B.C.A. (Sem – VI)

B.C.A. - 601

Building Application Using PHP

Purushottam Singh

Unit:- 1

Unit-1

Introduction to PHP: -

Web-Technologies: -

- There are many Web technologies, from simple to complex.
- They help you get started with developing your own Web sites.

1. **Markup Languages:-**
   - Markup is used to in text and word processing documents to describe how a document should look when displayed or printed.
   - The Internet uses markup to define how Web pages should look when displayed in a browser or to define the data contained within a Web document.
2. **PHP:-**
   - PHP is an interpreted scripting language that is used as an alternative to ASP on UNIX-based servers.
   - PHP is commonly used to access databases and provide server-side form and e-commerce processing.
   - As with ASP code, PHP code is contained within the body of an HTML page.
   - PHP code typically runs on Linux-based and UNIX-based Web servers, and can run on Windows-based servers with an installed interpreter.
   - "PHP, originally derived from Personal Home Page Tools, now stands for PHP: Hypertext Preprocessor."
3. **HTML:-**
   - HTML stands for Hypertext Markup Language.
   - HTML is the primary markup language that is used for Web pages.
   - HTML tells the browser what to display on a page. For example, it specifies text, images, and other objects and can also specify the appearance of text, such as bold or italic text.
4. **CSS:-**
   - CSS stands for cascading style sheets.
   - Cascading style sheets provide the ability to change the appearance of text (such as fonts, colors, spacing) on Web pages.
   - Using CSS, you can also position elements on the page, make certain elements hidden, or change the appearance of the browser, such as changing the color of scroll bars.
5. **XML:-**
   - XML stands for Extensible Markup Language.
   - Similar to HTML, XML is a markup language designed for the Internet.
   - However, unlike HTML, which was designed to define formatting of Web pages, XML was designed to describe data.
   - You can use XML to develop custom markup languages.
6. **JAVA Script (Jscript):-**
   - JavaScript is an interpreted scripting language commonly used on the Internet for creating Web pages that respond to user actions, such as when a user moves a mouse pointer over an image or clicks a form button.
   - Combined with HTML and CSS, JavaScript allows you to create Dynamic HTML pages.
   - JavaScript is generally used for client-side scripting; as a result, users can easily view JavaScript code along with the HTML code in a page.
7. **Programming Languages and Technologies:-**
   - Programming languages enable you to create custom applications and add functionality that is not already part of an application.
   - On the Internet, programming languages enable you to create visual animation, respond to user actions, validate forms, interact with databases, and provide e-commerce solutions.

Server Side Technologies:-

- Server-side scripting refers to the dynamic generation of Web pages served up by the Web server, as opposed to "static" web pages in the server storage that are served up to the Web browser.
- In other words, some part of the content sent in response to a HTTP request is determined on-the-fly by a program that executes on the server after the HTTP request has been received and generates content as a result of the execution.
- Insertion of continuously changing content into a web page, for example - weather or stock quotes.
- Authentication, authorization and session tracking - although rudimentary authentication and authorization is supported by most Web servers.

- Template-driven page generation. Including repeated content like header/footers and navigation menus around the "content area" of a web page.
- Dynamic image generation, e.g. page counters, human-readable characters for security, maps, overlays etc.
- Device mapping - generating different types of content (HTML, XMI , WML) based on the user agent that sent the HTTP request.

Web-Server:-

- A Web server is a program that uses HTTP (Hypertext Transfer Protocol) to serve the files that form Web pages to users, in response to their requests, which are forwarded by their computers' HTTP clients.
- Dedicated computers and appliances may be referred to as Web servers as well.
- The process is an example of the client/server model.
- All computers that host Web sites must have Web server programs.
- Leading Web servers include Apache (the most widely-installed Web server), Microsoft's Internet Information Server (IIS) and nginx (pronounced engine X) from NGNIX.
- Other Web servers include Novell's NetWare server, Google Web Server (GWS) and IBM's family of Domino servers.
- Web servers often come as part of a larger package of Internet- and intranet-related programs for serving email, downloading requests for File Transfer Protocol (FTP) files, and building and publishing Web pages.

Web-Browser:-

- A web browser (commonly referred to as a browser) is a software application for retrieving, presenting, and traversing information resources on the World Wide Web.
- An information resource is identified by a Uniform Resource Identifier (URI/URL) and may be a web page, image, video or other piece of content.
- A browser is an application program that provides a way to look at and interact with all the information on the World Wide Web.
- The word "browser" seems to have originated prior to the Web as a generic term for user interfaces that let you browse (navigate through and read) text files online.

Introduction of PHP:-

- PHP started out as a small open source project that evolved as more and more people found out how useful it was.
- Rasmus Lerdorf unleashed the first version of PHP way back in 1994.
- PHP is a recursive acronym for "PHP: Hypertext Preprocessor".
- PHP is a server side scripting language that is embedded in HTML. It is used to manage dynamic content, databases, session tracking, even build entire e-commerce sites.
- It is integrated with a number of popular databases, including MySQL, PostgreSQL, Oracle, Sybase, Informix, and Microsoft SQL Server.
- PHP Syntax is C-Like.
- PHP performs system functions, i.e. from files on a system it can create, open, read, write, and close them.
- PHP can handle forms, i.e. gather data from files, save data to a file, through email you can send data, return data to the user.
- Access cookies variables and set cookies.
- Using PHP, you can restrict users to access some pages of your website.

Future of PHP:-

- 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, and 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, and even Flash movies. You can also output any text, such as XHTML and XML.
- 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.
- PHP is free. Download it from the official PHP resource: www.php.net.
- PHP is easy to learn and runs efficiently on the server side.

Client Side Scriptiong:-

- The client-side environment used to run scripts is usually a browser.
- The processing takes place on the end users computer.
- The source code is transferred from the web server to the users computer over the internet and run directly in the browser.
- The scripting language needs to be enabled on the client computer.
- Client-side scripting generally refers to the class of computer programs on the web that are executed client-side, by the user's web browser, instead of server-side (on the web server).
- This type of computer programming is an important part of the Dynamic HTML (DHTML) concept, enabling web pages to be scripted; that is, to have different and changing content depending on user input, environmental conditions (such as the time of day), or other variables.

PHP Syntax:-

- A PHP script is executed on the server, and the plain HTML result is sent back to the browser.
- A PHP script can be placed anywhere in the document.
- A PHP script starts with

```
<?php
// PHP code goes here
?>
```

- The default file extension for PHP files is ".php".
- A PHP file normally contains HTML tags, and some PHP scripting code.
- Below, we have an example of a simple PHP file, with a PHP script that uses a built-in PHP function "echo" to output the text "Hello World!" on a web page.
- Example:-

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

<h1>My first PHP page</h1>

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

</body>
</html>
```

PHP Variables:-

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

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

- 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.

- **Rules for PHP variables:**
  1. A variable starts with the $ sign, followed by the name of the variable
  2. A variable name must start with a letter or the underscore character
  3. A variable name cannot start with a number
  4. A variable name can only contain alpha-numeric characters and underscores (A-z, 0-9, and _)
  5. Variable names are case-sensitive ($age and $AGE are two different variables)

Control Structure:-

- In PHP we have the following conditional statements:
  - if statement - executes some code if one condition is true
  - if...else statement - executes some code if a condition is true and another code if that condition is false
  - if...elseif....else statement - executes different codes for more than two conditions
  - switch statement - selects one of many blocks of code to be executed

PHP - The if Statement:-

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

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

- The example below will output "Have a good day!" if the current time (HOUR) is less than 20.

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

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

PHP - The if...else Statement:-

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

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

- The example below will output "Have a good day!" if the current time is less than 20, and "Have a good night!" otherwise:

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

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

PHP - The if...elseif....else Statement:-

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

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

- The example below will output "Have a good morning!" if the current time is less than 10, and "Have a good day!" if the current time is less than 20. Otherwise it will output "Have a good night!":

```
<?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!";
} ?>
```

The PHP switch Statement:-

- The switch statement is used to perform different actions based on different conditions.
- Use the switch statement to select one of many blocks of code to be executed.

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

- First we have a single expression n (most often a variable), that is evaluated once.
- The value of the expression is then compared with the values for each case in the structure.
- If there is a match, the block of code associated with that case is executed.
- Use break to prevent the code from running into the next case automatically.
- The default statement is used if no match is found.

```
<?php
$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, nor green!";
}
?>
```


PHP User Defined Functions:-

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

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

- Example

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

PHP Function Arguments:-

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

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

familyName("Jani");
familyName("Hege");
familyName("Stale");
familyName("Kai Jim");
familyName("Borge");
?>
```

Passing Information Between Pages:-

- We often have to pass values of variables between pages in our site.
- These are required in many different ways. Some time we collect some value from a database and want to retain the value for the particular user throughout the site as the user moves between pages.
- There are different ways for passing such values of the variables. It can be passed within a site or even outside the site.
- We will discuss some of the ways with their possible uses.
    1. Passing variable values between pages using session.
    2. Passing variables between pages using cookies.
    3. Passing variables between pages using URL.
    4. Passing variables between pages using Web Forms.
- These are some of the way of transferring data variables between pages.

Uses	Method	Reason
Login form with Userid and password	POST	Id and password should not be visible in query string
Sending Product id	GET	The url can be bookmarked for future uses.
Feedback form	Post	Several data can be posted
Favorite buying pattern of User	Cookies	Data stored in user computer
User Login status	Session	Secured data is maintained at server end

- Think SECURITY when processing PHP forms!
    1. This page does not contain any form validation, it just shows how you can send and retrieve form data.
    2. However, the next pages will show how to process PHP forms with security in mind! Proper validation of form data is important to protect your form from hackers and spammers!
- Information sent from a form with the GET method is visible to everyone (all variable names and values are displayed in the URL). GET should NEVER be used for sending passwords or other sensitive information!
- Information sent from a form with the POST method is invisible to others (all names/values are embedded within the body of the HTTP request) and has no limits on the amount of information to send. Developers prefer POST for sending form data.
- Moreover, POST supports advanced functionality such as support for multi-part binary input while uploading files to server.

Variable Function:-

Function Name	Syntax	Description
gettype	gettype ($var)	Get the type of a variable. Returns the type of the PHP variable $var.
settype	settype(var_name, var_type)	The settype() function is used to set the type of a variable.
isset	isset(variable1, variable2.....)	The isset () function is used to check whether a variable is set or not. If a variable is already unset with unset() function, it will no longer be set. The isset() function return false if testing variable contains a NULL value.
unset	unset (var1, var2....)	The unset() function destroys a given variable.
strval-	strval(var_name)	The strval() is used to convert a value or a variable to a string.
floatval	floatval (var1)	The floatval() function is used to convert a value to a float.
inval	intval(var_name, base)	The intval() function is used to get the integer value of a variable.
print_r	print_r(var_name, return_output);	The print_r() function is used to print human-readable information about a variable.

String Functions:-

Function	Description
addcslashes()	Returns a string with backslashes in front of the specified characters
addslashes()	Returns a string with backslashes in front of predefined characters
bin2hex()	Converts a string of ASCII characters to hexadecimal values
chop()	Removes whitespace or other characters from the right end of a string
chr()	Returns a character from a specified ASCII value
chunk_split()	Splits a string into a series of smaller parts
convert_cyr_string()	Converts a string from one Cyrillic character-set to another
convert_uudecode()	Decodes a uuencoded string
convert_uuencode()	Encodes a string using the uuencode algorithm
count_chars()	Returns information about characters used in a string
crc32()	Calculates a 32-bit CRC for a string
crypt()	One-way string hashing
echo()	Outputs one or more strings
explode()	Breaks a string into an array
fprintf()	Writes a formatted string to a specified output stream

get_html_translation_table()	Returns the translation table used by htmlspecialchars() and htmlentities()
hebrev()	Converts Hebrew text to visual text
hebrevc()	Converts Hebrew text to visual text and new lines (\n) into
hex2bin()	Converts a string of hexadecimal values to ASCII characters
html_entity_decode()	Converts HTML entities to characters
htmlentities()	Converts characters to HTML entities
htmlspecialchars_decode()	Converts some predefined HTML entities to characters
htmlspecialchars()	Converts some predefined characters to HTML entities
implode()	Returns a string from the elements of an array
join()	Alias of implode()
lcfirst()	Converts the first character of a string to lowercase
levenshtein()	Returns the Levenshtein distance between two strings
localeconv()	Returns locale numeric and monetary formatting information
ltrim()	Removes whitespace or other characters from the left side of a string
md5()	Calculates the MD5 hash of a string
md5_file()	Calculates the MD5 hash of a file
metaphone()	Calculates the metaphone key of a string
money_format()	Returns a string formatted as a currency string
nl_langinfo()	Returns specific local information
nl2br()	Inserts HTML line breaks in front of each newline in a string
number_format()	Formats a number with grouped thousands
ord()	Returns the ASCII value of the first character of a string
parse_str()	Parses a query string into variables
print()	Outputs one or more strings
printf()	Outputs a formatted string
quoted_printable_decode()	Converts a quoted-printable string to an 8-bit string
quoted_printable_encode()	Converts an 8-bit string to a quoted printable string
quotemeta()	Quotes meta characters
rtrim()	Removes whitespace or other characters from the right side of a string
setlocale()	Sets locale information
sha1()	Calculates the SHA-1 hash of a string
sha1_file()	Calculates the SHA-1 hash of a file
similar_text()	Calculates the similarity between two strings

soundex()	Calculates the soundex key of a string
sprintf()	Writes a formatted string to a variable
sscanf()	Parses input from a string according to a format
str_getcsv()	Parses a CSV string into an array
str_ireplace()	Replaces some characters in a string (case-insensitive)
str_pad()	Pads a string to a new length
str_repeat()	Repeats a string a specified number of times
str_replace()	Replaces some characters in a string (case-sensitive)
str_rot13()	Performs the ROT13 encoding on a string
str_shuffle()	Randomly shuffles all characters in a string
str_split()	Splits a string into an array
str_word_count()	Count the number of words in a string
strcasecmp()	Compares two strings (case-insensitive)
strchr()	Finds the first occurrence of a string inside another string (alias of strstr())
strcmp()	Compares two strings (case-sensitive)
strcoll()	Compares two strings (locale based string comparison)
strcspn()	Returns the number of characters found in a string before any part of some specified characters are found
strip_tags()	Strips HTML and PHP tags from a string
stripcslashes()	Unquotes a string quoted with addcslashes()
stripslashes()	Unquotes a string quoted with addslashes()
stripos()	Returns the position of the first occurrence of a string inside another string (case-insensitive)
stristr()	Finds the first occurrence of a string inside another string (case-insensitive)
strlen()	Returns the length of a string
strnatcasecmp()	Compares two strings using a "natural order" algorithm (case-insensitive)
strnatcmp()	Compares two strings using a "natural order" algorithm (case-sensitive)
strncasecmp()	String comparison of the first n characters (case-insensitive)
strncmp()	String comparison of the first n characters (case-sensitive)
strpbrk()	Searches a string for any of a set of characters
strpos()	Returns the position of the first occurrence of a string inside another string (case-sensitive)
strrchr()	Finds the last occurrence of a string inside another string
strrev()	Reverses a string
strripos()	Finds the position of the last occurrence of a string inside another string (case-insensitive)

strrpos()	Finds the position of the last occurrence of a string inside another string (case-sensitive)
strspn()	Returns the number of characters found in a string that contains only characters from a specified charlist
strstr()	Finds the first occurrence of a string inside another string (case-sensitive)
strtok()	Splits a string into smaller strings
strtolower()	Converts a string to lowercase letters
strtoupper()	Converts a string to uppercase letters
strtr()	Translates certain characters in a string
substr()	Returns a part of a string
substr_compare()	Compares two strings from a specified start position (binary safe and optionally case-sensitive)
substr_count()	Counts the number of times a substring occurs in a string
substr_replace()	Replaces a part of a string with another string
trim()	Removes whitespace or other characters from both sides of a string
ucfirst()	Converts the first character of a string to uppercase
ucwords()	Converts the first character of each word in a string to uppercase
vfprintf()	Writes a formatted string to a specified output stream
vprintf()	Outputs a formatted string
vsprintf()	Writes a formatted string to a variable
wordwrap()	Wraps a string to a given number of characters

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

Array:-

- An array stores multiple values in one single variable.
- An array is a special variable, which can hold more than one value at a time.
- If you have a list of items (a list of car names, for example), storing the cars in single variables could look like this:

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

(1) Numeric array

```
foreach($age as $value)
{
  echo $value;
}
```

- In PHP, there are three types of arrays.
- Indexed arrays - Arrays with a numeric index.
- Associative arrays - Arrays with named keys.
- Multidimensional arrays - Arrays containing one or more arrays.

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

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

(2) Associative array

```
echo $age['Peter'];
echo $age['Ben'];
```

- **Some Sort Functions For Arrays**
  1. sort() - sort arrays in ascending order
  2. rsort() - sort arrays in descending order
  3. asort() - sort associative arrays in ascending order, according to the value
  4. ksort() - sort associative arrays in ascending order, according to the key
  5. arsort() - sort associative arrays in descending order, according to the value
  6. krsort() - sort associative arrays in descending order, according to the key

Array Functions;-

Function	Description
array()	Creates an array
array_change_key_case()	Changes all keys in an array to lowercase or uppercase
array_chunk()	Splits an array into chunks of arrays
array_column()	Returns the values from a single column in the input array
array_combine()	Creates an array by using the elements from one "keys" array and one "values" array
array_count_values()	Counts all the values of an array
array_diff()	Compare arrays, and returns the differences (compare values only)
array_diff_assoc()	Compare arrays, and returns the differences (compare keys and values)
array_diff_key()	Compare arrays, and returns the differences (compare keys only)
array_diff_uassoc()	Compare arrays, and returns the differences (compare keys and values, using a user-defined key comparison function)
array_diff_ukey()	Compare arrays, and returns the differences (compare keys only, using a user-defined key comparison function)
array_fill()	Fills an array with values
array_fill_keys()	Fills an array with values, specifying keys
array_filter()	Filters the values of an array using a callback function
array_flip()	Flips/Exchanges all keys with their associated values in an array
array_intersect()	Compare arrays, and returns the matches (compare values only)
array_intersect_assoc()	Compare arrays and returns the matches (compare keys and values)
array_intersect_key()	Compare arrays, and returns the matches (compare keys only)
array_intersect_uassoc()	Compare arrays, and returns the matches (compare keys and values, using a user-defined key comparison function)
array_intersect_ukey()	Compare arrays, and returns the matches (compare keys only, using a user-defined key comparison function)
array_key_exists()	Checks if the specified key exists in the array
array_keys()	Returns all the keys of an array
array_map()	Sends each value of an array to a user-made function, which returns new values
array_merge()	Merges one or more arrays into one array
array_merge_recursive()	Merges one or more arrays into one array recursively
array_multisort()	Sorts multiple or multi-dimensional arrays

array_pad()	Inserts a specified number of items, with a specified value, to an array
array_pop()	Deletes the last element of an array
array_product()	Calculates the product of the values in an array
array_push()	Inserts one or more elements to the end of an array
array_rand()	Returns one or more random keys from an array
array_reduce()	Returns an array as a string, using a user-defined function
array_replace()	Replaces the values of the first array with the values from following arrays
array_replace_recursive()	Replaces the values of the first array with the values from following arrays recursively
array_reverse()	Returns an array in the reverse order
array_search()	Searches an array for a given value and returns the key
array_shift()	Removes the first element from an array, and returns the value of the removed element
array_slice()	Returns selected parts of an array
array_splice()	Removes and replaces specified elements of an array
array_sum()	Returns the sum of the values in an array
array_udiff()	Compare arrays, and returns the differences (compare values only, using a user-defined key comparison function)
array_udiff_assoc()	Compare arrays, and returns the differences (compare keys and values, using a built-in function to compare the keys and a user-defined function to compare the values)
array_udiff_uassoc()	Compare arrays, and returns the differences (compare keys and values, using two user-defined key comparison functions)
array_uintersect()	Compare arrays, and returns the matches (compare values only, using a user-defined key comparison function)
array_uintersect_assoc()	Compare arrays, and returns the matches (compare keys and values, using a built-in function to compare the keys and a user-defined function to compare the values)
array_uintersect_uassoc()	Compare arrays, and returns the matches (compare keys and values, using two user-defined key comparison functions)
array_unique()	Removes duplicate values from an array
array_unshift()	Adds one or more elements to the beginning of an array
array_values()	Returns all the values of an array
array_walk()	Applies a user function to every member of an array
array_walk_recursive()	Applies a user function recursively to every member of an array
arsort()	Sorts an associative array in descending order, according to the value
asort()	Sorts an associative array in ascending order, according to the value
compact()	Create array containing variables and their values
count()	Returns the number of elements in an array
current()	Returns the current element in an array
each()	Returns the current key and value pair from an array

end()	Sets the internal pointer of an array to its last element
extract()	Imports variables into the current symbol table from an array
in_array()	Checks if a specified value exists in an array
key()	Fetches a key from an array
krsort()	Sorts an associative array in descending order, according to the key
ksort()	Sorts an associative array in ascending order, according to the key
list()	Assigns variables as if they were an array
natcasesort()	Sorts an array using a case insensitive "natural order" algorithm
natsort()	Sorts an array using a "natural order" algorithm
next()	Advance the internal array pointer of an array
pos()	Alias of current()
prev()	Rewinds the internal array pointer
range()	Creates an array containing a range of elements
reset()	Sets the internal pointer of an array to its first element
rsort()	Sorts an indexed array in descending order
shuffle()	Shuffles an array
sizeof()	Alias of count()
sort()	Sorts an indexed array in ascending order
uasort()	Sorts an array by values using a user-defined comparison function
uksort()	Sorts an array by keys using a user-defined comparison function
usort()	Sorts an array using a user-defined comparison function

Date Functions:-

- The date/time functions allow you to get the date and time from the server where your PHP script runs. You can then use the date/time functions to format the date and time in several ways.
- Note: These functions depend on the locale settings of your server. Remember to take daylight saving time and leap years into consideration when working with these functions.

Function	Description
checkdate()	Validates a Gregorian date
date_add()	Adds days, months, years, hours, minutes, and seconds to a date
date_create_from_format()	Returns a new DateTime object formatted according to a specified format
date_create()	Returns a new DateTime object
date_date_set()	Sets a new date
date_default_timezone_get()	Returns the default timezone used by all date/time functions
date_default_timezone_set()	Sets the default timezone used by all date/time functions

date_diff()	Returns the difference between two dates
date_format()	Returns a date formatted according to a specified format
date_get_last_errors()	Returns the warnings/errors found in a date string
date_interval_create_from_date_string()	Sets up a DateInterval from the relative parts of the string
date_interval_format()	Formats the interval
date_isodate_set()	Sets the ISO date
date_modify()	Modifies the timestamp
date_offset_get()	Returns the timezone offset
date_parse_from_format()	Returns an associative array with detailed info about a specified date, according to a specified format
date_parse()	Returns an associative array with detailed info about a specified date
date_sub()	Subtracts days, months, years, hours, minutes, and seconds from a date
date_sun_info()	Returns an array containing info about sunset/sunrise and twilight begin/end, for a specified day and location
date_sunrise()	Returns the sunrise time for a specified day and location
date_sunset()	Returns the sunset time for a specified day and location
date_time_set()	Sets the time
date_timestamp_get()	Returns the Unix timestamp
date_timestamp_set()	Sets the date and time based on a Unix timestamp
date_timezone_get()	Returns the time zone of the given DateTime object
date_timezone_set()	Sets the time zone for the DateTime object
date()	Formats a local date and time
getdate()	Returns date/time information of a timestamp or the current local date/time
gettimeofday()	Returns the current time
gmdate()	Formats a GMT/UTC date and time
gmmktime()	Returns the Unix timestamp for a GMT date
gmstrftime()	Formats a GMT/UTC date and time according to locale settings
idate()	Formats a local time/date as integer
localtime()	Returns the local time
microtime()	Returns the current Unix timestamp with microseconds
mktime()	Returns the Unix timestamp for a date
strftime()	Formats a local time and/or date according to locale settings
strptime()	Parses a time/date generated with strftime()
strtotime()	Parses an English textual datetime into a Unix timestamp

time()	Returns the current time as a Unix timestamp
timezone_abbreviations_list()	Returns an associative array containing dst, offset, and the timezone name
timezone_identifiers_list()	Returns an indexed array with all timezone identifiers
timezone_location_get()	Returns location information for a specified timezone
timezone_name_from_abbr()	Returns the timezone name from abbreviation
timezone_name_get()	Returns the name of the timezone
timezone_offset_get()	Returns the timezone offset from GMT
timezone_open()	Creates new DateTimeZone object
timezone_transitions_get()	Returns all transitions for the timezone
timezone_version_get()	Returns the version of the timezone db

Working with Forms:-

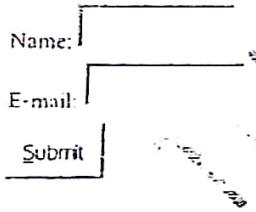
Creating and Handling Form:-

- A Document that containing black fields, that the user can fill the data or user can select the data.Casually the data will store in the data base.
- The PHP superglobals $_GET and $_POST are used to collect form-data.
- The example below displays a simple HTML form with two input fields and a submit button.

```
<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>
```



- When the user fills out the form above and clicks the submit button, the form data is sent for processing to a PHP file named "welcome.php". The form data is sent with the HTTP POST method.
- To display the submitted data you could simply echo all the variables. The "welcome.php" looks like this:

```
<html>
<body>

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

</body>
</html>
```

- The output could be something like this:

  Welcome John
  Your email address is john.doe@example.com

Form Validation:-

- The validation rules for the form above are as follows:

Field	Validation Rules
Name	Required. + Must only contain letters and whitespace
E-mail	Required. + Must contain a valid email address (with @ and .)
Website	Optional. If present, it must contain a valid URL
Comment	Optional. Multi-line input field (textarea)
Gender	Required. Must select one

- Note:- The $_SERVER["PHP_SELF"] is a super global variable that returns the filename of the currently executing script.
- So, the $_SERVER["PHP_SELF"] sends the submitted form data to the page itself, instead of jumping to a different page. This way, the user will get error messages on the same page as the form.
- The htmlspecialchars() function converts special characters to HTML entities. This means that it will replace HTML characters like < and > with < and >. This prevents attackers from exploiting the code by injecting HTML or Javascript code (Cross-site Scripting attacks) in forms.

```
<?php
// define variables and set to empty values
$name = $email = $gender = $comment = $website = "";

if ($_SERVER["REQUEST_METHOD"] == "POST") {
  $name = test_input($_POST["name"]);
  $email = test_input($_POST["email"]);
  $website = test_input($_POST["website"]);
  $comment = test_input($_POST["comment"]);
  $gender = test_input($_POST["gender"]);
}

function test_input($data) {
  $data = trim($data);
  $data = stripslashes($data);
  $data = htmlspecialchars($data);
  return $data;
}
?>
```

PHP - Required Fields:-

- From the validation rules table on the previous page, we see that the "Name", "E-mail", and "Gender" fields are required. These fields cannot be empty and must be filled out in the HTMl form.

Field	Validation Rules
Name	Required. + Must only contain letters and whitespace
E-mail	Required. + Must contain a valid email address (with @ and .)
Website	Optional. If present, it must contain a valid URL
Comment	Optional. Multi-line input field (textarea)
Gender	Required. Must select one

- In the following code we have added some new variables: $nameErr, $emailErr, $genderErr, and $websiteErr.
- These error variables will hold error messages for the required fields. We have also added an IF else statement for each $_POST variable. This checks if the $_POST variable is empty (with the PHP empty() function).

- If it is empty, an error message is stored in the different error variables, and if it is not empty, it sends the user input data through the test_input() function:

```
<?php
// define variables and set to empty values
$nameErr = $emailErr = $genderErr = $websiteErr = "";
$name = $email = $gender = $comment = $website = "";

if ($_SERVER["REQUEST_METHOD"] == "POST") {
  if (empty($_POST["name"])) {
    $nameErr = "Name is required";
  } else {
    $name = test_input($_POST["name"]);
  }

  if (empty($_POST["email"])) {
    $emailErr = "Email is required";
  } else {
    $email = test_input($_POST["email"]);
  }

  if (empty($_POST["website"])) {
    $website = "";
  } else {
    $website = test_input($_POST["website"]);
  }

  if (empty($_POST["comment"])) {
    $comment = "";
  } else {
    $comment = test_input($_POST["comment"]);
  }

  if (empty($_POST["gender"])) {
    $genderErr = "Gender is required";
  } else {
    $gender = test_input($_POST["gender"]);
  }
}
?>
```

File Upload:-

- With PHP, it is easy to upload files to the server.
- However, with ease comes danger, so always be careful when allowing file uploads!
- 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

```
<?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;
  }
}
```

```
// Check if file already exists
if (file_exists($target_file)) {
    echo "Sorry, file already exists.";
    $uploadOk = 0;
}
// Check file size
if ($_FILES["fileToUpload"]["size"] > 500000) {
    echo "Sorry, your file is too large.";
    $uploadOk = 0;
}
// Allow certain file formats
if($imageFileType != "jpg" && $imageFileType != "png" && $imageFileType != "jpeg"
&& $imageFileType != "gif" ) {
    echo "Sorry, only JPG, JPEG, PNG & GIF files are allowed.";
    $uploadOk = 0;
}
// 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.";
    }
}
?>

<!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>
```

Send an Email on Form Submission Using PHP:-

- Sending Mail to specific email becomes a global issue for website development.
- Online communication for any organisation with customers or users is done either by emailing them or through comments.
- These organisations or companie's feedback or contact form to communicate with users.
- Through these forms user is able to send his/her suggestion or feedback via email to respective organisation.
- Use mail() function of PHP to send informations like, suggestions/messages to specific email address on form submission.
- we used following PHP mail() function with four parameters to send email as follows:

```
mail("$to", $subject, $message, $headers);
```

- Here, $to variable is to store reciever's email id.
- $subject is a variable to store mail subject.
- $message is a variable to store user's message.
- $headers contains other email parameters like BCc, Cc etc.

```
<?php
//if "email" variable is filled out, send email
  if (isset($_REQUEST['email']))  {
```

```
//Email information
$admin_email = "someone@example.com";
$email = $_REQUEST['email'];
$subject = $_REQUEST['subject'];
$comment = $_REQUEST['comment'];

//send email
mail($admin_email, "$subject", $comment, "From:" . $email);

//Email response
echo "Thank you for contacting us!";
}

//if "email" variable is not filled out, display the form
else {
?>

<form method="post">
Email: <input name="email" type="text" /><br />
Subject: <input name="subject" type="text" /><br />
Message:<br />
<textarea name="comment" rows="15" cols="40"></textarea><br />
<input type="submit" value="Submit" />
</form>

<?php
}
?>
```


① Numeric array :- These arrays can store numbers, strings and any object but their index will be represented by numbers. By default array index starts from zero.

② Associative array :- The associative arrays are very similar to numeric arrays in terms of functionality but they are different in terms of their index.

→ This array can establish a strong association between key and values.

Ex. second method `$age['peter'] = "35";`

③ multidimentional array :- A multi-dimensional arrays each element in the main array can also be an array. And each element in the sub-array can be an array and so on. Values in the multi-dimentional array are accessed using multiple index.

Ex.

```
$marks = array("pooja" => array ('php' => 35,
                                 ".Net" => 30,
                                 "Dwom => 39),

         "Nikhil" => array("php" => 38,
                           ".Net" => 50,
                           "Dwom => 40),

         "Payal" => array("php" => 41,
                          ".Net" => 34,
                          "Dwom => 35);
);

echo  $marks['pooja']['php'];
echo  $meavis['Nikhil']['.net'];
ech   $meavis['Payal']['Dwom'];
```