **Comments in PHP :**
 - PHP supports several ways

```
// a single-line comment  
# also a single-line comment  
/*  
a multiple-lines comment block  
spanning multiple lines  
*/
```

PHP Variables

- **A variable starts with the \$ sign, followed by the name of the variable**
 - PHP has no command for declaring a variable. It is created the moment you first assign a value to it. Eg:

```
$txt = "Hello world!";
$x = 5;
$y = 10.5;
echo "I love $txt!";
echo "I love " . $txt . " | "
```

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

PHP Variables

- **A variable starts with the \$ sign, followed by the name of the variable**
 - PHP has no command for declaring a variable. It is created the moment you first assign a value to it. Eg:

```
$txt = "Hello world!";
$x = 5;
$y = 10.5;
```

----Concatenation operator .
echo "I love \$txt";
echo "I love " . \$txt . "!";

PHP Variables

- **Rules for PHP variables:**

- Starts with the \$ sign, followed by the name of the variable
- Must start with a letter or the underscore character only
- Can only contain alpha-numeric characters & underscores (A-z, 0-9, _)
- Are case-sensitive (\$age and \$AGE are different)

Example,

Valid variables:

\$popeye, \$one and \$INCOME

Invalid variables:

\$123 and \$48hrs are invalid.

PHP Variables

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

- **+ is pure math operator:**

```
$x = "15" + 27;  
echo ($x); //gives 42!!
```

Variables Scope

- **Variables can be declared anywhere in the script.**
- **PHP has three different variable scopes:**
 - Local : can only be accessed within that function
 - Global : can only be accessed outside a function
 - Static : retains value of local variable even after function ends

```
<?php
    $x = 5; // global scope
    function myTest() {
        echo "<p>Variable x inside function is: $x</p>"; // x inside function gives error
        $y = 10; // local scope
        echo "<p>Variable y inside function is: $y</p>";
    }
    myTest();
    echo "<p>Variable x outside function is: $x</p>"; // y outside function gives error
?>
```

Variables Scope

- The global Keyword

```
<?php  
$x = 5; $y = 10;  
  
function myTest() {  
    global $x, $y;  
    $y = $x + $y;  
}  
  
myTest();  
echo $y; // outputs 15  
?>
```

```
<?php  
$x = 5; $y = 10;  
  
function myTest() {  
    $GLOBALS['y'] = $GLOBALS['x'] + $GLOBALS['y'];  
}  
  
myTest();  
echo $y; // outputs 15  
?>
```

Variables Scope

- **The static Keyword**

- If we want a local variable NOT to be deleted when a function is completed/executed

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

Note: The variable is still local to the function.

Outputs : 0 1 2

Output data to the screen

- There are two basic ways to get output: echo and print

```
echo "<h2>PHP </h2>";  
echo "Hello <b>$name</b>!<br>";  
echo "This ", "string ", "was ", "made ", "with multiple  
parameters.:";
```

```
print "<h2>PHP</h2>";  
print "Hello world!<br>";  
print "I'm enjoying PHP!";
```

Output data to the screen

- **print_r() and var_dump()** : dumps out PHP data - it is used mostly for debugging

```
$stuff = array("name" => "Chuck", "course" => "SI664");
print_r($stuff);
```

```
$handle = fopen("note.txt", "r");
var_dump($handle);
```

Data Types

- **PHP supports the following data types:**
 - String Eg - \$x = "Hello world!";
 - Integer Eg - \$x = 1234;
 - Float (floating point numbers - also called double) Eg \$x = 10.365;
 - Boolean Eg - \$x = true;
 - Array Eg - \$cars = array("Samsung","Apple","Google","Nokia");
 - Object
 - NULL
 - Resource : are special variables that hold references to resources external to PHP (such as database connections)

Data Types

- **PHP supports the following data types:**

- String Eg - \$x = "Hello world!";
- Integer Eg - \$x = 1234;
- Float (floating point numbers - also called double) Eg \$x = 10.365;
- Boolean Eg - \$x = true;
- Array Eg - \$cars = array("Samsung", "Apple", "Google", "Nokia");
- Object
- NULL
- Resource : are special variables that hold references to resources external to PHP (such as database connections)

```
$handle = fopen("note.txt", "r");
var_dump($handle);
```

- **Example: `gettext()`: Get the type of a variable**

```
$s1 = "";
$v1 = null;
$bool = false;
$no1 = 1;
echo ('$s1 : ' . gettext($s1) . "<br>");
echo ('$v1 : ' . gettext($v1) . "<br>");
echo ('$bool : ' . gettext($bool) . "<br>");
echo ('$no1 : ' . gettext($no1) . "<br>");
```

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*).
- To create a constant, use the **define()** function.
 - Syntax : `define(name, value, case-insensitive)` //case: Default is false

```
<?php
define("GREETING", "Welcome to PHP!");
echo GREETING;
?>
```

```
// Valid constant names
define("const1", "first thing");
define("const2", "second thing");
define("const3", "third thing")
```

- Constants are automatically global & can be used across the entire script

- .

```
<?php
define("GREETING", "Welcome to PHP");
function myTest() {
    echo GREETING;
}
myTest();
?>
```

PHP Operators

- **Arithmetic Operators**

Operator	Example	Result
+	$4 + 2$	6
-	$4 - 2$	2
*	$4 * 2$	8
/	$4 / 2$	2
%	$4 \% 2$	0
++	$x = 4; x++;$	$x = 5$
--	$x = 4; x--;$	$x = 3$

- **Logical Operators**

Operator	Meaning
$\ $	or
$\&\&$	and
and	and
or	or
xor	xor
!	not

- **Assignment Operators**

Operator	Example	Meaning
$+=$	$y += x$	$y = y + x$
$-=$	$y -= x$	$y = y - x$
$*=$	$y *= x$	$y = y * x$
$/=$	$y /= x$	$y = y / x$
$%=$	$y %= x$	$y = y \% x$

PHP Operators

- **String Operators**

Operator	Example	Result
.	\$txt1 . \$txt2	Concatenation of \$txt1 & \$txt2
.=	\$txt1 .= \$txt2	Appends \$txt2 to \$txt1

- **Comparison Operators**

Operator	Meaning
==	is equal to
!=	is not equal to
>	is greater than
>=	is greater than or equal to
<	is less than
<=	is less than or equal to

Conditional Statements : syntax

```
if (condition) {  
    // if condition is true;  
} else {  
    //if condition is false;  
}
```

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

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

Conditional Statements : syntax

Conditional Statements : example

```
$sample = 10;  
IF ($sample > 5) {  
    print "Number is greater than 5";  
}  
ELSE {  
    print "Number is less than 5";  
}
```

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

```
<?php  
$favcolor = "red";  
  
switch ($favcolor) {  
    case "red":  
        echo "Your favorite color is red!";  
        break;  
    case "blue":  
        echo "Your favorite color is blue!";  
        break;  
    case "green":  
        echo "Your favorite color is green!";  
        break;  
    default:  
        echo "Your favorite color is neither  
red, blue, or green!";  
}  
?>
```

Loops : syntax

```
while (condition is true) {  
    code to be executed;  
}
```

```
do {  
    code to be executed;  
} while (condition is true);
```

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

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

```
foreach ($array_name as $key => $value) {  
    code to be executed;  
}
```

Note : foreach() works only on arrays, and is used to loop through each key/value pair in an array

User Defined Functions

- Syntax:

```
function "function_name" (arg1, arg2...){  
    [code to execute]  
    return [final_result];  
}
```

- Example:

```
function writeMsg() {  
    echo "Hello world!";  
}  
writeMsg(); // call the function
```

```
function writeMsg($fname) {  
    echo "Hello $fname!";  
}  
writeMsg(" PHP ");  
writeMsg(" Data ");
```

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

```
function sum($x, $y) {  
    $z = $x + $y;  
    return $z;  
}  
echo "5 + 10 = " . sum(5, 10);
```

Functions

- **Passing Parameters by Reference**

```
function square(&$value)      {  
    $value = $value * $value;  
}  
$a = 3;  
square($a);  
echo $a;
```

- **Default Parameters**

```
function getPreferences($whichPreference = 'all') {  
    //code that uses the parameter  
}
```

Functions

- **Variable Parameters:** PHP provides three functions you can use in the function to retrieve the parameters passed to it.
 - `func_get_args()` returns an array of all parameters provided to the function;
 - `func_num_args()` returns the number of parameters provided to the function;
 - `func_get_arg()` returns a specific argument from the parameters.

```
$array = func_get_args();
$count = func_num_args();
$value = func_get_arg(argument_number);
```

PHP include

- **INCLUDE appends the code from an external file into the current file.**

- Including files is very useful when you want to include the same PHP, HTML, or text on multiple pages of a website
- Syntax : INCLUDE ("external_file_name");

```
//index.php
<?php
    print "This is the original content<br>";
    include ("external.php");
?
?
```

```
//external.php
<?php
    print "This is the external content";
?>
```

-----Output-----

This is the original content
This is the external content

```
<?php include "header.html"; ?>
content
<?php include "footer.html"; ?>
```

Indexed Arrays

- array() function is used to create an array

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

or

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

```
$cars = array("Volvo", "BMW", "Toyota");
echo "I like " . $cars[0] . ", " . $cars[1] . " and " . $cars[2] . ".ias
```

Output : I like Volvo, BMW and Toyota.

```
$stuff = array();
$stuff[2] = "Hello";
$stuff[9] = "World";
echo $stuff[9]; //o/p : world
```

```
$arr = array(1, 2, 3, 4);
foreach ($arr as $value) {
    $value = $value * 2;
}
```

- count()** : returns length of array
 - Can use sizeof() too

```
$cars = array("Volvo", "BMW", "Toyota");
for($x = 0; $x < count($cars); $x++) {
    echo $cars[$x] . "<br>";
}
```

Associative Arrays

- Are arrays that use named keys that you assign to them

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

or

```
$age['Peter'] = "35";
$age['Ben'] = "37";
$age['Joe'] = "43";
echo "Peter is " . $age['Peter'] . " years old.";
```

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

```
//output:
Key=Peter, Value=35
Key=Ben, Value=37
Key=Joe, Value=43
```

Multidimensional array

- A multidimensional array is an array containing one or more arrays
 - Can be two, three, four, five, or more levels deep.

```
$cars = array(  
    array("Volvo",22,18),  
    array("BMW",15,13),  
    array("Saab",5,2),  
    array("Land Rover",17,15)  
);
```

Table for this
data

data

Name	Stock	Sold
Volvo	22	18
BMW	15	13
Saab	5	2
Land Rover	17	15

Volvo: In stock: 22, sold: 18.
BMW: In stock: 15, sold: 13.
Saab: In stock: 5, sold: 2.
Land Rover: In stock: 17, sold: 15.

output

Multidimensional array

```
echo $cars[0][0].": In stock: ".$cars[0][1].", sold: ".$cars[0][2].".<br>";  
echo $cars[1][0].": In stock: ".$cars[1][1].", sold: ".$cars[1][2].".<br>";  
echo $cars[2][0].": In stock: ".$cars[2][1].", sold: ".$cars[2][2].".<br>";  
echo $cars[3][0].": In stock: ".$cars[3][1].", sold: ".$cars[3][2].".<br>";
```

```
for ($row = 0; $row < 4; $row++) {  
    echo "<p><b>Row number $row</b></p>". "<ul>";  
    for ($col = 0; $col < 3; $col++)  
        echo "<li>".$cars[$row][$col]."</li>";  
    echo "</ul>"; }
```

Sorting Arrays

- **array sort functions and other functions:**

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

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

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

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

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

Sorting Arrays

- `array_keys()` : Return an array containing the keys
- `array_pop()` : deletes the last element of an array
- `array_push()` : inserts one or more elements to the end of an array
- `array_values()` : returns an array containing all the values of an array
- `array_slice()` : returns selected parts of an array.
`array_slice(array,start,length)`

String Functions

- `strlen()` : returns the length of a string
- `str_word_count()` : counts the number of words in a string
- `strrev()` : reverses a string
- `strchr()` : searches for the first occurrence of a string inside another string.
- `strpos()` : searches for a specific text within a string. `strpos(string,find)`
- `str_replace()` : replaces some characters with others in a string
- `strtolower()` and `strtoupper()`
- `explode()` : breaks a string into an array. `explode(separator,string)`
- `parse_str()` : parses a query string into variables
- `strcmp()` : compares two strings. `strcmp(string1,string2)`

String Functions

```
$str = "Hello world. It's a beautiful day.";  
echo strlen("Hello")."<br>";           //returns 5  
echo strpos($str,"world")."<br>";      //returns 6  
echo strrev("Hello World!")."<br>";    //returns !dlroW olleH  
echo strtoupper($str)."<br>";          //HELLO WORLD. IT'S A BEAUTIFUL DAY.  
print_r(explode(" ",$str));            //Break a string into an array  
//Array ( [0] => Hello [1] => world. [2] => It's [3] => a [4] => beautiful [5] => day. )  
echo "<br>";  
echo str_replace("world","everybody",$str)."<br>";  
//Hello everybody. It's a beautiful day.  
parse_str("name=Peter&age=43");  
echo $name."<br>";                      //Peter  
echo $age;                                //43
```

Date Function

- There is no way to directly get date/time in PHP
 - The PHP `date()` function formats a timestamp to a more readable date and time.
- `date(format [, timestamp])`
 - `format`: Required. Specifies the format of the timestamp
 - `timestamp` : Optional. Specifies a timestamp. Default is current date & time
- Some of the formatting characters commonly used:
 - `d` - The day of the month (from 01 to 31)
 - `D` - A textual representation of a day (three letters) (Fri, Sun etc)
 - `j` - The day of the month without leading zeros (1 to 31)
 - `N` - Numeric representation of a day (1 for Monday, 7 for Sunday)
 - `F` - Textual representation of a month (January through December)
 - `m` - A numeric representation of a month (from 01 to 12)
 - `M` - A short textual representation of a month (three letters)

```
Eg : echo date("m/d/y"); // output : 04/10/13  
echo date("l"); // Wednesday
```

Refer to <http://php.net/manual/en/function.date.php> for more

Date Function

- PHP core also provides a number of date and time; since they are built in, you can use these functions directly within your script.
 - `date_create()` : returns a new `DateTime` object.
 - `date_format()` : returns a date formatted according to the specified format.
 - `getdate()` : returns date/time info of a timestamp or current date/time.

```
$date=date_create("2013-03-15"); print_r($date);
//DateTime Object ( [date] => 2013-03-15 00:00:00.000000
[timezone_type] => 3 [timezone] =>
Europe/Berlin )
echo date_format($date,"Y/m/d H:i:s"); //2013/03/15 00:00:00
```

```
print_r(getDate()); //Returns an associative array of information
Array ( [seconds] => 31 [minutes] => 45 [hours] => 19 [mday] => 18 [wday] =>
3 [mon] => 2 [year] => 2015 [yday] => 48 [weekday] => Wednesday [month]
=> February [0] => 1424285131 )
```

Date Function

- PHP core also provides a number of date and time; since they are built in, you can use these functions directly within your script.
 - `date_create()` : returns a new `DateTime` object.
 - `date_format()` : returns a date formatted according to the specified format.
 - `getdate()` : returns date/time info of a timestamp or current date/time.

```
$date=date_create(  
"2013-03-15");  
print_r($date);  
//DateTime Object ( [date] => 2013-03-15 00:00:00.000000  
[timezone_type] => 3 [timezone] =>  
Europe/Berlin )  
echo date_format($date,"Y/m/d H:i:s"); //2013/03/15  
00:00:00
```

Global Variables - Superglobals

- Several predefined variables in PHP are "superglobals"
 - This means that they are always accessible, regardless of scope - and you can access them from any function, class or file without having to do anything special.
- They are:
 - `$GLOBALS` : used to access global variables from anywhere in the script
 - `$_SERVER` : holds information about headers, paths, and script locations
 - `$_REQUEST` : used to collect data after submitting an HTML form
 - `$_POST` : used to collect form data after submitting form with method="post"
 - `$_GET` : used to collect form data after submitting form with method="get"
 - `$_FILES` : An associative array of items uploaded to the current script via the HTTP POST method
 - `$_COOKIE` : represents data available to a PHP script via HTTP cookies.
 - `$_SESSION` : represents data available to a PHP script that has previously been stored in a session.

`$_SERVER` Superglobal

- Holds information about headers, paths, and script locations
- Some of the most important elements that can go inside `$_SERVER`:

Element/Code	Description
<code>\$_SERVER['PHP_SELF']</code>	Returns the filename of the currently executing script
<code>\$_SERVER['SERVER_NAME']</code>	Returns the name of the host server (such as www.mysite.com)
<code>\$_SERVER['REQUEST_METHOD']</code>	Returns the request method used to access the page (eg POST)
<code>\$_SERVER['QUERY_STRING']</code>	Returns the query string if the page is accessed via a query string. Everything after the ? in the URL (e.g., name=Fred+age=35).
<code>\$_SERVER['REMOTE_ADDR']</code>	Returns the IP address from where the user is viewing the current page
<code>\$_SERVER['REMOTE_HOST']</code>	Returns the Host name from where the user is viewing the current page
<code>\$_SERVER['SCRIPT_FILENAME']</code>	Returns the absolute pathname of the currently executing script
<code>\$_SERVER['PATH_TRANSLATED']</code>	Returns the file system based path to the current script
<code>\$_SERVER['SCRIPT_NAME']</code>	Returns the path of the current script
<code>\$_SERVER['SCRIPT_URI']</code>	Returns the URI of the current page

\$_SERVER Demo

```
<?php  
echo "Filename of executing script : " .  
$_SERVER['PHP_SELF']; echo "Name of the host  
server : " . $_SERVER['SERVER_NAME'];  
echo "Host header from the current request : " .  
$_SERVER['HTTP_HOST']; echo "User agent : " .  
$_SERVER['HTTP_USER_AGENT'];  
echo "Path of the current script : " . $_SERVER['SCRIPT_NAME'];  
?>
```

Filename of executing script : /phpdemo/test9_SERVER.php

Name of the host server : localhost

Host header from the current request : localhost

User agent : Mozilla/5.0 (Windows NT 10.0; WOW64; rv:52.0) Gecko/20100101 Firefox/52.0

Path of the current script : /phpdemo/test9_SERVER.php

\$_POST and **\$_GET**

- **\$_POST** : used to collect form data after submitting an HTML form with method="post".

```
<?php  
    $name = $_POST['fname']; // collect value of input field  
    if (empty($name)) echo "Name is empty";  
    else echo $name;  
?>
```

- **\$_GET**: used to collect form data after submitting an HTML form with method="get“

Mainpage.php

```
<body>  
<a href="test_get.php?fname=Anil&lname=Joshi">Test  
$GET</a>  
</body>
```

test_get.php

```
<body>  
<?php echo "Study " . $_GET['fname'] . " at " . $_GET['lname'];?>  
</body>
```

Form Handling (\$_POST)

```
//form.html
```

```
<html>
<body>
<form action="welcome.php" method="post">
Name: <input type="text" name="name"><br>
E-mail: <input type="text" name="email"><br>
<input type="submit">
</form>
</body>
</html>
```

```
//welcome.php
```

```
<html>
<body>
Welcome <?php echo $_POST["name"]; ?><br>
Your email address is: <?php echo $_POST["email"]; ?>
</body>
</html>
```

\$_REQUEST

- Used to collect data after submitting an HTML form

```
<body>
<form method="get" action=<?php echo $_SERVER['PHP_SELF'];?>>
    Name: <input type="text" name="fname">
    <input type="submit">
</form>

<?php
if ($_SERVER["REQUEST_METHOD"] == "GET") {
    $name = $_REQUEST['fname']; // collect value of input field
    if (empty($name)){
        echo "Name is empty";
    } else {
        echo "Welcome to PHP " . $name;
    }
}
?>
</body>
```

Form Validation

- **Isset()** : Determine if a variable is set and is not NULL
 - The isset() function returns TRUE, if the variable inside parentheses is set

```
<?php
if(isset($_POST["selRating"])) {
    $number = $_POST["selRating"];
    if((is_numeric($number)) && ($number > 0) && ($number < 6))
        echo "Selected rating: " . $number;
    else
        echo "The rating has to be a number between 1 and 5!";
}
?>
<form action=<?php echo $_SERVER['PHP_SELF']; ?>" method="post">
    Select a rating from 1 to 5:
    <select name="selRating">
        <option value="1">1</option>
        <option value="2">2</option>
        <option value="3">3</option>
        <option value="4">4</option>
        <option value="5">5</option>
    </select>
    <input type="submit" name="btnSendForm" value="Send" />
</form>
```

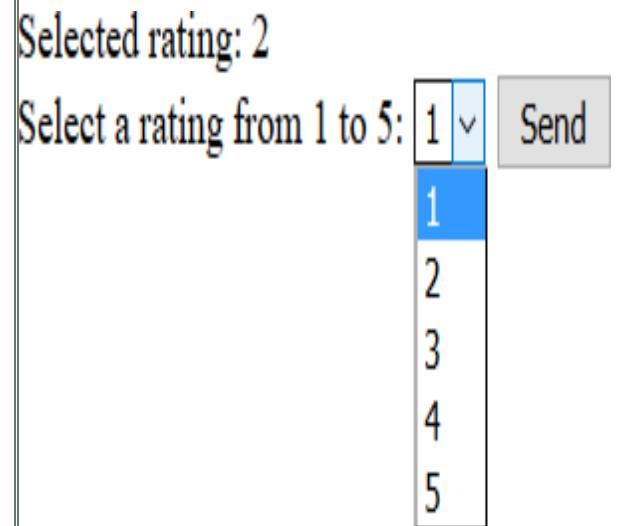
```
$s1 = isset($name); //
$s1 is false
$name = "Fred";
$s2 = isset($name); //
$s2 is true
```

Selected rating: 2

Select a rating from 1 to 5:

1 2 3 4 5

Send



Advanced form handling

- **Multivalued Parameters in select element:**
- **HTML's select tag allows multiple selections**
 - To process such multiple selection, you need to make the name of the field in the HTML form end with []. Eg:

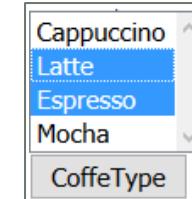
```
<select name="languages[]>
  <option name="c">C</input>
  <option name="c++">C++</input>
  <option name="php">PHP</input>
  <option name="perl">Perl</input>
</select>
```

- Now, when the user submits the form, `$_GET['languages']` contains an array instead of a simple string.
- This array contains the values that were selected by the user.

```
/selectElement.php?coffee[]=latte&coffee[]=espresso&submitbtn=CoffeType
```

```
<form action=<?php echo $_SERVER['PHP_SELF']; ?>" method="GET">
<select name="coffee[]" multiple>
    <option value="cappuccino">Cappuccino</option>
    <option value="latte">Latte</option>
    <option value="espresso">Espresso</option>
    <option value="mocha">Mocha</option>
</select><br />
<input type="submit" name="submitbtn" value="CoffeType" />
</form>
```

```
<?php if (array_key_exists('submitbtn', $_GET)) {
/* array_key_exists() : checks an array for a specified key & returns true if the key
exists else false. $_GET is an array and so we check if submit button has been
clicked $_GET['coffee'] contains an array which contains the values that were
selected by the user */
$description = join('', $_GET['coffee']); //join() converts array into string
echo "You have choosen <b><font color='blue'>$description</font></b> coffee." ;
}
?>
```



You have choosen **latte espresso** coffee.

Advanced form handling

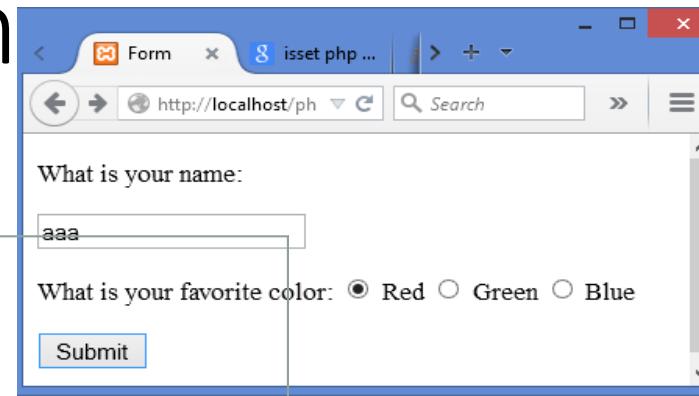
- Multivalued Parameters in checkbox element

```
<html>
<body>
<form action=<?php $_SERVER['PHP_SELF']; ?>" method="GET">
Select your Language:<br />
<input type="checkbox" name="language[]" value="cpp" /> C++<br />
<input type="checkbox" name="language[]" value="java" /> Java<br />
<input type="checkbox" name="language[]" value="csharp" /> C#<br />
<input type="checkbox" name="language[]" value="cobol" /> Cobol<br />
<input type="submit" name="submitBtn" value="Choose Language!" />
</form>
<?php if (array_key_exists('s', $_GET)) {
    $description = join (' ', $_GET['language']);
    echo "You have chosen {$description} languages.";
} ?>
</body>
</html>
```

Advanced form handling

- Working with radio buttons

```
<body>
  <form method="post" action="handler.php">
    <p>What is your name:</p>
    <input type="text" name="username"></p>
    <p>What is your favorite color:</p>
    <input type="radio" name="favoritecolor" value="r" /> Red
    <input type="radio" name="favoritecolor" value="g" /> Green
    <input type="radio" name="favoritecolor" value="b" /> Blue </p>
    <input type="submit" value="Submit" />
  </form>
</body>
```



```
switch ($_POST["favoritecolor"]) {
  case "r": $strBackgroundColor = "rgb(255,0,0)"; break;
  case "g": $strBackgroundColor = "rgb(0,255,0)"; break;
  case "b": $strBackgroundColor = "rgb(0,0,255)"; break;
  default: $strBackgroundColor = "rgb(255,255,255)"; break;
}
```

PHP Session

- A session is a way to store information (in variables) to be used across multiple pages.
 - `session_start()` : starts a session
 - The `session_start()` function first checks for an existing session ID.
 - If it finds one, i.e. if the session is already started, it sets up the session variables and if doesn't, it starts a new session by creating a new session ID.
- Session variables are set as key-value pairs with the global variable: `$_SESSION`
 - The stored data can be accessed during lifetime of a session

```
$_SESSION["firstname"] = "Peter";  
$_SESSION["lastname"] = "Parker";
```

- `session_unset()` : to remove session data, simply unset the corresponding key of the `$_SESSION`
- `session_destroy()` : destroy the session

Session Demo

```
<?php  
session_start(); // Start the session  
?  
<body>  
<?php  
// Set session variables  
$_SESSION["favcolor"] = "green";  
$_SESSION["favanimal"] = "cat";  
echo "Session variables are set.";  
?  
<a href="SessionExample2.php">Follow </a> link to go to SessionExample2.php  
</body>
```

SessionExample1.php

Session variables are set.

[Follow](#) link to go to SessionExample2.php

```
<?php  
session_start();  
?  
<?php  
echo "Favorite color is " . $_SESSION["favcolor"] . ".<br>";  
echo "Favorite animal is " . $_SESSION["favanimal"] . ".";  
?>
```

SessionExample2.php

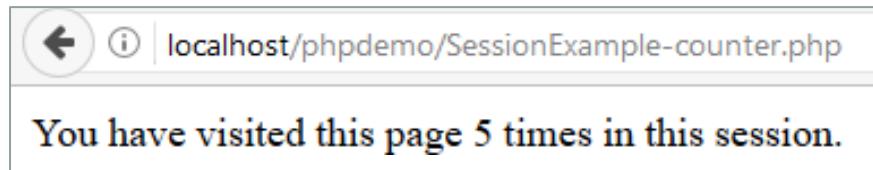
Favorite color is green.
Favorite animal is cat.

Session Demo

```
<?php
session_start();

if( isset( $_SESSION['counter'] ) ){
    $_SESSION['counter'] += 1;
} else {
    $_SESSION['counter'] = 1;
}
$msg = "You have visited this page ". $_SESSION['counter'];
$msg .= " times in this session.";
?>
<html>
<body>
    <?php echo ( $msg ); ?>
</body>

</html>
```



E-mail With PHP

- Server side scripting language must provide a way of sending e-mail from the server and, in particular, to take form input and output it to an e-mail address
- **mail(\$to,\$subject,\$body[\$headers]);** - for sending mail

```
$to = "abc@gmail.com";  
$subject = "Hi There!";  
$body = "PHP is one of the best scripting languages around";  
$headers = "From:  
xyz@gmail.com\n";  
mail($to,$subject,$body,$he  
ders);
```

Cookies

- **setcookie(cookie-name,cookie-value): instructs the browser to save a cookie**
 - `setcookie(name, value, expire, path, domain, secure);`
 - `setcookie()` goes via response header, which means that it has to be called before any output is made to the browser (including text, HTML etc)
 - The value you set can't be read until next time the page is loaded, ie you can't save a cookie and read the value in the same page execution
 - Eg : `setcookie("user_name", "John Doe"); //cookie expires when session ends`
 - Eg : `setcookie("age", "36", time() + 3600); //cookie expires after 1 hour`
- **The value can be retrieved again by using the `$_COOKIE` superglobal**
 - Eg : `echo $_COOKIE["user_name"];`
- **To delete a cookie, use the `setcookie()` function with an expiration date in the past**
 - Eg : `setcookie("user", "", time() - 3600); //set expiration date to one hour ago`

Cookie Demo

```
<?php  
setcookie("name", "John Watkin", time() + 3600, "/", "", 0); //expires after one hour  
setcookie("age", "36", time() + 3600, "/", "", 0);  
?>  
<body>  
<?php echo "Set Cookies"?>  
</body>
```

cookieSetExample.php

```
<body>  
<?php  
echo "Getting cookie info.....<hr><br>";  
if(!isset($_COOKIE["name"]))  
    echo "No cookie set!";  
else {  
    echo "Cookie is set!<br>";  
    echo "Name is : " . $_COOKIE["name"] . "<br />";  
    echo "Age is : " . $_COOKIE["age"] . "<br />";  
}  
?>  
</body>
```

cookieGetExample.php

Getting cookie info.....

Cookie is set!
Name is : John Watkin
Age is : 36

File handling (read)

- **File handling is an important part of any web application. You often need to open and process a file for different tasks.**
 - PHP has several functions for creating, reading, uploading, and editing files
 - **readfile(filename):** reads a file and store data to the output buffer
 - is useful if all you want to do is open up a file and read its contents
 - **fopen(filename,mode) :** similar to readfile(), but gives more options
 - If an attempt to open a file fails then fopen returns a value of false otherwise it returns a file pointer which is used for further reading or writing to that file.

Mode	What it does
r	Opens the file for reading only.
r+	Opens the file for reading and writing.
w	Opens the file for writing only and clears the contents of file. If file does not exist then it attempts to create a file.
w+	Opens the file for reading and writing and clears the contents of file. If file does not exist then it attempts to create a file.
a	Append. Opens the file for writing only. Preserves file content by writing to the end of the file. If file does not exist then it attempts to create a file.
a+	Read/Append. Opens the file for reading and writing. Preserves file content by writing to the end of the file. If file does not exist then it attempts to create a file.

File handling (read)

- **filesize(filename)** : returns file length in bytes
- **fread(filepointer, length of file)** : read a file that is opened with fopen()
- **fclose(filepointer)** : close the file

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

```
<?php
$filename = "sample.txt";
$fileptr = fopen( $filename, "r" );
if( $fileptr == false ) {
    echo ( "Error in opening file" );
    exit();
}
$filesize = filesize( $filename );
$filetext = fread( $fileptr, $filesize );
fclose( $fileptr );
echo ( "File size : $filesize bytes" );
echo ( "<pre>$filetext</pre>" );
?>
```



File size : 56 bytes
twinkle twinkle
Little star
How i wonder
What you are

File handling (read)

- **file_get_contents()** : reads file contents into a string

```
$dataStr =  
file_get_contents($filename);  
echo $dataStr;
```

- **file()** - Reads entire file into an array with each line of the file corresponding to an element of the array

```
$filename = "sample.txt";  
$dataArr = file($filename);  
print_r($dataArr);
```

File handling (read)

- **fgets(filename)** : used to read a single line from a file

```
<?php  
$myfile = fopen("webdictionary.txt", "r") or die("Unable to open file!");  
//die() : prints a message and exits the current script  
echo fgets($myfile); //reads first line only  
fclose($myfile);  
?>
```

```
while(!feof($myfile)){  
    echo fgets($myfile) . "<br>";  
}
```

File handling

- **file_exists(filename) : checks existence of file**
- **feof() : checks if the "end-of-file" (EOF) has been reached**

```
$file_open = '/home/httpd/index.htm';
if (file_exists($file_open))
    echo "The file $file_open exists";
else
    echo "The file $file_open does not exist"
```

- **fwrite(filepointer, text to be written,[data length]) : A new file can be written or text can be appended to an existing file**

```
$filename = "newfile.txt";
$file = fopen( $filename, "w" );
fwrite( $file, "This is a simple test\n" );
fclose( $file );
if( file_exists( $filename ) ) {
    $filesize = filesize( $filename );
    echo "File created with name $filename containing $filesize bytes ";
}
else echo ("File $filename does not exist" );
```

File handling

- **copy()** : creates a copy of a file

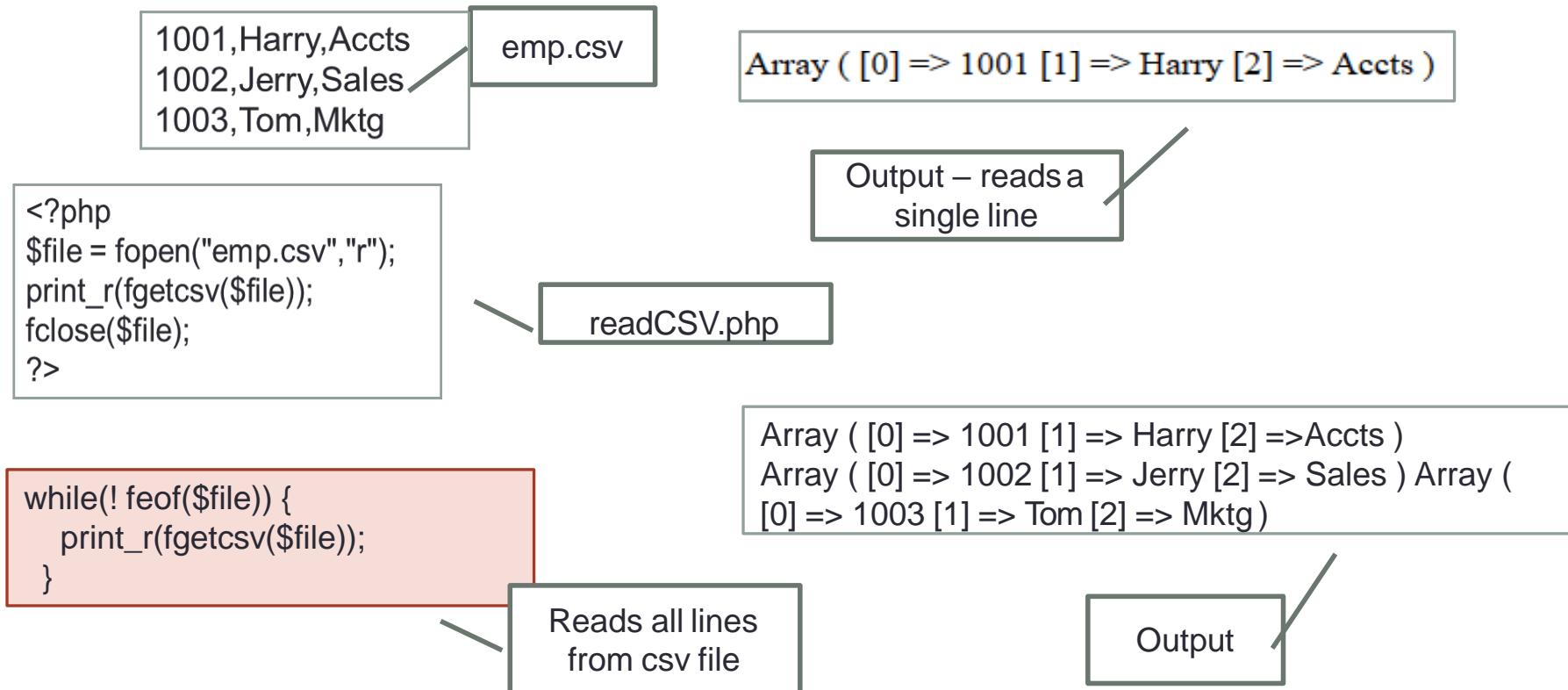
```
$source = "dummy.txt";
$destination = "dummy.txt.backup";
// copy file if it exists, else exit
if (file_exists($source)) {
    copy ($source, $destination) or die ("Cannot copy file '$source'");
    echo "File successfully copied.";
} else die ("Cannot find file '$source'");
```

- **unlink()** : delete a file

```
$file= "dummy.txt";
if (file_exists($file)) {
    unlink ($file) or die("Cannot delete file '$file'");
    echo "File successfully deleted.";
} else die ("Cannot find file '$file'");
```

File handling : CSV (Comma Separated Values) files

- `fgetcsv()` : allows you to work with CSV file; similar to `fgets()`
 - `fgetcsv()` separates each line on the commas & puts each part into an array.
 - Syntax : `fgetcsv(file,length,separator)`
 - Length : Optional. Specifies the maximum length of a line. Must be greater than the longest line (in characters) in the CSV file. Required in versions prior to PHP 5
 - Separator : A character that specifies the field separator. Default is comma (,)



File handling : CSV (Comma Separated Values) files

- update or write to a CSV file using the function **fputcsv()**

```
$file = "employees.csv";
$f = fopen($file, "a");

$newFields = array(
    array('Tom', 'Jones', 36, 'Accountant'),
    array('Freda', 'Williams', 45, 'Analyst'),
    array('Brenda', 'Collins', 34, 'Engineer'));

foreach($newFields as $fields) {
    fputcsv($f, $fields);
}
fclose($f);
```

Using the die() function

```
<?php  
$file=fopen("welcome.txt","r");  
?> //if file does not exist? --->
```

```
Warning: fopen(welcome.txt) [function.fopen]:  
failed to open stream:  
No such file or directory in C:\webfolder\test.php  
on line 2
```

If the file does not exist you might get an error like above

- We can test whether the file exist before we try to access it. die() is a simple error handling mechanism to stop the script after the error.
- die(message) : prints a message and exits the current script.

```
<?php  
if(!file_exists("welcome.txt")) {  
    die("File not found");  
} else {  
    $file=fopen("welcome.txt","r");  
}  
?>
```

DB operations

- **Step-1: Open a Connection using MySQLi procedural**
 - `mysqli_connect(host,[username][,password][,dbname])`
 - Upon success it returns an identifier to the server; otherwise, FALSE

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

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

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

The connection will be closed automatically when the script ends. To close the connection before, use the `mysqli_close($conn);`

Steps to perform DB operations

- **Step 2 : Create a database**

- **mysqli_query(connection,query)** : Perform queries against the database

```
$servername = "localhost";
$username = "root";
$password = "";

$conn= mysqli_connect($servername, $username, $password) or die ("could
not connect to mysql" . mysqli_connect_error());

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

mysqli_close($conn);
```

Steps to perform DB operations

- **Step-3 : Create a Table**

```
$servername = "localhost";
$username = "root";
$password = "";
$dbname = "mySqlDB";
$conn= mysqli_connect($servername, $username, $password, $dbname) or
    die ("could not connect to mysql" . mysqli_connect_error());

$sql = "CREATE TABLE test (
id INT(6) UNSIGNED PRIMARY KEY,
name VARCHAR(15) NOT NULL )";

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

mysqli_close($conn);
```

Steps to perform DB operations

- Step-4a : Insert single record into table

```
<?php  
$servername = "localhost";  
$username = "root";  
$password = "";  
$dbname = "mySqlDB";  
$conn= mysqli_connect($servername, $username, $password, $dbname) or  
die ("could not connect to mysql" . mysqli_connect_error()); ?>
```

db4-getConn.php

```
include_once("db4-getConn.php");  
  
$sql = "INSERT INTO test (id, name) VALUES (10,'John')";  
  
if (mysqli_query($conn, $sql))  
    echo "New record created successfully";  
else  
    echo "Error: " . $sql . "<br>" . mysqli_error($conn);  
  
mysqli_close($conn);
```

db5-InsertRow.php

Steps to perform DB operations

- **Step-4b : Insert multiple records into table**
 - **mysqli_multi_query(connection,query)** : Perform multiple queries against the database

```
<?php
include_once("db4-getConn.php");

$sql = "INSERT INTO test (id, name) VALUES (20,'Geeta');";
$sql .= "INSERT INTO test (id, name) VALUES (30,'Meeta');";
$sql .= "INSERT INTO test (id, name) VALUES (40,'Vanita');"

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

mysqli_close($conn);
?>
```

Steps to perform DB operations

- **Step-5 : Fetching result**

- `mysqli_result=mysqli_query(connection,query);`
 - For successful SELECT queries it will return a mysqli_result object. For other successful queries it will return TRUE; FALSE otherwise
- `mysqli_num_rows(result)` : returns the number of rows in a result set
- `mysqli_fetch_assoc()` : fetches a result row as an associative array; NULL if there are no more rows in result-set

```
include_once("db4-getConn.php");
$sql = "SELECT id, name FROM test";
$result = mysqli_query($conn, $sql);

if (mysqli_num_rows($result) > 0) {
    while($row = mysqli_fetch_assoc($result)){
        echo "id: " . $row["id"]. " - Name: " . $row["name"]. "<br>";
    }
} else echo "0 results";

mysqli_close($conn);
```

id:	10 - Name: John
id:	20 - Name: Geeta
id:	30 - Name: Meeta
id:	40 - Name: Vanita

Steps to perform DB operations

- **Step-5 : Fetching result**
- **Other ways of fetching result:**
 - `mysqli_fetch_all(result,resulttype)` : fetches all result rows and returns the result-set as an associative array, a numeric array, or both.

```
$result = mysqli_query($conn, $sql);
$arr = mysqli_fetch_all($result,MYSQLI_NUM);
print_r($arr);
```

```
Array ( [0] => Array ( [0] => 10 [1] => John )
[1] => Array ( [0] => 20 [1] => Shrilata ) [2]
=> Array ( [0] => 30 [1] => Meeta ) )
```

```
$result = mysqli_query($conn, $sql);
$arr = mysqli_fetch_all($result, MYSQLI_ASSOC);
print_r($arr);
```

```
Array ( [0] => Array ( [id] => 10 [name] => John
) [1] => Array ( [id] => 20 [name] => Shrilata )
[2] => Array ( [id] => 30 [name] => Meeta ) )
```

- `mysqli_fetch_row()` : fetches one row from a result-set and returns it as an enumerated array

```
while ($row=mysqli_fetch_row($result)){
    print_r($row);echo "<br>";
}
```

```
Array ( [0] => 10 [1] => John )
Array ( [0] => 20 [1] => Shrilata )
Array ( [0] => 30 [1] => Meeta )
```

Steps to perform DB operations

- **Step-6 : Update/Delete rows**

```
include_once("db4-getConn.php");

$sql = "DELETE FROM test WHERE id=40"; // sql to delete a record

if (mysqli_query($conn, $sql))
    echo "Record deleted successfully";
else echo "Error deleting record: " . mysqli_error($conn);

$sql = "UPDATE test SET name='Shrilata' WHERE id=20";

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

mysqli_close($conn);
```

PHP & XML

Parsing XML With SimpleXML

- **PHP 5's new SimpleXML module simplifies parsing an XML document**
 - SimpleXML is a tree-based parser.
 - Turns an XML doc into an object that provides structured access to the XML.

```
$xmlstr = "<?xml version='1.0' encoding='UTF-8'?> <person>
<fname>Navin</fname><lname>Dutt</lname>
<email>navin.dutt@gmail.com</email><phone>99800675432</phone>
</person>";

$xmlobj=simplexml_load_string($xmlstr) or die("Error: Cannot create object");
print_r($xmlobj); //printing entire SimpleXmlObject

echo "First name : " . $xmlobj->fname . "<br>";
echo "Last name : " . $xmlobj->lname . "<br>";
echo "Email : " . $xmlobj->email . "<br>";
echo "Phone : " . $xmlobj->phone;
```

Loading XML file

```
$xmlObj=simplexml_load_file("student.xml") or die("Error: Cannot create object");
print_r($xmlObj); //printing entire SimpleXmlObject

echo $xmlObj->student[0]->firstname . "<br>";
echo $xmlObj->student[1]->firstname . "<br>";
echo $xmlObj->student[2]->firstname . "<br>

foreach($xmlObj->children() as $students) {
    echo $students->firstname . ", ";
    echo $students->lastname . ", ";
    echo $students->marks . "<br>";
}

//getting attribute values:
echo $xmlObj->student[0]['rollno'] . "<br>";
```

```
----- printing SimpleXmlObject -----
Dinkar
Vaneet
Jasvir
----- Get Node Values - Loop -----
Dinkar, Kad, 85
Vaneet, Gupta, 95
Jasvir, Singh, 90
----- getting attribute values -----
393
```

```
<?xml version="1.0"?>
<class>
    <student rollno="393">
        <firstname>Dinkar</firstname>
        <lastname>Kad</lastname>
        <nickname>Dinkar</nickname>
        <marks>85</marks>
    </student>
    <student rollno="493">
        <firstname>Vaneet</firstname>
        <lastname>Gupta</lastname>
        <nickname>Vinni</nickname>
        <marks>95</marks>
    </student>
    <student rollno="593">
        <firstname>Jasvir</firstname>
        <lastname>Singh</lastname>
        <nickname>Jazz</nickname>
        <marks>90</marks>
    </student>
</class>
```

Address.xml

```
<contact-info>
    <name>peter</name>
    <company>KLFS services</company>
    <phone>(020) 1230-4567</phone>
</contact-info>
```

Reading XML using the DOM library

- **Built-in DOM parser makes it possible to process XML documents in PHP.**
 - How to do? Initialize the XML parser, load the xml, and output it

```
<?php
$xmlDoc = new DOMDocument(); //create a DOMDocument-Object
$xmlDoc->load("Address.xml"); //loads the XML into it
print $xmlDoc->saveXML(); //saveXML() puts the internal XML doc into a string

echo "<br>" . "----- Get Node Values - Loop -----" . "<br>";

$x = $xmlDoc->documentElement;
foreach ($x->childNodes AS $item){
    print $item->nodeName . " = " . $item->nodeValue . "<br>";
}
?>
```

- Jpspan
- It is a framework to send AJAX request from PHP to some other web service.

BOOTSTRAP

Getting Bootstrap Ready

- **There are two ways to start using Bootstrap on your own web site.**
 - Download Bootstrap from the oficial website <http://getbootstrap.com/> and include it in your HTML file with little or no customization.
 - Include Bootstrap from a CDN
 - The <http://getbootstrap.com/getting-started/> page gives CDN links for CSS and js files
 - A Basic template can be created using the bootstrap files and jquery libraries

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <title>First Bootstrap demo</title>
    <link rel="stylesheet" type="text/css" href="../scripts/css/bootstrap.css">
</head>
<body>
    <H1>Hello World!!</H1>
    <script src="../scripts/js/jquery-1.11.0.min.js"></script>
    <script src="../scripts/js/bootstrap.js"></script>
</body>
</html>
```

Bootstrap Container

- **Container is used to wrap the site contents and contain its grid system**
 - Thus dealing with the responsive behaviors of your layout.
 - There are two container classes in Bootstrap:
 - Fixed Container : A fixed container is a (responsive) fixed width container.
 - Fluid Container : A fluid container spans the full width of the viewport.
 - Note: A container cannot be placed inside a container

```
<body>
  <div class="container">
    <h1>Container</h1>
    <p>container content</p>
  </div>
</body>
```

Bootstrap Grid System

- **Bootstrap grid system allows to properly house the website's content.**
 - Grid system divides the screen into columns—up to 12 in each row.
 - The column widths vary according to the size of screen they're displayed in.
 - Hence, Bootstrap's grid system is responsive, as the columns resize themselves dynamically when the size of browser window changes.
- 4 types of col classes for different size displays:
 - col-xs for extra small displays (screen width < 768px)
 - col-sm for smaller displays (screen width ≥ 768px)
 - col-md for medium displays (screen width ≥ 992px)
 - col-lg for larger displays (screen width ≥ 1200px)

col-xs-*

- Extra small devices
- Mobile devices

col-sm-*

- Small devices
- tablets

col-md-*

- medium devices
- laptops , desktops

col-lg-*

- extra large desktops devices

```
<div class="container">
  <div class="row">
    <div class="col-xs-12 col-sm-6 col1">
      <h4>Column 1</h4>
    </div>
    <div class="col-xs-12 col-sm-6 col2">
      <h4>Column 2</h4>
    </div>
  </div>
</div>
```

Page Components

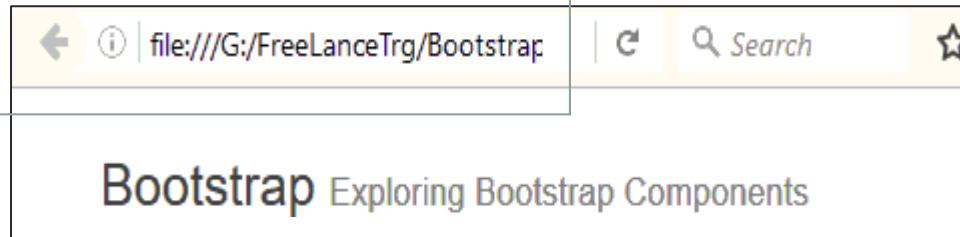
- **Page components form the basic structure of a web page.**
 - Examples include page headers, standout panels for displaying important info, nested comments sections, image thumbnails, and stacked lists of links
- **Page Headers :**
 - Eliminates efforts to neatly display a title with cleared browser default styles, the proper amount of spacing around it, and a small subtitle beside it

```
<div class="page-header">  
    <h1>BootStrap</h1>  
</div>
```



- To add a subtitle beside the title of the page, you can put it inside the same `<h1>` tag that we used before; wrap the subtitle inside a `<small></small>` tag

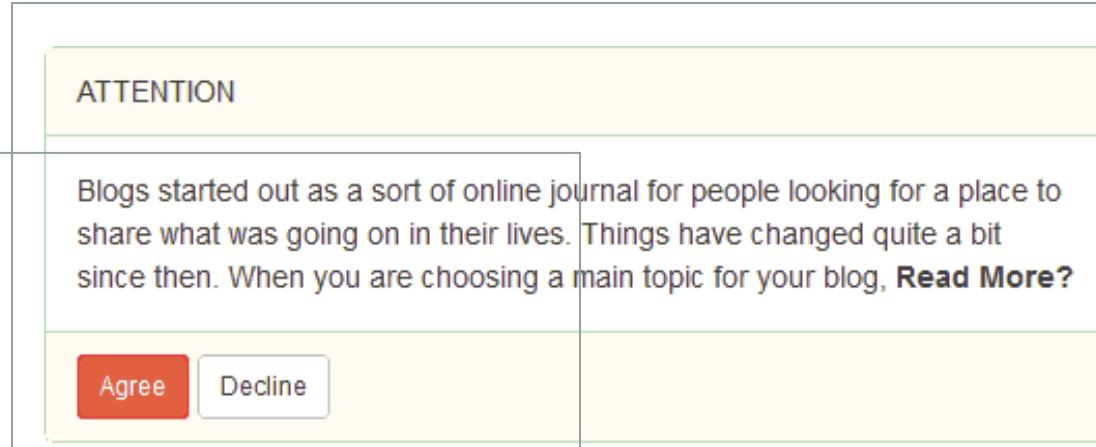
```
<div class="container">  
    <div class="page-header">  
        <h1> Bootstrap <small>Exploring Bootstrap Components  
</small></h1>  
    </div>  
</div>
```



Page Components : Panels

- **Panels are used to display text/images within a box with rounded corners.**
 - The panel div is divided into three parts: the panel-head, panel-body, and panel-footer. Each of these panel parts is optional.

```
<div class="panel panel-default">
  <div class="panel-heading">
    ATTENTION
  </div>
  <div class="panel-body">
    Body.....,
  </div>
  <div class="panel-footer">
    <a href="#" class="btn btn-danger btn-sm">Agree</a>
    <a href="#" class="btn btn-default btn-sm">Decline</a>
  </div>
</div>
```



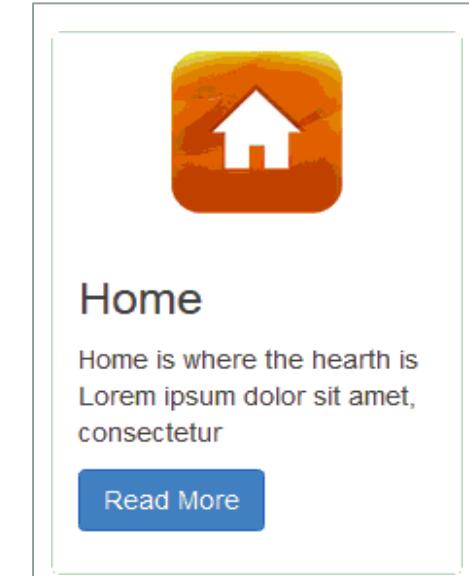
- Panels come with various color options

- panel-primary for dark blue
- panel-success for green
- panel-info for sky blue
- panel-warning for yellow
- panel-danger for red

Page Components : Thumbnails

- You can add some excerpts to each thumbnail and a Read More button for linking to different pages in your website.
 - Use `<div>` instead of `<a>`.
 - Then add an excerpt using `<p>` inside the “caption” div and a link with a “Read More” anchor and classes `btn` and `btn-primary` inside the same “caption” div.

```
<div class="col-xs-4">
  <div class="thumbnail">
    
    <div class="caption">
      <h3>Home</h3>
      <p>Home is where the hearth .....</p>
      <p>
        <a href="#" class="btn btn-primary">Read More</a>
      </p>
    </div>
  </div>
</div>
```

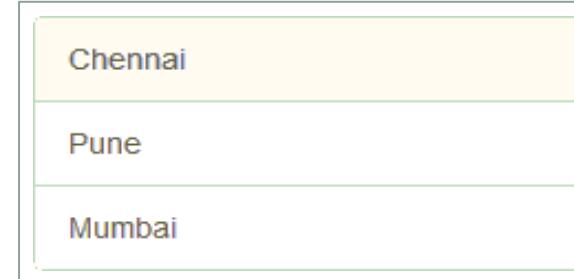


Page Components : List Group

- List group is used for creating lists, such as a list of useful resources or a list of recent activities.
 - You can also use it for a complex list of large amounts of textual content.
 - Add the **class list-group** to a **ul** element or a **div** element to make its children appear as a list.
 - The children can be **li** or **a** element, depending on your parent element choice.
 - The child should always have the class **list-group-item**.

```
<ul class="list-group">
  <li class="list-group-item">Inbox</li>
  <li class="list-group-item">Sent</li>
  <li class="list-group-item">Drafts</li>
  <li class="list-group-item">Deleted</li>
  <li class="list-group-item">Spam</li>
</ul>
```

```
<div class="list-group">
  <a href="#" class="list-group-item">Chennai</li>
  <a href="#" class="list-group-item">Pune</li>
  <a href="#" class="list-group-item">Mumbai</li>
</div>
```



Page Components : List Group

- We can display a number (eg to indicate pending notifications) beside each list item using the badge component.

- Add this inside each “list-group-item” to display a number beside each one : `14`
- The badges align themselves to the right of each list item

Chennai	14
Pune	22
Mumbai	5

```
<div class="list-group">
  <a href="#" class="list-group-item">Chennai<span class="badge">14</span></a>
  <a href="#" class="list-group-item">Pune<span class="badge">22</span></a>
  <a href="#" class="list-group-item">Mumbai<span class="badge">5</span></a>
</div>
```

- We can also apply various colors to each list item by adding list-group-item-* classes along with list-group-item.

- list-group-item-success for green
- list-group-item-info for sky blue
- list-group-item-warning for pale yellow
- list-group-item-danger for light red

Chennai	14
Pune	22
Mumbai	5

```
<li class="list-group-item list-group-item-success">Chennai</li>
```

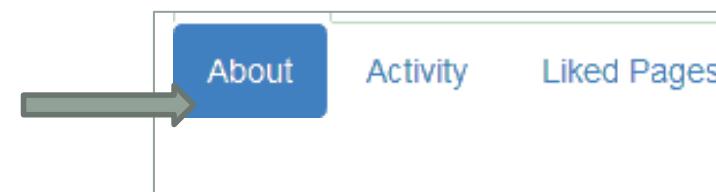
Navigation Components : Navs

- **Navs are a group of links that are placed inline with each other for navigation**
 - There are options to make this group of links appear either as tabs or small buttons, the latter known as pills in Bootstrap.
 - To create tab-like navigation links:

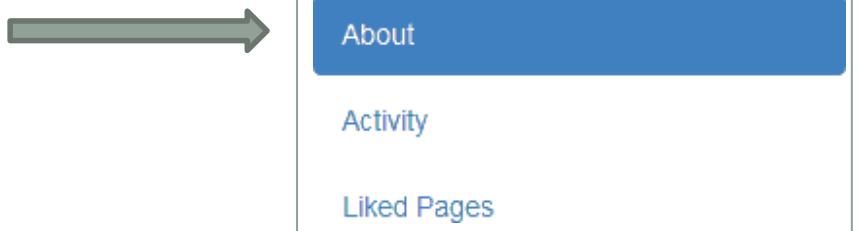
```
<ul class="nav nav-tabs">
  <li class="active"><a href="#">About</a></li>
  <li><a href="#">Activity</a></li>
  <li><a href="#">Liked Pages</a></li>
</ul>
```



```
<ul class="nav nav-pills">
  <li class="active"><a href="#">About</a></li>
  <li><a href="#">Activity</a></li>
  <li><a href="#">Liked Pages</a></li>
</ul>
```



- Vertically stack these pills by attaching an additional class **nav-stacked** to it



```
<body>
  <div class="container">

    <!-- tab-like navigation links -->
    <ul class="nav nav-tabs">
      <li class="active"><a href="#">About</a></li>
      <li><a href="#">Activity</a></li>
      <li><a href="#">Liked Pages</a></li>
    </ul>
    <!-- button (pills) navigation links -->
    <ul class="nav nav-pills nav-stacked">
      <li class="active"><a href="#">About</a></li>
      <li><a href="#">Activity</a></li>
      <li><a href="#">Liked Pages</a></li>
    </ul>
```

Navigation Components : Navbar

- **navbar gives you the power to generate portions of self-contained bars, which could be used as whole-application headers, versatile secondary menus for page content, or as a shell of various navigation-related elements.**
 - First build a div element, with two classes navbar and navbar-default.

```
<div class="navbar navbar-default">  
  </div>
```

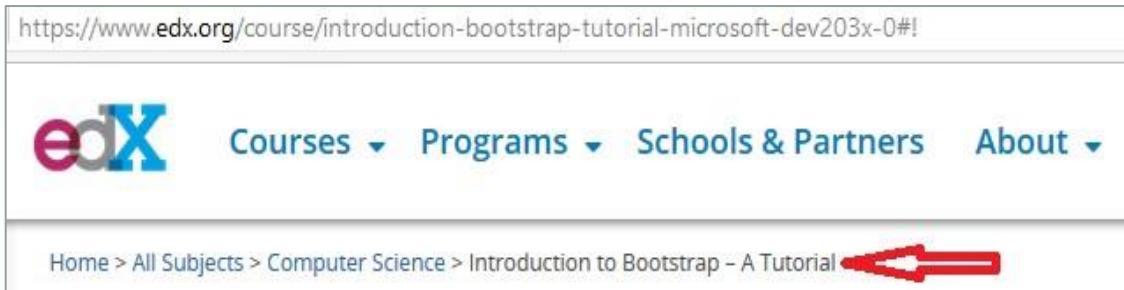
- Next, we'll use a div with class container-fluid inside this navbar element.

```
<div class="navbar navbar-default">  
  <div class="container-fluid">  
    </div>  
  </div>
```

```
<nav class = "navbar navbar-default" role = "navigation">
    <div class = "navbar-header">
        <a class = "navbar-brand" href = "#">Bootstrap Point</a>
    </div>
    <div>
        <ul class = "nav navbar-nav">
            <li class = "active"><a href = "#">iOS</a></li>
            <li><a href = "#">SVN</a></li>
            <li class = "dropdown">
                <a href = "#" class = "dropdown-toggle"
                   data-toggle = "dropdown">Java<b class = "caret"></b></a>
                <ul class = "dropdown-menu">
                    <li><a href = "#">jmeter</a></li>
                    <li><a href = "#">EJB</a></li>
                    <li><a href = "#">Jasper Report</a></li>
                    <li class = "divider"></li>
                    <li><a href = "#">Separated link</a></li>
                    <li class = "divider"></li>
                    <li><a href = "#">One more separated link</a></li>
                </ul>
            </li>
        </ul>
    </div>
</nav>
```

Navigation Components : Breadcrumbs

- Breadcrumbs are used to enhance the accessibility of your websites by indicating the location using a navigational hierarchy, especially in websites with a significant number of web pages.



```
<ol class="breadcrumb">
  <li><a href="#">Home</a></li>
  <li><a href="#">About</a></li>
  <li class="active">Author</li>
</ol>
```

You can use `` instead of ``

[Home](#) / [About](#) / [Author](#)

Standing Out : labels

- Labels are used to display short text beside other components, such as important messages and notes.
 - To display a label, you need to add a label class to inline HTML elements such as span and i.

Jump Start Bootstrap New

```
<h3>Jump Start Bootstrap <span class="label label-default">New</span></h3>
```

- class label-default is necessary to tell Bootstrap which variant of label we want to use.

The available label variants are:

- label-default for gray
- label-primary for dark blue
- label-success for green
- label-info for light blue
- label-warning for orange
- label-danger for red

```
<span class="label label-default">Default</span>
<span class="label label-success">Success</span>
<span class="label label-warning">Warning</span>
<span class="label label-primary">Important</span>
<span class="label label-info">Info</span>
<span class="label label-danger">Danger</span>
```

Default

Success

Warning

Important

Info

Danger

Standing Out : Buttons

- Its easy to convert an a, button, or input element into a fancy bold button in Bootstrap; just have to add the **btn** class

```
<a href="#" class="btn btn-primary">Click Here</a>
```

Click Here

- Buttons come in various color options:

- btn-default for white
- btn-primary for dark blue
- btn-success for green
- btn-info for light blue
- btn-warning for orange
- btn-danger for red

- And in various sizes:

- btn-lg for large buttons
- btn-sm for small buttons
- btn-xs for extra small button

Large button

Default button

Small button

Extra small button

```
<button type="button" class="btn btn-primary btn-lg">Large button</button>
<button type="button" class="btn btn-primary">Default button</button>
<button type="button" class="btn btn-primary btn-sm btn-success">Small button</button>
<button type="button" class="btn btn-primary btn-xs active">Extra small button</button>
```

Standing Out : Glyphicons

- **Glyphicons are used to display small icons.**
 - They are lightweight font icons and not images.
 - There are around 260 glyphicons available for buttons, navbars , lists and other components . Eg : To display a heart icon

```
<span class="glyphicon glyphicon-heart">  
</span>
```



```
<button type="submit" class="btn btn-default">  
<span class="glyphicon glyphicon-envelope"></span> Mail</button>  
<button type="submit" class="btn btn-default">  
<span class="glyphicon glyphicon-user"></span> Find Friends </button>  
<button type="submit" class="btn btn-warning">  
<span class="glyphicon glyphicon-trash"></span> Empty Trash </button>  
<button type="submit" class="btn btn-danger">  
<span class="glyphicon glyphicon-log-out"></span> Clean System</button></p>  
<button type="submit" class="btn btn-success">  
<span class="glyphicon glyphicon-log-out"></span> Logout</button>
```



Forms

- **Bootstrap greatly simplifies the process of styling and alignment of form controls like labels, input fields, selectboxes, textareas, buttons, etc. through predefined set of classes.**
 - Bootstrap provides three different types of form layouts:
 - Vertical Form (default form layout)
 - Horizontal Form
 - Inline Form
 - Standard rules for all three form layouts:
 - Wrap labels & form controls in `<div class="form-group">`
 - Add class `.form-control` to all `<input>`, `<textarea>`, and `<select>` elements
- **Construct a form with `form` element with the `form` class added to it**

The image shows a login form enclosed in a light gray border. The form fields are labeled with red text: "Name", "Enter Email Address as the Username", "Password", and "Browse to find file". The "Name" field contains the placeholder "Your Name". The "Enter Email Address as the Username" field contains the placeholder "Enter email". The "Password" field contains the placeholder "Password". Below these is a file upload field with a "Browse..." button and the message "No file selected.". There are two checkboxes: one checked labeled "Keep me signed in" and one unchecked labeled "Male". A radio button labeled "Female" is selected. At the bottom is a "Login" button.

Vertical form

```
<form class="form">
  <div class="form-group">
    <label for="enterusername"> Enter Email Address as the Username</label>
    <input type="email" class="form-control" id="enterusername" placeholder="Enter email">
  </div>
  <div class="form-group">
    <label for="enterpassword">Password</label>
    <input type="password" class="form-control" id="enterpassword" placeholder="Password">
  </div>
  <div class="form-group">
    <label for="filebrowse">Browse to find file</label>
    <input type="file" id="filebrowse">
  </div>
  <div class="checkbox">
    <label> <input type="checkbox" checked=""> Keep me signed in </label>
  </div>
  <div class="radio">
    <label><input type="radio" name="optionsRadios" value="option1" id="radio1">Male</label>
  </div>
  <div class="radio">
    <label><input type="radio" name="optionsRadios" value="option2" id="radio2">Female</label>
  </div>
  <br>
  <button type="submit" class="btn btn-default">Login</button>
</form>
```

Forms : Horizontal Forms

- In horizontal form layout labels are right aligned and floated to left to make them appear on the same line as form controls.

- Following markup changes required :

- Add the class `.form-horizontal` to the `<form>` element.
- Add the class `.control-label` to the `<label>` element.
- Use Bootstrap's predefined grid classes to align labels and form controls.



A screenshot of a web form with a light gray background. It contains two horizontal groups. The first group has a label "Email:" to the left of an input field with placeholder "Enter email". The second group has a label "Password:" to the left of an input field with placeholder "Enter password". Below these is a checkbox labeled "Remember me" and a "Submit" button at the bottom right.

```
<form class="form-horizontal">
  <div class="form-group"><label class="control-label col-sm-2" for="email">Email:</label>
    <div class="col-sm-10">
      <input type="email" class="form-control" id="email" placeholder="Enter email"></div></div>
  <div class="form-group"><label class="control-label col-sm-2" for="pwd">Password:</label>
    <div class="col-sm-10">
      <input type="password" class="form-control" id="pwd" placeholder="Enter password">
    </div></div>
  <div class="form-group">
    <div class="col-sm-offset-2 col-sm-10">
      <div class="checkbox"><label><input type="checkbox"> Remember me</label></div>
    </div></div>
  <div class="form-group">
    <div class="col-sm-offset-2 col-sm-10">
      <button type="submit" class="btn btn-default">Submit</button>
    </div> </div> </form>
```

Forms : Inline Form

- In an inline form, all of the elements are inline, left-aligned, and the labels are alongside.
 - Note: This only applies to forms within viewports that are at least 768px wide!
 - Add class `.form-inline` to the `<form>` element

```
<form class="form-inline">
  <div class="form-group">
    <label for="email">Email:</label>
    <input type="email" class="form-control" id="email" placeholder="Enter email">
  </div>
  <div class="form-group">
    <label for="pwd">Password:</label>
    <input type="password" class="form-control" id="pwd" placeholder="Enter password">
  </div>
  <div class="checkbox">
    <label><input type="checkbox"> Remember me</label>
  </div>
  <button type="submit" class="btn btn-default">Submit</button>
</form>
```

The image shows a screenshot of a web page with an inline form. The form consists of several input fields and labels. On the left, there is a label "Email:" followed by an input field containing the placeholder "Enter email". Next to it is a label "Password:" followed by another input field with the placeholder "Enter password". Below these is a label "Remember me" followed by a checkbox. To the right of the checkbox is a "Submit" button. All elements are arranged horizontally and left-aligned.

Bootstrap Typography

- **HTML uses default font and style to create headings, paragraphs, lists and other inline elements.**
 - Bootstrap overrides default and provides consistent styling across browsers for common typographic elements.
 - Eg, Bootstrap provides its own style for all six standard heading levels

```
<p class="text-muted">This text is muted.</p>
<p class="text-primary">This text is important.</p>
<p class="text-success">This text indicates success.</p>
<p class="text-info">This text represents some information.</p>
<p class="bg-primary">This text is important.</p>
<p class="bg-success">This text indicates success.</p>
<p class="bg-info">This text represents some information.</p>
<p class="text-lowercase">Lowercased text.</p>
<p class="text-uppercase">Uppercased text.</p>
<p class="text-capitalize">Capitalized text.</p>
<p class="text-left">Left-aligned text.</p>
<p class="text-right">Right-aligned text.</p>
<p class="text-center">Center-aligned text.</p>
```

Tables

- Bootstrap provides an efficient layout to build elegant tables
 - A basic Bootstrap table has a light padding and only horizontal dividers.
 - The .table class adds basic styling to a table

```
<table class = "table">
  <caption>Basic Table Layout</caption>
  <thead>
    <tr><th>Name</th><th>City</th></tr>
  </thead>
  <tbody>
    <tr><td>Soha</td><td>Bangalore</td></tr>
    <tr><td>Shrilata</td><td>Pune</td></tr>
  </tbody>
</table>
```

Basic Table Layout	
Name	City
Soha	Bangalore
Shrilata	Pune

Name	City
Soha	Bangalore
Shrilata	Pune
Sandeep	Mumbai
Sheela	Delhi

- The .table-striped class adds zebra-stripes to a table
- The .table-bordered class adds borders on all sides of the table and cells
- The .table-condensed class makes a table more compact by cutting cell padding in half

Jumbotron

- A jumbotron indicates a big box for calling extra attention to some special content or information.
 - A jumbotron is displayed as a grey box with rounded corners. It also enlarges the font sizes of the text inside it.
 - Tip: Inside a jumbotron you can put nearly any valid HTML, including other Bootstrap elements/classes.
 - Use a `<div>` element with class `.jumbotron` to create a jumbotron

```
<div class="jumbotron">
  <h1>Bootstrap Tutorial</h1>
  <p>Bootstrap is the most popular HTML, CSS, and JS framework for
     developing responsive, mobile-first projects on the web.</p>
</div>
```

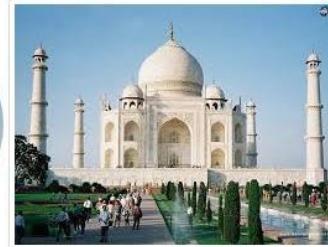
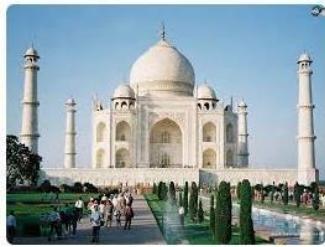
Bootstrap Tutorial

Bootstrap is the most popular HTML, CSS, and JS framework for developing responsive, mobile-first projects on the web.

images

- To add images on the webpage use element **** , it has three classes to apply simple styles to images.
 - .img-rounded : To add rounded corners around the edges of the image, radius of the border is **6px**.
 - .img-circle : To create a circle of radius is **500px**
 - .img-thumbnail : To add some padding with grey border , making the image look like a polaroid photo.

```
 <!-- rounded edges-->
 <!-- circle -->
 <!-- thumbnail -->
```



Alerts

- **Bootstrap comes with a very useful component for displaying alert messages in various sections of our website**
 - You can use them for displaying a success message, a warning message, a failure message, or an information message.
 - These messages can be annoying to visitors, hence they should have dismiss functionality added to give visitors the ability to hide them.

```
<div class="alert alert-success">  
    Amount has been transferred successfully.  
</div>
```

```
<div class="alert alert-success alert-dismissible">  
    <button type="button" class="close" data-dismiss="alert">&times;</button>  
    Amount has been transferred successfully.  
</div>
```

