WEB PROGRAMING

PHP (Hypertext Preprocessor Or Personal Home Page)

UNIT – 1 (PHP Basic)

❖ **What is website ?**

- A set of related web pages will create a website. The web pages which are created and accessed using a simple **Uniform Resource Locator** (**URL**) is known as web address.
- Any website can be categorized as per the requirement. Such categories like,
  - Educational,
  - Social networking site,
  - Informational site,
  - Organization site,
  - Search engine, etc.

❖ **What is webpage ?**

- A webpage is a part of a website; it means a website contains different web pages.
  - **Static Web Page**
  - **Dynamic Web Page**

✓ **Static Web Page**

- Static web page are the html document which works only with client. Static web pages are also known as flat page.
- To update contents of static web page we must edit coding for web page.
- Any static web page consists of only HTML, CSS and JavaScript.
- All static web page only interaction with client and web browser.

✓ **Dynamic Webpage**

- **A dynamic web-page contains three things:**

1. **Client:** Which is useful and the necessary part to send the request to get the required information.
2. **Server:** Whose work is to process the request sent by the client.
3. **Database:** Where there is the permanent storage of all the information of the client.

❖ **Client Side Scripting language:**

- Client side scripting is the language which can be understood only by the client.
- Generally, the client is used to send the request.
- Now these request can be sent by the browser, so one can say that the browser acts as a Client and the Client can understand HTML only.
- Examples of Client side scripting are HTML, JavaScript, VBScript, etc.

➢ **Advantages and Disadvantages of Client Side Scripting language.**

Advantages	Disadvantages
● Validations can be provided, so that the page is no need to refreshed. ● The web-pages developed are static pages, so they cannot store the information into the database.	● The coding of HTML cannot be made hidden to the user. ● One cannot access the files and directories using HTML. ● Interaction with database and other user is not possible with client side scripting language.

❖ **Server Side Scripting Language:**

- Server side scripting, is the language that can be understood by the server only.
- The work of the server is to process the request which is sent by the client.
- After the completion of the process servers gives back the response to the client.
- Server can understand both the client side language as well as its own language.

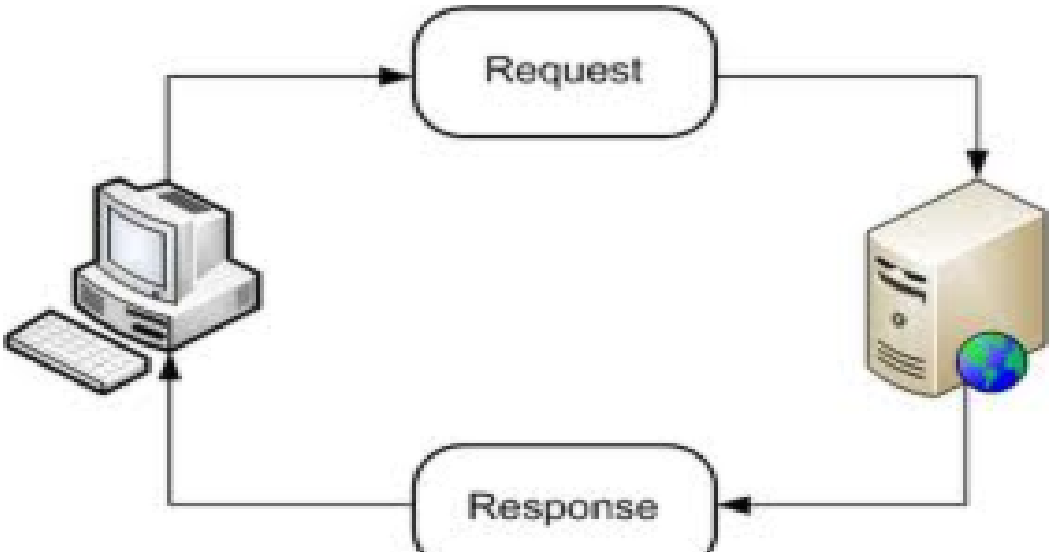
➢ **Advantages and Disadvantages of Server Side Scripting language.**

Advantages	Disadvantages
● Server side code is executed before the HTML is sent to the browser and the code is hidden. ● Through server side scripting the files and directories can be accessed on the local machine. ● Server side code is browser independent. And also the dynamic web-pages are created, so that it can deal with database.	● The server refreshes the page whenever the client sends any request. ● If the validations are given using server side scripting then the whole page is again refreshed. ● Sometimes the heavy code may slow down the web-sites.

❖ **Client/Server Architecture:**

- The Client are all the different kind of browsers like Internet Explorer, Mozilla Firefox, Opera, Netscape Navigator, etc.
- The responsibility of Client is to send the request to the server and display the designing part.
- The Server's work is to process the request given by the client and give the response back to the client.

➢ **Client Server Architecture Work Like below :**



❖ **URL Types:**

- **Absolute Path:**
  - The full path is known as the absolute path.

    For example,

    http://www.computerhope.com/**index.html**
- **Relative Path:**
  - An indirect path is known as relative path.It uses the reference of the current directory where we are working.

    For example,

    "index.html"

❖ **What is Protocol ?**

- When a computer communicates with each other, there are set of rules and instruction that each computer follows.
- Such set of set of Rules and instruction for communication is known as "Protocol".
- There are various kinds of protocol like TCP/IP, FTP, SMTP, etc..

➢ **HTTP Protocol.**

- HTTP means **Hyper Text Transfer Protocol** which is also known as **Request/Response protocol**.
- Protocol is a one kind of language for understand the request.
- Generally HTTP is used Client/Server Architecture.
- It is a language through which the client can send the request and the server can process it.

➢ **HTTPS Protocol:**

- https stands for HTTP Secure.
- Its syntax is similar to that of the http://
- This system was designed by **Netscape Communications Corporation** to provide World Wide Web with security.
- Generally HTTPS are used in websites for payment transactions

❖ FTP (File Transfer Protocol):

- FTP is referred as “File Transfer Protocol”. It is network protocol used to transfer file from one host to another host over network like internet.
- FTP is based on client server architecture where client and server both are use individual control and data connection.
- Every website create has FTP creation to upload all the files created on local machine to make website live. To connect to FTP use as below: **ftp://www.websitename.com .**

➢ Advantages and Disadvantages of FTP.

Advantages	Disadvantages
● FTP is the fast and efficient way of transferring bulk of data across the internet. ● FTP has automatic backup. ● FTP give you control over transfer.	● FTP was not designed to be a secure protocol. ● FTP causes the following attacks during the transfer of data. ○ Bounce attacks ○ Spoof attacks ○ Brute attacks ○ User name protection.

❖ ISP (Internet Service Provider):

- ISP refers to that provides internet services.
- ISP support one or more forms of internet access.
- There are various kind of ISP based on services provided as below :
  1. Home ISP service.
  2. Dial-up Service
  3. DSL Service

4. Cable Service

❖ **Web Hosting:**

- A web hosting service is a kind of Internet hosting service which allows organizations, companies and individuals to make their website live via the World Wide Web.
- Web hosting companies provide some space on owned or server to their clients.
- The server for web hosting is referred to as hardware (Computer) with a very large space and configuration.
- A customer may also choose hosting platform. Like PHP, Perl , Asp etc...
- There are different types of Hosting server of which are listed below:
    1. Free Web Hosting
    2. Shared Web Hosting
    3. Reseller Web Hosting
    4. Virtual Dedicated Server
    5. Dedicated Hosting Service

❖ **Virtual Host:**

- The term Virtual Hosting refers to running more than one web site on a single machine.
- There are generally **two** types of virtual host configuration :

[1] IP-Based Virtual Host :

For this type of configuration where more than one website reffering the server that runs with different IP-Address, the server should have respectively different IP-Address configuration

[2] Name-Based Virtual Host :

In this configuration, when Apache web server receives a requests, it looks for the hostname in the HTTP header, and depending on the hostname, it serves different website.

❖ **Multi-Homing:**

- Multi-Homing describe a computer host that has multiple IP address to connected networks.
- A multihomed host is physically connected to multiple data link that can be on same or different network.
- Multihoming means that redundant local area networks(LANs) can be used to support local access.

❖ **Distributed Server:**

- Large networks have multiple server. The server are often distributed around the network on server on every subnet.
- The client server model is a distributed application structure in computing.
- The distributed server work like Peer-to-Peer (P2P) model.

❖ **Document root:**

- The Document root informs where all PHP script runs from. Many pages scripted using PHP seem to assume that the script is running under Apache.
- Apache provides an environment variable called DOCUMENT_ROOT while IIS does not.
- This variable tells the code about where the web pages are delivered like :

C:\xampp\htdocs

❖ **<u>History of PHP:</u>**

Versions	Released Date	Description
1	1995	It was the first version which is referred to as Personal Home Page introduced by **Rasmus Ledorf.**
2	1997	The newer version of PHP was used over by **50,000** websites for creating dynamic web pages.
3	1998	Then after two scientists **Zeev Suraski** and **Andi**

		Gutmans introduced some extra features into the older version. So that it become faster and simple tool for creating web-sites.
4	2000	When PHP's 4th version was released, the concept of **Zend Engine** was introduced, Super **global variables** like **$_GET**, $_**POST**, $_**SESSION** were introduced, more security was provided.
5	2004	The newer version introduced the **Zend Engine II** with the object oriented features, enabled the filter extension, many bugs were fixed.

❖ **Why PHP?**

- There are various reasons which are listed…
  1. Database Connectivity
  2. Free Ware
  3. Easy to Use
  4. Speed
  5. Compatible with web-server
  6. Cross platform
  7. HTML Support
  8. Stability

➢ **Database Connectivity**

- PHP supports many databases Like, MySQL, Informix, Oracle, Sybase, Solid, PostgreSQL, Generic ODBC, etc.

➢ **Free Ware**

- PHP and MYSQL are freeware which means that they can be easily downloaded without any cost.
- It is an open source software. So no charge is to be paid for it.
- PHP is FREE to download from the official PHP resource: www.php.net

➢ **Easy to Use**

- Compared to other languages it is easy to use as the coding of PHP are similar to that of C language and it is not hard to learn.

- **Speed**
  - Compared to other heavy software like, .Net & Java, which are used to develop websites the speed required to open the webpage create in PHP is much faster.
- **Compatible with web-server**
  - PHP code is processed by different WebServer.
  - Generally, there are three types of Web Server in use, they are
    - Apache,
    - IIS (Internet Information Services) and
    - Netscape Enterprise Service.
    - PHP can execute on any of the above WebServer, but Apache is the webserver which is used widely with PHP
- **Cross platform**
  - CrossPlatform interoperability means operating system independent and machine independent.
  - PHP can work on any Operating System starting from low level like UNIX to high level like Windows-8.
- **HTML Support**
  - PHP is a scripting language and we can also use HTML tags in side of PHP file.
  - PHP uses various controls from html.
- **Stability**
  - Compared to other language like JSP where the server is required to be rebooted each time when any modification is made in the web-page.
  - For PHP there is no issue of rebooting the server.
  - For PHP, the stability comes by two ways:
  - The WebServer is not required to be rebooted again and again.

- If the subversion of PHP and MYSQL are changed then also they are compatible.

➢ **What is a PHP File?**
- PHP files may contain text, HTML tags and scripts
- PHP files are returned to the browser as plain HTML page.
- PHP files have a file extensions like,
- ".php“ (Mostly used with php file)
- ".php3" or
- ".phtml"

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

❖ **PHP + MySQL**
- PHP combined with MySQL are cross-platform (means that we can develop web-pages using Windows and serve on a Unix or any other platform.)

➢ **How to execute PHP Codes…**
- To Execute PHP Codes…
  - Start XAMPP server
  - Open Internet Explorer or any other web-browser.
  - Type the URL as http://localhost
  - Now Press ENTER & it will display the index page…
  - Type Name of your PHP file after ***localhost*** and execute it…

❖ **Variables in PHP:**
- Computer has memory cells same as human mind to remembering the values.

- Each memory cells in the computer has unique address which will be in binary format.
- It is not possible to remember the address of each memory cell so that cell is introduced with a name which is called “variable”.
- Variable is a storage area where the values can be stored. These values may be a fixed value or the value is inputted by the user.

➢ **How to define Variables in PHP?**

- Syntax : $variable_name=value
- Purpose : To define variable in PHP we can use above syntax
- **Example : <?php $a=10; $b=20; $c=$a+$b; echo $c; ?>**

➢ **Rules for variable naming:**

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

➢ **Data Types**

- The variable which are defined in PHP has some values.
- In PHP the data type is not specified while declaring a variable.
- But still there may be situation when it is required to identify the data type.
- **PHP supports the following data types:**
  - **String**
  - **Integer**
  - **Float (floating point numbers - also called double)**
  - **Boolean**
  - **NULL**

➢ **Integer Types**

- An integer data type is a non-decimal number between -2,147,483,648 and 2,147,483,647.

- Rules for integers:
  - An integer must have at least one digit
  - An integer must not have a decimal point
  - An integer can be either positive or negative
  - Integers can be specified in: decimal (base 10), hexadecimal (base 16), octal (base 8), or binary (base 2) notation.
- **Example**: **<?php $x = 5985;var_dump($x); ?>**

➢ **Floating Point Types (Double):**

- A float (floating point number) is a number with a decimal point or a number in exponential form.
- **Example: <?php $percent=62.66666; var_dump($percent);?>**

➢ **String Type:**

- A string is a sequence of characters, like **"Hello world!"**.
- A string can be any text inside quotes. You can use single or double quotes.
- **Example: <?php $name="Monarch"; echo "<b>",$name; ?>**

➢ **Logical Data Type Boolean:**

- This type of data type has 2 kind of values that is either true or false.
- The memory size of such variable is 1 byte.
- **Example:** <?php $flag=false; echo $flag; ?>

❖ **PHP NULL Value:**

- Null is a special data type which can have only one value: NULL.
- A variable of data type NULL is a variable that has no value assigned to it.
- **Example**: <?php $x = null; var_dump($x); ?>

➢ **How to Know (get) the data type of Variable:**

- **gettype() function:**

  Syntax : gettype(variable name / value);

  Purpose : To know the data type of any variable we can use get type

  Example :

```
<?php
    $a=25000;
    echo gettype($a);
```

?>

- **How to set the data type of Variable:**
  - **settype() function:**

    Syntax : settype(variable name, data type);

    Purpose : To set the data type of any variable we can use get type

    - **Example:**

```
<?php
$a=25000;
echo gettype($a);
settype($a,'double');
echo gettype($a);
?>
```

- **COMMON USAGE OF PHP:**
  - 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.
  - You add, delete, modify elements within your database through PHP.Access cookies variables and set cookies.
  - you can restrict users to access some pages of your website.
  - It can encrypt data.

- **CHARACTERISTICS OF PHP:**

Five important characteristics make PHP's practical nature possible −

1. Simplicity
2. Efficiency
3. Security
4. Flexibility
5. Familiarity

1. Simplicity:

PHP is particularly famous for its simplicity. It is organized and easy to learn. Even beginners won't face any hard time learning and using PHP. It is a very well-organized programming language, and it comes with a lot of pre-defined functions, which makes the task of the programmer easy. There is no need to include libraries in PHP like C. With a lot of pre-defined functions, PHP is easy to optimize as well.

2. **Efficiency:**

   PHP is a versatile, reliable, and efficient programming language. The memory management of PHP is very efficient. Great session management, eliminating unnecessary memory allocation, are some of the features that make PHP efficient.

3. **Security:**

   PHP has many pre-defined functions for data encryption. Users can also use third-party applications for security. Security and flexibility are often contrasting features, but PHP somehow manages to offer them both, and that's great. PHP is designed specifically to be a more secure language for writing CGI (Computer-generated Imagery) programs. Security algorithms such as Sha1 (secure Hash algorithm 1) and MD5(Message digest 5) are used to encrypt the strings in PHP. Filter_var and strip_tags functions help to keep the environment more secure and safe for users.

4. **Flexibility:**

   PHP scripts can run on any device- mobile, tablet, or PC. It is very compatible with various databases. It can be easily embedded and integrated into HTML, XML, and JavaScript. Likewise, it is also compatible with almost all servers used today like Apache, IIS, etc.

5. **Familiarity:**

   If you are in programming background then you can easily understand the PHP syntax. And you can write PHP script because of most of PHP syntax inherited from other languages like C or Pascal.

- **PHP CONFIGURATION IN IIS & APACHE WEB SERVER**
  - **PHP configuration in IIS:**

1. On the Start page, type Control Panel, and then click the Control Panel icon in the search results.
2. In Control Panel, click Programs, and then click Turn Windows features on or off.
3. In the Windows Features dialog box, click Internet Information Services, note the preselected features that are installed by default, and then select CGI. This selection also installs Fast CGI, which is recommended for PHP applications.
4. type the following into a web browser:

   http://localhost

- **Understanding of PHP.INI file**
  - we will learn about the purpose of the php.ini file. At the time of PHP installation, php.ini was a special file provided as a default configuration file.
  - It's a very essential configuration file that controls what a user can or cannot do with the website.
  - Each time PHP is initialized, the php.ini file is read by the system.
  - Sometimes you need to change the behaviour of PHP at runtime, then this configuration file is to use.
  - The php.ini file is the default configuration file for running applications that require PHP. It is used to control variables such as upload sizes, file timeouts, and resource limits.
  - php.ini file is the configuration file. It is always checked when the server gets started or HTTP is restarted in the module and it configures the website to know what a user can do or can't do with a website.
- **Understanding of PHP .htaccess file**
  1. The .htaccess (Hypertext Access) file is an Apache distributed server configuration file.
  2. You can use the .htaccess file to set server configurations for a specific directory.

3. .htaccess file is created in order to enable extra features for that subdirectory.
4. You can use the .htaccess file to modify various configurations and thus make changes to your website. These changes include authorization, error handling, redirects for specific URLs, user permissions, etc.

❖ **Common uses of the .htaccess file:**

1. Change the default start page
2. Redirect from HTTP to HTTPS.
3. Block a specific IP or range of IPs.
4. Customize your Error Page.
5. Denying access from a specific domain.
6. Authenticated folder
7. Block or allow ranges of IP addresses .
8. 301 Permanent Redirect.
9. WWW to non-WWW and non-WWW to WWW.

❖ **Operators in PHP:**

➢ **What is Operator?**

- The symbols which are used to perform mathematical or logical operation those symbols are known as operator.
- Like,
- +, -, *, /, &&, ||, !=, <, >, etc**.**

➢ **What is Operand?**

- Operand means the variable or value which is used before and after the operator.
- **Like,**
  - A+B (Here, A and B are operand and + is operator)

➢ **Available Operators in PHP:**

- There are different types of operators in PHP as given.
  1. Arithmetic Operator
  2. Relational Operator
  3. Logical Operator

4. Assignment Operator
5. String Operator

1) Arithmetic Operators in PHP:

Arithmetic operators are used when perform any mathematical operation.

Operator	Description
+	Adds two operands
-	Subtracts second operand from the first
/	Divide numerator by denumerator
*	Multiply both operands
%	ModulusOperator and remainder of after an integer division

2) Logical Operators in PHP:

When more then two conditions are to be checked at the same time then logical operators can be helpful

Operator	Logic	Description
&&	AND	Returns TRUE If all the conditions are TRUE
||	OR	Returns TRUE If any of the condition is TURE
!	NOT	Returns TRUE If Condition is not TRUE

3) Relational Operators in PHP:

When variables are related to some value at that the relational operator is used.

Operator	Description
==	This operator is used to compare the two variables or the values to check the equality. It checks only the value and not the data type.
===	This operator is used for comparing the equality of the variables or the values to check the equality. It will check value and data types also.
!=	This operator is for not equality. It will return true if the values are not equal. It will checks

	the only values not data type.
!==	**This operator is also for not equality. It will compare two values and data types too.**
<	**Used for less than**
>	**Used for greater than**
>=	**Used for greater than equal to**
<=	**Used for less than equal to**

4) Assignment Operators in PHP:

There are different assignment operators as given.

Operator	Description
=	**Assigning the value $a=5**
***=**	**$a*=10 is similar to $a=$a*10**
/=	**$a+=10 is similar to $a=$a/10**
+=	**$a+=10 is similar to $a=$a+10**
-=	**$a+=10 is similar to $a=$a-10**
%=	**$a+=10 is similar to $a=$a%10**
.=	**a.=$a is similar to $a=$a.$a used for concatenation of the variable**

5) String Operators in PHP:

- The operator . (dot) is used for the concatenation of the two strings and so . (dot) is also known as String Operator.
- **Writing Output Command:**

  Syntax : void echo (string $arg1 [, string $.])

  Purpose : To writing output to a web page.

  Example :

  ```
  <?php  $a="Hello";  $b=$a."World"; echo $b;
  ```

  Output: Hello World

❖ **Conditional Structure:**

- Like most programming languages, PHP also allows you to write code that perform different actions based on the results of a logical or comparative test conditions at run time. This means, you can create test conditions in the form of expressions that evaluates to either true or false and based on these results you can perform certain actions.
  1. The if statement
  2. The if...else statement
  3. The if...elseif....else statement
  4. The switch...case statement

1) The if Statement:

- The if statement is used to execute a block of code only if the specified condition evaluates to true.
- This is the simplest PHP's conditional statements and can be **written like:**

  Syntax: if(condition) { // Code to be executed }

  Example: <?php $d = 10; if($d == 10) { echo " same "; }

2) The if...else Statement:

- The if...else statement allows you to execute one block of code if the specified condition is evaluates to true and another block of code if it is evaluates to false.
- **It can be written, like this:**

```
Syntax:
if(condition){
   // Code to be executed if condition is true
} else {
   // Code to be executed if condition is false
}
```

a. **Example:**

```
<?php
$d = 10;
if($d == 10) {
        echo "Have a nice Day!";
    } else {
          echo “Enter Right Value”;
    }
   ?>
```

3) The if...elseif...else Statement (Ledder IF):

Example:

```
<?php
$d = 10;
if($d == 10) {
echo "Have a nice Day!";
} elseif( $d==11 ) {
   echo “Enter greate day”;
} else {
        echo “Enter Right value”;
    }
    ?>
```

4) The Ternary Operator:

- The ternary operator provides a shorthand way of writing the if...else statements. The ternary operator is represented by the question mark (?) symbol and it takes three operands: a condition to check, a result for true, and a result for false.
- To understand how this operator works, consider the following examples:

```
<?php
echo ($age < 18) ? 'Child' : 'Adult’ ;
?>
```

5) Switch Case:

- The switch statement causes a particular group of statements to be chosen from several available groups.
- The selection is tasted upon the current value of an expression that is included within the switch statement.
- Switch case is a built in multi-way decision statement.
- It tests value of given variable (or Text Expression) against a list of case values & block of statements associated with it is executed when a match is found.
- **Switch(expression):**

a. Here expression result is an integer value or char type**.**

Syntax :

```
switch(TxtExpression)
{
    Case TxtExp1:
        Statements;
    Break;
    Case TxtExp2:
        Statements;
    Break;
    Case TxtExpN:
        Statements;
    Break;
    Default:
        Statements;
}
```

b. Purpose:

- Switch case statement will provide us easy features of working it will provide selection based switching in group of statements. After completion of group execution we must put a break; statement to stop execution in each group. We can use either integer number or character for switching from list of group.

❖ **Looping Structure:**

- Loops are used when we want to execute a part for a block of statements for specified number of times or depending on some specified condition.
- The main thing about any of the loop is that it executed for the number of times based on the logical expression (condition) that is specified to the particular loop.
- When loop checks for the condition to execute at that time there are four types of loops available.
  - **for loop**
  - **while loop**
  - **do-while loop**
  - **foreach loop**

1) For Loop:

- **Syntax :**

```
for(initialization; condition; increment or decrement)
{
        statements;
}
```

- **Purpose :** To provide a looping while condition will be true we can use for loop.
- **initialization :** Initialize the value of variable.
- **test Condition :** Test condition at here.
- **Increment/decrement:** To increase or decrease the value.

Example :

```
<?php
for($a=1;$a<=10;$a++){
   echo "<br>".$a;
}
?>
```

2) While… Loop:

Syntax :

```
while(test condition)  {
   statements;
   }
```

Purpose : To provide a looping till condition will be true we can use while loop.While loop will check the condition first.If the condition will be true then it will execute the statements given in the loop, after once complete the loop, it will again and again check the condition if condition will false then it will auto exit for the loop.

Example:

```
<?php
$a=10;
while($a>=1) {
   echo "<br>".$a;
   $a=$a-1;
}
?>
```

3) Do…while( )

Syntax:

```
do
   {
         statements;
   }while(test condition);
```

Purpose:

To provide a looping till condition will be true we can use do…while loop.Do…while loop will check the condition after execution of statements provided in to loop.If the condition will be true then it will again execute the statements given in the loop, if condition will false then it will auto exit for the loop.

Example:

```
<?php
$num = 2;
do {
   $num += 2;
   echo $num, "<br>";
} while ($num < 12);
?>
```

4) foreach Loop:

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

For every loop iteration, the value of the current array element is assigned to $value and the array pointer is moved by one, until it reaches the last array element.

Syntax:

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

Example:

```
<?php
$colors = array("red", "green", "blue", "yellow");
foreach ($colors as $value) {
   echo $value."<br>";
}
?>
```

- **Nesting of Loop:**

When one looping statement is within another looping statement is called as nesting of loop. The nesting may continue up to any desired levels.

- **Break Statement:**

Syntax : break;

Purpose : The break command will break the loop.

Example :

```
<?php
        for($i=1;$i<=10;$i++)
        {
        if($i==5)
                break;
            echo "<br>Num. is ".$i;
        }
    ?>
```

- **Continue Statements:**
  - **Syntax** : continue;
  - **Purpose** : The continue command will break the current loop and continue with the next loop.
  - **Example** :

```
<?php
for($i=1;$i<=10;$i++)
{
    if($i==5)
        continue;
    echo "<br>Num. is ".$i;
}
?>
```

- **PHP Arrays:**
- **What is an Array?**
  - An array is a data structure that stores one or more similar type of values in a single value. For example if you want to store 100 numbers then instead of defining 100 variables its easy to define an array of 100 length.
  - There are three different kind of arrays and each array value is accessed using an ID which is called array index.
  - **In PHP, there are three types of arrays:**
    1. **Indexed arrays** - Arrays with a numeric index
    2. **Associative arrays** - Arrays with named keys

3. **Multidimensional arrays** - Arrays containing one or more arrays

i. **Indexed Array:**
   - PHP index is represented by number which starts from 0. We can store number, string and object in the PHP array. All PHP array elements are assigned to an index number by default.
   - **Example:**

```
<?php
$season=array("summer","winter","spring","autumn");
echo "Season are: $season[0], $season[1], $season[2] and $season[3]";
?>
```

ii. **Associative Array:**
   - We can associate name with each array elements in PHP using => symbol.
   - **Example:**

```
<?php
$info=array("name"=>"Raj","email"=>"raj@gmail.com","contact"=>"
9898989898");
echo "Name: ".$info["name"]."<br/>";
echo "Email: ".$info["email"]."<br/>";
echo "Contact: ".$info["contact"]."<br/>";
?>
```

iii. **Two or Multidimensional Array with indexed:**
   - A multi or tow dimensional arrays consists of array inside that another array.
   - This type of array can be considered as a table having rows and columns.
   - A table consists of many rows and each row consists of many columns.
   - **Syntax :**

```
$variable=array(array(value1, value2..),      array(value1, value2..));
```

   - **Example:**

```
<?php
        $name=Array(array("1","Ravi"), array("2","Raj"));
        echo "<br>".$name[0][0];
        echo "<br>".$name[0][1];
        echo "<br>".$name[1][0];
        echo "<br>".$name[1][1];
?>
```

- **Two or Multi Dimensional Array with KEY:**
  - **Syntax:**
    $variable=array(key1=>array(childKey1=>value1,childKey2=>value2..),key2=>array(childKey1=>vlaue1,childKey2=>value2..));
  - **Purpose**: To define a multi dimensional array with key.
  - **Example:**

```
<?php
$student=array("rno1"=>array("no"=>"1","name"=>"Mit),"rno2"=>array("no"=>"2","name"=>"Harsh"),"rno3"=>array("no"=>"3","name"=>"Monarch"));
   echo "<br>".$student["rno1"]["no"];
   echo "<br>".$student["rno1"]["name"];
   echo "<br>".$student["rno2"]["no"];
   echo "<br>".$student["rno2"]["name"];
   echo "<br>".$student["rno3"]["no"];
   echo "<br>".$student["rno3"]["name"];
?>
```

- **Foreach with Single Dimension**
  - **Example**:

```
<?php
        $array=array("Banana","Mango","Graps");
        foreach($array as $fruit)
        {
            echo "<br>".$ fruit;
        }
```

```
?>
```

- **Foreach with Multi Dimension:**
  - **Example:**

```
<?php
$student=array(
"rno1"=>array("no"=>"1","name"=>“Mit"),
"rno2"=>array("no"=>"2","name"=>"Harsh"),
"rno3"=>array("no"=>"3","name"=>"Monarch")
);
   foreach($student as $data)
      {     echo "<br>".$data['no'];
            echo "<br>".$data['name'];
      }
?>
```

- **User Defined Functions**:
  - The library functions are those functions whose definitions is inbuilt in PHP. There is no need to write the definition for library functions.
  - Some times the programmer is in need to use some codes many time. It will add the need of UDF in programming filed.
  - We can write definitions for required functions which are known as **User Defined Functions.**

    b. **Advantages of UDF are as below:**
    - It facilitates top-down modular programming.
    - The length of the source code of program is reduced.
    - A function may be used at several points instead of writing the same code again and again.
  - **Parts of UDF:**
    - In PHP we must define a function using keyword “function”.
    - The name of the function.
    - The optional argument that contains the variable.
    - The statements which are to be written inside the function

- The keyword return with the variable if the function returns any value.

- **UDF Syntax:**

```
function function_name(arguments list)
{
   Statements…
   [return ]
}
```

c. **Types of UDFs**:

1. No argument and no return value
2. With argument no return value
3. No argument with return value
4. With argument and with return value

1. No argument and no return value:

- This type of function does not take any argument and does not have any return value.
- **Example**:

```
<?php
        function welcome()
        {
            echo “WelCome to My Page”;
        }
        welcome();
?>
```

1. With argument no return value:

- This type of function takes some argument and does not have any return value.
- **Example:**

```
<?php
   function welcome($nm)
   {
        echo $nm, “ WelCome to My Page”;
```

```
    }
    welcome("Mr. Ravi");
?>
```

2. No argument and with return value:

- This type of function takes no argument and return some values.
- **Example:**

```
<?php
   function sum()
   {     $tot=10+6;
         return $tot;
   }
   $a=sum();
   echo $a;
?>
```

3. With argument and with return value:

- This type of function takes some argument and return some values.
- Remember that before this type of function we must use variable of output command.
- **Example**:

```
<?php
   function sum($v1,$v2)
   {     $tot=$v1+$v2;
         return $tot;
   }
   $a=sum(20,30);
   echo $a;
?>
```

➢ **Default argument value functions:**

- When programmer wants to provide some default argument to a function, at that time in PHP function with default value is used.

- When the function is called if no argument is passed then it will return the default value which is set in function definition.
- If any argument will passed then it will overwrite the value which is passed.
- **Example** :

```
<?php
    function name($str="Welcome to World")
    {
        echo $str;
    }
    name("Wel Come To EARTH");
    echo "<br>";
    name();
?>
```

➢ **Variable function:**

- PHP supports the variable function.
- Variable has the value of the function name and if the function is called using that variable then it will call the function having the value same as function name.
- We can define variable as function name.

  - $funname=“sum”;
  - $funname(a,b);

- **Example** :

```
<?php
    function witharg($value)
    {       echo $value;        }
    function withoutarg()
    {       echo "Welcome";   }
```

```
$value="witharg";
$value("Hello World");
$value1="withoutarg";
$value1();
?>
```

- **Variable Scope:**
  - The scope of variable means lifetime of variable.
  - The scope of variable is use when functions are used.
  - Scope of variables are as given:
    1. Local Variable
    2. Global Variable
    3. Static Variable

1. Local Variable

- The variables which are used only within the function is known as ***local variable***.
- The scope of the variable is till the function ending only.
- **Example**:

```
<?php
        function A()
        {   $a="Monarch";
        }
        function B()
        {   echo $a;
        }
        A();
        B();
?>
```

- It will display an error variable is not defined…

2. Global Variable:

- Global variables are the variable which are defined with **global** keyword.
- This kind of variable can be used in different functions too…
- The function defined with **global** keyword can be used in full of the webpage.
- To define global variable the keyword global is to be used.
- **Syntax**:
  global $a;
- **Example:**

```
<?php
    function SetName()
    {       global $a;
            $a="Raju";
    }
    function GetName()
    {       echo $GLOBALS['a']      ;
    }
    SetName();
    GetName();
?>
```

3. Static Variable:

- In static variable, the value of the variable will not change even thought the page is refreshed.
- The static variable holds the value which was set and it keeps that value till the page is closed.
- The static variable declared with **static** keyword.
  - Syntax:
    Static $a=1;
- **Example**:

```
<?php
    function increment()
```

```
{       static $a=0;
        $a++;
        echo "<br>",$a;
}
increment();
increment();
increment();
?>
```

- **Variable Length Argument Functions:**

1. func_num_args
2. func_get_arg
3. func_get_args

1. func_num_args():

- This function will count the number of the arguments which are passed to the function when called.
- The return type will be the integer.
- **Example:**

```
<?php
   function a()
   {
        $argnum=func_num_args();
        echo $argnum;
   }
   a(40,50,60);
?>
```

2. func_get_arg():

- To use particular argument we can use this function.
- It accepts and integer offset.
- If the offset is out of range then it will generate the warning.
- The offset starts from zero.
- **Example:**

```
<?php
```

```
function a()
{
        $argnum=func_get_arg(2);
        echo $argnum;
}
a(40,50,60);
?>
```

3. func_get_args():

- This function will display the array of the arguments which are obtained when they are passed during the call of function**.**
- **Syntax:**
  Array func_get_args();
- **Example:**

```
<?php
   function a()
   {    $num=func_num_args();
        $array=func_get_args();
        for($i=0;$i<$num;$i++)
        {    echo $array[$i];
        }
   } a(40,50,60);
?>
```

- **<u>Built in Functions:</u>**

1. Variable Functions
2. String Function
3. Math Function
4. Date Function
5. Array Function
6. Miscellaneous Function
7. File handling Function

➢ **Variable Functions:**

1) **gettype()**
2) **settype()**
3) **isset()**
4) **unset()**

1) **gettype( ) function:**
   - **Syntax** : gettype(variable name / value);
   - **Purpose** : To know the data type of any variable we can use get type
   - **Example:**

```
<?php
        $a=25000;
        $b=25.100;
        echo gettype($a);
        echo gettype($b);
    ?>
```

2) **settype( ) function:**
   - **Syntax** : settype(variable name, data type);
   - **Purpose:** To set the data type of any variable we can use get type

   Example:

```
<?php
    $a=25000;
    echo gettype($a);
    settype($a,'double');
    echo gettype($a);
?>
```

3) **isset( ):**
   - **Syntax** : Boolean isset(MixedVariableName);
   - **Purpose:** To check that any value is stored in variable or not.

   Example:

```
<?php
```

```
$value="";
if(isset($value))
    echo "Has some value";
else
    echo "Has No Value";
?>
```

4) unset( ):

- **Syntax : ** unset(variableName);
- **Purpose:** To remove variable from memory we can use unset.
- **Example:**

```
<?php  $a=10;  unset($a);  echo $a;  ?>
```

➢ **String Functions:**

String functions are used to perform some operations with the string.

String Functions		
Chr	**Ord**	**strtolower**
strtoupper	**strlen**	**ltrim**
rtrim	**trim**	**substr**
strcmp	**strcasecmp**	**strpos**
strrpos	**strstr**	**stristr**
str_replace	**strrev**	**echo**
print		

1) Chr():

Syntax : chr(ascii code in integer)

Purpose : This function is used to convert the ascii code values into character values.

Example : <?php echo chr(65); ?>

Output : A

2) ord():

Syntax : ord(character code in string)

Purpose : This function is used to convert the character code values into ascii value.

Example : <?php echo ord(“A”); ?>

Output : **65**

3) strtolower():

Syntax : string strtolower(string variable)

Purpose : This function returns all the characters of the string into lower case.

Example : <?php echo strtolower (“Hello”); ?>

Output : **hello**

4) strtoupper():

Syntax : string strtoupper(string variable)

Purpose : This function returns all the characters of the string into Upper case.

Example : <?php echo strtoupper(“hello”); ?>

Output : **HELLO**

5) ucfirst

Syntax : string ucfirst(string variable)

Purpose : This function returns first character of the string into Upper case.

Example : <?php echo ucfirst(“hello”); ?>

Output : **Hello**

6) ucwords

Syntax : string ucwords(string variable)

Purpose : This function returns first character of all the words into Upper case.

Example : <?php echo ucwords(“good morning”); ?>

Output : **Good Morning**

7) Strlen

Syntax : Integer strlen(string variable)

Purpose : This function returns total number of characters with space present in the string.

Example : <?php echo strlen("monarch computer"); ?>

Output : **16**

8) Ltrim

Syntax : string ltrim(string variable [,String Char])

Purpose : This function also known as left trim. This function is used to remove the blank space or specified character from beginning (left side) of the string..

Example : <?php echo strlen(ltrim(" monarch"));
echo ltrim("***monarch***","*"); ?>

Output : 7 monarch***

9) Rtrim

Syntax : string rtrim(string variable[,String Char])

Purpose : This function also known as right trim. This function is used to remove the blank space or specified character from ending (right side) of the string..

Example : <?php echo strlen(rtrim(" monarch"));
echo rtrim("***monarch***","*"); ?>

10) Trim

Syntax : string trim(string variable)

Purpose : This function is used to remove the blank space or specified character from beginning and ending (left & right side) of the string.

Example :<?php echo strlen(trim(" monarch ")); echo trim("***monarch***","*"); ?> **Output** : 7 monarch

11) substr

Syntax : string substr(string variable, integer position, integer length)

Purpose : This function is used to retrieved specified part of the string.

Note : **Location index starts from 0.**

Example :<?php echo substr("Computer",0,3); ?>

Output : Com

12) strcmp

Syntax : Integer strcmp(String1, String2)

Purpose : This function is used to compare two string with each other.

Note : It is case sensitive…

If string1 is greater than string2 then it will return>0.

If string2 is greater then string1 then it will return<0.

If the comparison of both string are equal then it will return 0.

13) Strcasecmp

Syntax : Int strcasecmp(String1, String2)

Purpose : This function is used to compare two string with each other.

Note : It is NOT case sensitive…

If string1 is greater than string2 then it will return>0.

If string2 is greater then string1 then it will return<0.

If the comparison of both string are equal then it will return 0.

14) Strncasecmp

Syntax : Int strncasecmp(String1, String2,int length)

Purpose : This function is used to compare two string with each other using fixed length given.

Note : It is NOT case sensitive…

If string1 is greater than string2 then it will return>0.

If string2 is greater then string1 then it will return<0.

Int length is length to compare two string…

If the comparison of both string by given length are equal then it will return 0.

15) Strpos

Syntax : Int strpos(String, Character)

Purpose : This function is used to find the position of first occurrence of the character in the string.

Note : This function is case sensitive.

Example: echo strpos(“NITIN”,”I”);

Output : 1

16) Strrpos:

Syntax : Int strrpos(String, Character)

Purpose : This function is used to find the position of Last occurrence of the character in the string.

Note : This function is case sensitive.

Example: echo strrpos(“NITIN”,”I”);

Output: 3

17) Strstr

Syntax : String strstr(String variable, character)

Purpose : This function is used to get string from specified character to end of string.

Note : This function is case sensitive.

Example : echo strstr(“NITIN”,”I”);

Output : ITIN

18) Strrev

Syntax : String strrev(String variable)

Purpose : This function is used to get string in reverse order.

Example : echo strrev(“Hello”);

Output : olleh

19) str_replace

Syntax : String str_replace(Search String/Array, Character to Replace, Original String)

Purpose: This function is used to find specified character and replace it with given character.

Note : str_replace function is case sensitive.

Example: echo str_replace(“a”,”*”,”Monarch”);

Output : Mon*rch

20) str_ireplace

Syntax : String str_ireplace(Search String/Array,Character to Replace, Original String)

Purpose : This function is used to find specified character and replace it with given character.

Note : str_ireplace function is NOT case sensitive.

Example: echo str_ireplace("A","*","Monarch");

Output : Mon*rc

21) Strrchr

Syntax : strrchr(String, Character to be search)

Purpose : This function is used to retrieve the string from last occurrence of given character.

Note : strrchr function is case sensitive.

Example : echo strrchr("Nana Nani","N");

Output : Nani

22) Strval

Syntax : String strval(Variable of other data type)

Purpose : This function is used to convert any data type value of PHP into string value and will return string value.

Example : <?php $digit=12345; echo strval($digit); ?>

23) Echo

Syntax : echo "String Value", variable list

Purpose : This function is used display any value on the web-page. This function will not return anything and so that return type of this function is void.

Example : <?php $digit=12345; echo "Value Given :",$digit; ?>

24) Print

Syntax : print(argument)

Purpose : This function is used display any value on the web-page. This function will not support multi variables in same statements.

Example :

<?php $a="WelCome"; $b=" Bye"; print $a; print $b; ?>

25) Addslashes :

Syntax : string addslashes(String variable)

Purpose : This function is used to insert the backward slash (\) in front of backslash (\), double quote("), single quote('), and NULL.

Example : <?php echo "monarch's"; echo addslashes("monarch's"); **?>**

Output : monarch's monarch\'s

26) Stripslashes :

Syntax : string stripslashes(String variable)

Purpose : This function is used to remove the backward slash (\) from in front of backslash(\), double quote("), single quote('), and NULL.

Example : echo "monarch\'s"; echo stripslashes("monarch\'s");

Output : monarch \'s monarch's

- **Maths Functions:**

1) abs():

Syntax : mixed abs(mixed number)

Purpose : This function is used to find the absolute value of given number. If value is in integer then the return type is integer and if it is float then the return type is float.

Example: echo abs(-65); echo abs(-42.5);

Output : 65 42.5

2) ceil( )

Syntax : int ceil(float value)

Purpose : This function will return nearest integer larger value for given float value.

Example : echo ceil(5.7); echo ceil(-2.2);

Output : 6 -2

3) floor( )

Syntax : int floor(float value)

Purpose : This function will return nearest integer small value for given float value.

Example : echo floor(5.7);

Output : 5

4) **round( )**

Syntax : float round(float value, int precision)

Purpose : This function will round the floating value as given precision.

Example : echo round(5.7565,2); echo round(2.2222,3);

Output : 5.76 2.222

5) **fmod( )**

Syntax : mixed fmod(value1, value2)

Purpose : This function will act as an operator module which is used to find reminder between two digits.

Example : echo fmod(5.75,2.25); echo fmod(4.5,1.5);

Output : 1.25 0

6) **min( )**

Syntax : mixed min(array or value list)

Purpose: This function will return the minimum value from given list.

Example : echo min(3,55,1,9);

Output : 1

7) **max( )**

Syntax : mixed max(array or value list)

Purpose: This function will return the maximum value from given list.

Example : echo max(3,55,1,9);

Output : 55

8) **pow( )**

Syntax : mixed pow(base value, exponential value)

Purpose : This function is used to find the power of base value.

Example : echo pow(3,2);

Output : 9

9) **sqrt( )**

Syntax : mixed sqrt(mixed value)

Purpose: This function is used to find square root of the given number.

Example : echo sqrt(16);

Output : 4

10) rand( )

Syntax : int rand(int min, int max)

Purpose : This function will generates the random integer value each time the page is refreshed.

Example: echo rand(1,100);

Output : Each time different…

11) bindec( )

Syntax : Mixed bindec(string binary value)

Purpose : This function converts the number from binary to decimal format.

Example: echo bindec(“110110”);

Output : 54

12) octdec( )

Syntax : Mixed octdec(string octal value)

Purpose: This function converts the number from octal to decimal format.

Example: echo octdec(“54”);

Output :54

13) hexdec( )

Syntax : Mixed hexdec(string hexadecimal value)

Purpose : This function converts the number from hexadecimal to decimal format.

Example: echo hexdec(“A7”);

Output : 167

14) decbin( )

Syntax : Mixed decbin(string decimal value)

Purpose : This function converts the number from decimal to binary format.

Example: echo decbin(“54”);

Output : 110110

15) decoct( )

Syntax : Mixed decoct(string decimal value)

Purpose : This function converts the number from decimal to octal format.

Example: echo decoct(“44”);

Output : 54

16) dechex( )

Syntax : Mixed dechex(string decimal value)

Purpose : This function converts the number from decimal to hexadecimal format.

Example: echo dechex(“167”);

Output : a7

- **Date Functions:**
  - To work with date we can use date function
    - date
    - getdate
    - setdate
    - checkdate
    - time
    - mktime

1) date():

Syntax : date date(‘Format’);

Purpose : To return date with specified format we can use Date function

Example : <?php echo date('d-F-Y'); ?>

Output : 28-December-2013

a	am or pm (lowercase)
A	AM or PM (uppercase)
d	Day of month (number with leading zeroes)
D	Day of week (three letters)
e	Timezone identifier
F	Month name
h	Hour (12-hour formatleading zeroes)
H	Hour (24-hour formatleading zeroes)
g	Hour (12-hour formatno leading zeroes)
G	Hour (24-hour formatno leading zeroes)
i	Minutes
j	Day of the month (no leading zeroes)
l	Day of the week (name)
L	Leap year (1 for yes, 0 for no)
m	Month of year (numberleading zeroes)
M	Month of year (three letters)
n	Month of year (numberno leading zeroes)
s	Seconds of hour
S	Ordinal suffix for the day of the month
r	Full date standardized to RFC 822 (http://www.faqs.org/rfcs/rfc822.html)
U	Time stamp
y	Year (two digits)
Y	Year (four digits)
z	Day of year (0-365)
Z	Offset in seconds from GMT

2) getdate():

Syntax : Array getdate('Format');

Purpose : To return date in different format and store it as an array we can use getdate function.

Example :

```
<?php
    $dt=getdate(seconds);
    foreach($dt as $key=>$val)
        echo "$key = $val<br>";
?>
```

Output :

```
seconds = 4
minutes = 38
hours = 19
mday = 28
wday = 6
mon = 12
year = 2013
yday = 361
weekday = Saturday
month = December
```

3) setdate():

Syntax : Void setdate(int year, int mon, int day);

Purpose : To set user defined date we can use setdate function.

Example :

```
<?php
    $date = new DateTime();
    $date->setDate(1979,06,11);
    echo $date->format('D:d-F-Y');
?>
```

Output : Mon : 11-June-1979

4) checkdate():

Syntax : Boolean checkdate(int month, int day, int year)

Purpose : This function will check whether the given date is in proper format then TRUE or not then FALSE.

Example :

```
<?php
    $format=checkdate(06,11,1979);
        if($format==true)
            echo "Given Date is Valid";
        else
            echo "Given Date is NOT Valid";
?>
```

Output : Given Date is Valid

5) time():

Syntax : DateTime time()

Purpose : This function will returns the current time from January 1, 1970 to current time in seconds format.

Example :

```
<?php
    echo time();
?>
```

Output : 1388261763

6) mktime():

Syntax : Date mktime(int hour, int min, int sec, int month, int date, int year)

Purpose : This function will returns time stamp that can be used with date or getdate function.

Example :

```
<?php
   $userdate=mktime(1,50,20,06,11,1979);
   echo date('d-m-Y',$userdate);
?>
```

Output : 11-06-1979

- **Array Handling Functions :**

1) count( ):

Syntax : int count(Array variable)

Purpose **:** This function is used to count number of element of array.

Example **:**

```
<?php
$student=array(1,"Ravi","Lathi");
echo “No of Ele:”.count($student);
?>
```

2) list( ):

Syntax : void list(variable1, variable2…)=ArrayVar

Purpose **:** To assign value of array in individual variable list.

Example **:** $student=array(1,"Ravi","Lathi");

```
list($no,$name,$city)=$student;
echo "<br>No   = ".$no;
echo "<br>Name = ".$name;
```

3) is_array():

Syntax : int is_array(variable name)

Purpose **:** This function is used to check whether the given variable is an array variable or not.

Example:

```
$student=array(1,"Ravi","Lathi");
if(is_array($student))
   echo "It is array variable";
else
   echo "Sorry its not an array variable";
```

4) in_array( ):

Syntax : Boolean in_array(val. to search, array vari.)

Purpose **:** This function is used to search the given value is present in array or not. If value is found then it will return logical true else false.

Example:

```
$student=array(1,"Ravi","Lathi");
if(in_array("Ravi",$student))
    echo "Value is found";
else
    echo "Value is NOT Found";
```

5) current( ):

Syntax : mixed current(array variable)

Purpose **:** This function will return the current item element from selected array variable basically new declared array will return current records as first record.

Example:

```
$student=array("Ravi","Umang");
echo current($student);
```

6) next( ):

Syntax : mixed next(array variable)

Purpose **:** This function will moves the pointer to the next position in array and returns the value of element.

Example:

```
$student=array("Ravi","Umang");
echo current($student);
echo "<br>";
echo next($student);
```

7) prev( ):

Syntax : mixed prev(array variable)

Purpose **:** This function will moves the pointer to the previous position in array and returns the value of element.

Example **:**

```
$student=array("Ravi","Umang");
echo next($student);
echo “<br>”;
echo prev($student);
```

8) reset( )

Syntax : mixed reset(array variable)

Purpose : This function will moves the pointer to the **first** position in array and returns the value of first element.

Example:

```
$student=array("Ravi",“Umang");
echo next($student);
echo reset($student);
```

9) end( ):

Syntax : mixed end(array variable)

Purpose : This function will moves the pointer to the last position in array and returns the value of last element.

Example:

```
$student=array("Ravi",“Umang”,“Uma”);
echo end($student);
echo reset($student);
```

10) each( ):

Syntax : Array each(array variable)

Purpose : This function returns the current key and value pair from an array and advance the array cursor.

Note: The each() function is deprecated in PHP 7.2.

Example:

```
<?php
    $student=array(1,"Monarch");
    $each=each($student);
    print_r($each);
?>
```

11) sort( ):

Syntax : void sort(array variable)

Purpose : This function will sort the items in the array in an ascending order.

Note: It will reorder the index too.

Example:

```
$rno=array(2,5,4,3,1);
sort($rno);
foreach($rno as $roll)
echo $roll;
```

12) rsort( )

Syntax : void rsort(array variable)

Purpose : This function will sort the items in the array in a descending order.

Note: It will reorder the index too.

Example:

```
$rno=array(2,5,4,3,1);
rsort($rno);
foreach($rno as $roll)
echo $roll;
```

13) asort( )

Syntax : void asort(array variable)

Purpose : Associative sorting function will sort the items in the array in an ascending order.

Note: It will NOT reorder the index.

Example:

```
$rno=array(2,5,4,3,1);
asort($rno);
foreach($rno as $roll)
echo $roll;
```

14) array_merge( )

Syntax : void array_merge(array1, array2…)

Purpose : This function is used to merge more then one array variable in array variable.

Example:

```
$no1=array(1,2,3);
$no2=array(4,5,6);
$no=array_merge($no1,$no2);
foreach($no as $roll)
echo $roll;
```

15) array_values( )

Syntax : Array array_values(array)

Purpose : This function is used to store only values to array variable it will not return any keys/index title.

Example:

```
$student=array("Rno"=>1,"nm"=>"Ramesh");
$onlyval=array_values($student);
foreach($onlyval as $val)
echo "<br>".$val;
```

16) array_keys( ):

Syntax : Array array_keys(array)

Purpose : This function is used to store only keys/index to array variable it will not return any values.

Example:

```
$student=array("Rno"=>1,"nm"=>"Ram");
$onlykey=array_keys($student);
foreach($onlykey as $key)
echo $key;
```

17) print_r( )

Syntax : void print_r(array variable)

Purpose : This function is used to display the array in the format as it is declared in the code.

Example :

```
$student=array("Rno"=>1,"nm"=>"Ram");
```

print_r($student);

18) array_key_exists( )

Syntax : Boolean array_key_exists(value to search, array variable)

Purpose : This function is used to check that given string/value key is present in the array or not.

Example:

```
$student=array("Rno"=>1,"nm"=>"Ram");
if(array_key_exists("Rno",$student))
echo "Given Key is found in array";
```

19) array_reverse( ):

Syntax : array array_reverse(array variable)

Purpose : To reverse the items present in the array.

Example:

```
$sub=array("PHP", "Oracle", "VB.Net");
$revsub=array_reverse($sub);
echo print_r($revsub);
```

20) array_push( )

Syntax : array array_push(array variable,value1,value2…)

Purpose : To insert the item in array at end of existing array.

Example:

```
$sub=array("PHP", "Oracle");
array_push($sub,"VB.Net");
echo print_r($sub);
```

21) array_pop( ):

Syntax : array array_pop(array variable)

Purpose : To remove the last item from array at end of existing array.

Example:

```
$sub=array("PHP", "Oracle");
array_pop($sub);
echo print_r($sub);
```

- **Miscellaneous Functions :**

1) Include():

Syntax **:**include(StringFilename)

Purpose **:**To access contents and function from included file we can use include function.

Example :

```
<?php
    $name=“Creative”;
?>      //Save this file with nm.php name

<?php
    include(nm.php);
    echo $name;
?>
```

2) require():

Syntax **:**void require(string file name)

Purpose **:**To access various values from file provided in require function It is as same as include function but it will generate fatal error if file not found.

Example: require(“myfun.php”);

3) **header():**

Syntax **:** header(“Location:filename”);

Purpose **:** To transfer the page from one web-page to another web-page directly without using link command.

Example :

```
<?php
$id="creative";
$pass=“bca";
if($id=="creative" and $pass==“bca")
        header("location:welcome.php");
else
        echo "Wrong ID or Password";
```

```
?>
```

4) define():

Syntax :define(“variablename”,”value”,Boolean value for case sensitivity);

Purpose : To define variable as constant with value. The value defined with **define** will not change during the program.

Example:

```
<?php
        define(“PI”,3.14);
        echo PI;
?>
```

5) constant():

Syntax :constant(String Constant Var. Name);

Purpose : To store constant name as constant in variable.

Example:

```
<?php
        define("PI",3.14);
        $val="PI";
        echo $val;
        $val=constant("PI");
        echo $val;
?>
```

6) die():

Syntax : void die( );

Purpose : To exit from current script coding.

Example :

```
<?php
    echo "Welcome";
    die( );
    echo "Use Of DIE Function";
?>
```

7) isset():

Syntax : Boolean isset(Mixed Variable Name);

Purpose : To check that any value is stored in variable or not.

Example :

```
<?php
    $value="";
    if(isset($value))
        echo "Has some value";
    else
        echo "Has No Value";
?>
```

- **File handling Functions:**

1) fopen():

Syntax **:**handler fopen(string filename, string mode)

Purpose **:**This function is used to open the file or the URL.

Example :

```
<?php
    fopen("testing.txt","r");
    fopen("d:\\creative.txt","w");  ?>
```

2) fwrite():

Syntax **:**int fwrite(file handler variable, String value)

Purpose **:**This function is used to writes contents into specified file.

Example :

```
<?php
    $fp=fopen("d:\\creative.txt","w");
    fwrite($fp,"Wel Come to MY File"); ?>
```

3) fread():

Syntax **:**int fread(file handler variable, int length)

Purpose **:**This function is used to read contents from specified file.

Example :

```
<?php
    $fp=fopen("d:\\creative.txt","r");
    $fileData=fread($fp,filesize("d:\\creative.txt"));
    echo $fileData;
?>
```

4) fclose():

Syntax : fclose(file handler variable);

Purpose : This function is used to close a specified file.

Example :

```
<?php
    $fp=fopen("d:\\monarch.txt","r");
    $fileData=fread($fp,filesize("d:\\monarch.txt"));
    echo $fileData;
    fclose($fp);
?>
```

5) file_exists():

Syntax :boolean file_exists(string file name);

Purpose : This function is used to check that given file is exists or not. If exists then it return the logical TRUE else it will returns logical FALSE.

Example :

```
<?php
    if(file_exists("d:\\creative.txt"))
        echo " creative.txt file is EXISTS";
    else
        echo " creative.txt file not FOUND";
?>
```

6) is_readable():

Syntax : boolean is_readable(string file name);

Purpose : This function is used to check that given file is readable or not. If readable then it return the logical TRUE else it will returns logical FALSE.

Example:

```
<?php
  if(is_readable("d:\creative.txt"))
        echo " creative.txt file is READABLE";
  else
        echo " creative.txt file not READABLE";
```

```
?>
```

7) is_writeable():

Syntax : boolean is_writeable(string file name);

Purpose : This function is used to check that given file is writeable or not. If writeable then it return the logical TRUE else it will returns logical FALSE.If file is read only then it will become not writable and it returns logically false.

Example :

```
<?php
        if(is_writeable("d:\monarch.txt"))
            echo "monarch.txt file is WRITEABLE";
        else
            echo "monarch.txt file not WRITEABLE";
?>
```

8) fgets():

Syntax **:** string fgets(file handler variable, integer length)

Purpose : This function is used to reads the contents of the file line by line. The type of return is string.

The difference between fgets and fread is that,

The function fgets() reads the contents till the new line is found.

The function fread() reads the contents word by word.

To get the end of file the function feof() is to be used.

Example :

```
<?php
        $fp=fopen ("d:\\monarch.txt","r");
        while(!feof($fp))
        {
                $data=fgets($fp);
                echo $data;
        }
    ?>
```

9) Fgetc

Syntax :string fgetc(file handler variable)

Purpose :This function is used to reads the contents of the file character by character.

The type of return is string.

The difference between **fgets** and **fread** is that,

The function **fgets()** reads the contents till the new line is found.

The function **fread()** reads the contents word by word.

To get the end of file the function **feof(**) is to be used.

Example :

```
<?php
    $fp=fopen ("d:\\monarch.txt","r");
    while(false !==($data=fgetc($fp)))
    {
            echo "<br>",$data;
    }
?>
```

10) file():

Syntax : Array file(string filename)

Purpose : This function is used to read the entire file in the array format and so the return type of this function is Array.This function is used to read the entire file in the array format and so the return type of this function is Array.There is no requirement to open the file, it gets automatically when the file function is used.

Example :

```
<?php
   $file=file("d:\\monarch.txt");
   foreach($file as $newarr)
   {
        echo $newarr;
   }
?>
```

11) file_get_contents():

Syntax : string file_get_contents(string filename)

Purpose : It read the contents of the file and stores it in the string format.There is no requirement to open the file, it gets automatically opened, There is no need to give the length. It will read till the end of the file.

Example :

```
<?php
   $data=file_get_contents("d:\\monarch.txt");
   echo $data;
?>
```

12) file_put_contents():

Syntax : void file_put_contents(string filename, string data to be written)

Purpose : This function writes the contents into the file.

There is no requirement to open the file it gets automatically opened for writing.

Example :

```
<?php
   $data=file_get_contents("E:\\monarch.txt");
   echo $data;
   $newdata="Other Branch : Monarch - Babra";
   file_put_contents("E:\\monarch.txt",$data);
   $data=file_get_contents("E:\\monarch.txt");
   echo $data;
?>
```

13) ftell():

Syntax : int ftell(file handler varibale);

Purpose : This function used to give the current position of the pointer in the file. It returns integer value. Its integer value starts from 0.

Example:

```
<?php
   $fp=fopen("d:\\monarch.txt","r");
```

```
    $data=fgets($fp,1);
    echo ftell($fp);
?>
```

14) rewind():

Syntax : Boolean rewind(file handler variable);

Purpose : This function moves the file pointer to the start position of the line. It returns true if start position is found, otherwise will returns false.

Example:

```
<?php  $fp=fopen("d:\\monarch.txt","r");
    $data=fgets($fp,10);
    rewind($fp);
    echo ftell($fp); ?>
```

15) Copy():

Syntax : Boolean copy(string oldfilename, stringnewfilename);

Purpose : This function used to copy the file with same contents into the new file. It returns true on successful copy of the file else it will return false.

Example :

```
<?php
    $ans=copy("d:\\monarch.txt","d:\\abc.txt");
    echo "File is Copied";
?>
```

16) unlink():

Syntax : Boolean unlink(string filename);

Purpose : This function used to delete the given file.

```
    <?php
unlink("d:\\abc.txt");
    ?>
```

17) rename():

Syntax :Boolean rename(Oldfilename, NewFilename);

Purpose :This function used to rename file.

Example :

```
<?php
   rename("d:\\monarch.txt","d:\\mon.txt");
   echo "File is Renamed"; ?>
```

18) Filesize():

Syntax :integer filesize(string filename);

Purpose :This function used to return the number of characters from file. This function will work as length function of string.

Example :

```
<?php echo filesize("d:\\mon.txt"); ?>
```

19) Fileatime:

Syntax :integer fileatime(string filename);

Purpose :This function used to return the time of last time accessing of the file.

Example : <?php echo fileatime("d:\\mon.txt"); ?>

20) Filemtime():

Syntax :integer filemtime(string filename);

Purpose :This function used to return the time of last time modification of the file.

Example : <?php echo filemtime("d:\\mon.txt"); ?>

21) fseek():

Syntax : integer fseek(file handler variable, integer offset);

Purpose : The function enables you to change your current position within a file.

Example :

```
<?php
    $filename="d:\mon.txt";
   $fp=fopen($filename,"r");
   fseek($fp,27);
   $seek=fread($fp,filesize($filename));
   echo $seek;
```

?>

22) move_uploaded_file():

Syntax : Boolean move_uploaded_file (source file, destination location)

Purpose :The function will move and uploaded file into the new location. It returns true if the file is moved else will return false.

Example:

move_uploaded_file(“D:\Student.txt”, “E:\”);

echo “File Moved Successfully”;

UNIT – 2

(Handling Form, Session Tracking & PHP Components)

➢ **What is Form**?

- A Document that containing black fields, that the user can fill the data or user can select the data. The form data is sent with the POST and GET method.
- **Ex**.

```
<form method=“ Define Method Name “>
        Name: <input type="text" name="name"><br>
        E-mail: <input type="text" name="email"><br>
       <input type="submit">
 </form>
```

❖ **GET Method:**

- Information sent from a form with the GET method is visible to everyone (all variable names and values are displayed in the URL).
- GET also has limits on the amount of information to send.
- The limitation is about 2000 characters.
- However, because the variables are displayed in the URL, it is possible to bookmark the page.
- This can be useful in some cases.
- GET may be used for sending non-sensitive data.
- GET should NEVER be used for sending passwords or other sensitive information!
- **Ex:**

```
<html>
    <body>
        <form method=“ GET “>
        Name: <input type="text" name="name"><br>
        E-mail: <input type="text" name="email"><br>
        <input type="submit">
        </form>
    </body>
</html>
```

❖ **Post Method:**

- 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.
- Moreover POST supports advanced functionality such as support for multi-part binary input while uploading files to server.
- However, because the variables are not displayed in the URL, it is not possible to bookmark the page**.**
- **EX:**

```
<html>
   <body>
      <form method=“ POST “>
         Name: <input type="text" name="name"><br>
         E-mail: <input type="text" name="email"><br>
         <input type="submit">
      </form>
   </body>
</html>
```

❖ **Session:**

- Sessions is the temporary storage area where the information to be stored. The sessions are used when there is any kind of login.
- Where there is login there is a session. Once the user gets logged in the username of the user is developed on each page until the user gets logged out.
- So the entire session depends on the user.
- **To create session follow steps are shown below**

1. Start session.
2. Set the value of the session.
   a. To start a session Using **session_start()** .
   b. To set session value Using global variable **$_SESSION [‘session_name’] =value.**
   c. Next to display session value there are two steps :

3. Start session.
4. Display using the echo function.

- **To destroy session there are 3 steps to be used**

1. Start session.
2. Clear the session value.
3. Destroy the session.
    a. To clear a session Using unset($SESSION [‘sessionname’]).
    b. To destroy a session using session_destroy().

- **Ex:**

```
<?php
    $user=“admin”; $pass=“ad123”;
  if($user==“admin” and $pass==“ad123”)
    {
      session_start();
      $SESSION[‘username’]=‘admin’;
      header(“next.php”);
    }
?>
```

- **Next.php**

```
    <?php
      session_start();
      echo $_SESSION[‘username’].
      “<a href= ‘logout.php’>Logout</a>”;
  ?>
```

- **Logout.php**

```
    <?php
      session_start();
      unset($_SESSION[‘username’]);
      session_destroy();
  ?>
```

❖ **Cookie:**

- Cookies are the temporary storage area where the information is to be stored. The cookies are seen in the address bar of the browser.
- The cookies which are created are present in the local machine and are located in c:\documents and settings\admin\cookies.
- The cookies depend on the time. To destroy the cookie it has to be done manually by deleting it from the given above location.
- **To create a cookie the setcookie().**
  - To display the cookie value using $_COOKIE ['cookie name'].
- **EX:**
  - **<?php**
    setcookie("tempCookie","cookie will expire in 1 minute",time()+60);
    echo $_SESSION["tempCookie"];
    ?>

❖ **Server Variable:**

- PHP provides various predefined variables.
- PHP provides a set of predefined arrays containing variables from the web server, the web server environment, and user input.
- Such new arrays are called superglobal variables.
- Below are super global variables which are automatically variable in every scope.

Variable	Description
$GLOBALS	Contains a reference to every variable which is currently available within the global scope of the script.
$_SERVER	This is an array containing information such as header,path etc.
$_GET	An associative array of variables passed to the current script via the HTTP GET method.
$_POST	An associative array of variables passed to the current script via the HTTP POST method.
$_FILES	An associative array of items uploaded to the current script via the HTTP POST method.
$_REQUEST	An associative array consisting of the contents of

	$_GET,$_POST, and $_COOKIE.
$_COOKIE	An associative array of variables passed to the current script via HTTP cookies.
$_SESSION	An associative array containing session variables available to the current script.

➢ **PHP Components:**

- **GD Library:**
  - GD is an open source code library for the dynamic creation of images.
  - GD is used for creating PNG, JPEG and GIF images and is commonly used to generate charts, graphics, and thumbnails on the fly.
  - While not restricted to use on the web, the most common applications of GD involve website development.
  - Using gd requires an ANSI C compiler. All popular Windows 95 and NT C compilers are ANSI C compliant.
  - The GD library can be added to the php installation if the php services are compiled besides the bundle package offered within the apache installation and therefore any additional installation won't be required and will be available within the php itself.
  - GD supports a varity of formats, below is a list of formats supported by GD and notes to their availability including read/write support.

Format	Read support	Write support	Notes
JPEG	True	True	
PNG	True	True	
GIF	True	True	
XBM	True	True	
XPM	True	True	
WBMP	True	True	
WebP	True	True	
BMP	True	True	AvailablePHP 7.2.0

- **Imagecreate(width, height):**

- The imagecreate() function is an inbuilt function in PHP which is used to create a new image. This function returns the blank image of given size

- **Imagecolorallocate($image_name , red , green , blue):**
  - The imagecolorallocate() function is an inbuilt function in PHP which is used to set the color in an image. This function returns a color which is given in RGB format.
- **Imagepng(image_name):**
  - The imagepng() function is an inbuilt function in PHP which is used to display image to browser or file. The main use of this function is to view an image in the browser,
- **Imagedestroy(image_name):**
  - The imagedestroy() function is an inbuilt function in PHP which is used to destroy an image and frees any memory associated with the image.

➢ **Example:**

```
<?php
    header('Content-type: image/jpg');
        $png_image = imagecreate(150, 150);
    imagecolorallocate($png_image, 220, 0, 0);
    imagepng($png_image);
    imagedestroy($png_image);
    ?>
```

❖ **Regular Expression:**

- A regular expression is a sequence of characters that forms a search pattern. When you search for data in a text, you can use this search pattern to describe what you are searching for.
- A regular expression can be a single character, or a more complicated pattern.
- Regular expressions can be used to perform all types of text search and text replace operations.

- **Syntax**
  - **$exp = "/creative/i";**
  - In the example above, / is the delimiter.
  - “i” is a modifier that makes the search case-insensitive.
- PHP provides a variety of functions that allow you to use regular expressions.

preg_match()	Returns 1 if the pattern was found in the string and 0 if not
preg_match_all()	Returns the number of times the pattern was found in the string, which may also be 0
preg_replace()	Returns a new string where matched patterns have been replaced with another string

- The preg_match(), preg_match_all() and preg_replace() functions are some of the most common use.

➢ **preg_match():**

- The preg_match() function will tell you whether a string contains matches of a pattern.

Ex:

```
<?php
    $str = "Visit cdmi.in";
    $pattern = "/cdmi/i";
    echo preg_match($pattern, $str); // Outputs 1
?>
```

➢ **preg_match_all():**

- The preg_match_all() function will tell you how many matches were found for a pattern in a string.

Ex:

```
<?php
    $str = "The rain in SPAIN falls mainly on the plains.";
    $pattern = "/ain/i";
    echo preg_match_all($pattern, $str); // Outputs 4
```

```
?>
```

- **preg_replace()**
  - The preg_replace() function will replace all of the matches of the pattern in a string with another string.

Ex:

```
<?php
    $str = "Visit cdmi";
    $pattern = "/cdmi/i";
    echo preg_replace($pattern, “cdmi.in", $str); // Outputs "Visit cdmi.in!"
?>
```

- **File Upload:**
  - A PHP script can be used with a HTML form to allow users to upload files to the server. Initially files are uploaded into a temporary directory and then relocated to a target destination by a PHP script.

- The process of uploading a file follows these steps.
  - The user opens the page containing a HTML form featuring a text files, a browse button and a submit button.
  - The user clicks the browse button and selects a file to upload from the local PC.
  - The full path to the selected file appears in the text filed then the user clicks the submit button.
  - The selected file is sent to the temporary directory on the server.
  - The PHP script that was specified as the form handler in the form's action attribute checks that the file has arrived and then copies the file into an intended directory.
  - The PHP script confirms the success to the user.

- **File Upload Html Form:**

```
<html>
```

```
<body>
    <form method="POST" enctype="multipart/form-data">
        <input type="file" name="image" />
        <input type="submit“ name=“upload”/>
    </form>
</body>
</html>
```

- **File Upload PHP Script:**

```
<?php  if(isset( $_POST[ ‘ save‘ ])) {
    $image_name = $_FILES[ ‘ image ’ ][ ‘ name ’ ];
    $path = ‘ image_upload ’/.$image_name;
    move_uploaded_file($_FILES[‘image‘][‘tmp_name‘],$path); }
?>
```

  - There is one global PHP variable called **$_FILES**. This variable is an associate double dimension array and keeps all the information related to uploaded file. So if the value assigned to the input's name attribute in uploading form was **file**.

$_FILES['file']['tmp_name'] – the uploaded file in the temporary directory on the web server.
$_FILES['file']['name'] – the actual name of the uploaded file.
$_FILES['file']['size'] – the size in bytes of the uploaded file.
$_FILES['file']['type'] – the MIME type of the uploaded file.
$_FILES['file']['error'] – the error code associated with this file upload.

- **AJAX:**
  - **Ajax: Asynchronous JavaScript and XML**
    - not a programming language; a particular way of using JavaScript
    - downloads data from a server in the background
    - allows dynamically updating a page without making the user wait
    - avoids the "click-wait-refresh" pattern
    - Example: Google Suggest
  - **A typical Ajax request :**

- user clicks, invoking an event handler
- handler's code creates an XMLHttpRequest object
- XMLHttpRequest object requests page from server
- server retrieves appropriate data, sends it back
- XMLHttpRequest fires an event when data arrives
- this is often called a callback
- you can attach a handler function to this event
- your callback event handler processes the data and displays it

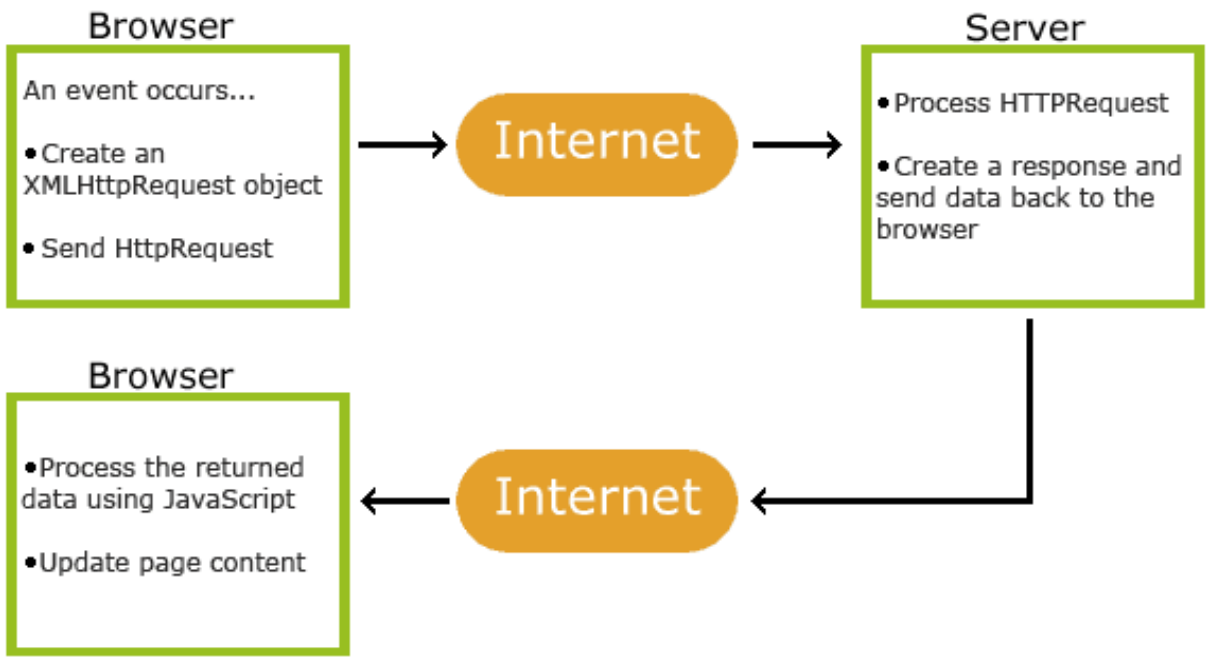
➢ XMLHttpRequest Object Methods:

Method	Description
new XMLHttpRequest()	Creates a new XMLHttpRequest object
abort()	Cancels the current request
getAllResponseHeaders()	Returns header information
getResponseHeader()	Returns specific header information
open(*method,url,async,user,psw*)	Specifies the request *method*: the request type GET or POST *url*: the file location *async*: true (asynchronous) or false (synchronous) *user*: optional user name *psw*: optional password
send()	Sends the request to the server Used for GET requests
send(*string*)	Sends the request to the server. Used for POST requests
setRequestHeader()	Adds a label/value pair to the header to be sent

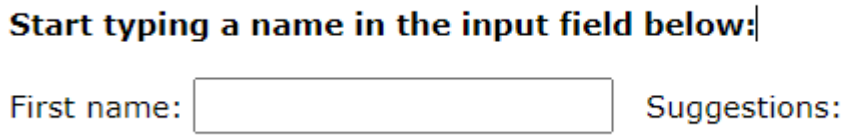
➢ XMLHttpRequest Object Properties:

Property	Description
onreadystatechange	Defines a function to be called when the readyState property changes
readyState	Holds the status of the XMLHttpRequest. 0: request not initialized 1: server connection established 2: request received 3: processing request 4: request finished and response is ready
responseText	Returns the response data as a string
responseXML	Returns the response data as XML data
status	Returns the status-number of a request 200: "OK" 403: "Forbidden" 404: "Not Found"
statusText	Returns the status-text (e.g. "OK" or "Not Found")

➢ How to Work Ajax:



Example:



In the example, when a user types a character in the input field, a function called "showHint()" is executed.

The function is triggered by the onkeyup event.

Here is the HTML code:

ajax.html

```
<html>
<head>
<script>
function showHint(str) {
   if (str.length == 0) {
      document.getElementById("txtHint").innerHTML = "";
      return;
   } else {
      var xmlhttp = new XMLHttpRequest();
      xmlhttp.onreadystatechange = function() {
         if (this.readyState == 4 && this.status == 200) {
            document.getElementById("txtHint").innerHTML=
this.responseText;
         }
      };
      xmlhttp.open("GET", "gethint.php?q=" + str, true);
      xmlhttp.send();
   }
}
</script>
</head>
```

```
<body>
<p><b>Start typing a name in the input field below:</b></p>
<form>
First name: <input type="text" onkeyup="showHint(this.value)">
</form>
<p>Suggestions: <span id="txtHint"></span></p>
</body>
</html>
```

- **Code explanation:**

- First, check if the input field is empty (str.length == 0). If it is, clear the content of the txtHint placeholder and exit the function.
- However, if the input field is not empty, do the following:
  - Create an XMLHttpRequest object
  - Create the function to be executed when the server response is ready
  - Send the request off to a PHP file (gethint.php) on the server
  - Notice that q parameter is added gethint.php?q="+str
  - The str variable holds the content of the input field
  - The PHP file checks an array of names, and returns the corresponding name(s) to the browser:

gethint.php

```
<?php
// Array with names
$a[] = "Anna";
$a[] = "Brittany";
$a[] = "Cinderella";
$a[] = "Diana";
$a[] = "Eva";
$a[] = "Fiona";
$a[] = "Gunda";
$a[] = "Hege";
$a[] = "Inga";
```

```
$a[] = "Johanna";
$a[] = "Kitty";
$a[] = "Linda";
$a[] = "Nina";
$a[] = "Ophelia";
$a[] = "Petunia";
$a[] = "Amanda";
$a[] = "Raquel";
$a[] = "Cindy";
$a[] = "Doris";
$a[] = "Eve";
$a[] = "Evita";
$a[] = "Sunniva";
$a[] = "Tove";
$a[] = "Unni";
$a[] = "Violet";
// echo '<pre>';print_r($a);
// get the q parameter from URL
$q = $_REQUEST["q"];
$hint = "";
// lookup all hints from array if $q is different from ""
if ($q !== "") {
   $q = strtolower($q);
   $len=strlen($q);
   foreach($a as $name) {
      if (stripos($name,$q)!==false) {
         if ($hint === "") {
            $hint = $name;
         } else {
            $hint .= ", $name";
         }
      }
```

```
    }
  }
  // Output "no suggestion" if no hint was found or output correct values
  echo $hint === "" ? "no suggestion" : $hint;
  ?>
```

- **Ajax With Post Method:**

```
function showHint(str) {
    if (str.length == 0) {
        document.getElementById("txtHint").innerHTML = "";
        return;
    } else {
        var xmlhttp = new XMLHttpRequest();
        xmlhttp.onreadystatechange = function() {
            if (this.readyState == 4 && this.status == 200) {
                document.getElementById("txtHint").innerHTML=
this.responseText;
            }
        };
        xmlhttp.open("POST", "gethint.php", true);
        xmlhttp.setRequestHeader("Content-type",        "application/x-www-
form-urlencoded");
        xmlhttp.send("q="+str);
    }
}
```

- **MySql with AJAX:**

hint.php:

```
<?php
$q = $_REQUEST["q"];
$hint = "";
if ($q !== "") {
    $con = mysqli_connect("localhost","root","","school");
    $sql = "select * from students where name like '%$q%'";
```

```
    $res = mysqli_query($con,$sql);
    while($row = mysqli_fetch_assoc($res)) {
        if ($hint === "") {
            $hint = $row['name'];
        } else {
            $hint .= ", ".$row['name'];
        }
    }
}
echo $hint === "" ? "no suggestion" : $hint;
?>
```

- **jQuery Ajax:**

Index.html

```
<button id="demo">click to get Randomno</button>
<h1 id="print_data"></h1>

<script type="text/javascript" src="jquery-3.6.0.min.js"></script>
<script type="text/javascript">
    $(document).ready(function(){
        $('#demo').click(function(){

            $.ajax({
                type:'get',
                url:'second.php',

                success:function(res)
                {
                    $('#print_data').html(res);
                }
            })
        })
    })
</script>
```

Second.php

```
<?php   $a = rand(0,100);  echo $a;  ?>
```

jQuery Ajax with MySQL:

ajax_jquery.html

```
<html>
<head>
<script src="https://code.jquery.com/jquery-3.6.3.min.js" ></script>
<script>
   $(document).ready(function(){
      $('#txt').keyup(function(){
         var name = $(this).val()
         $.ajax({
            type:'get',
            url:'hint.php',
            data:{'q':name},
            success:function(response){
               $('#txtHint').html(response)
            }
         })
      })
   })
</script>
</head>
<body>
<p><b>Start typing a name in the input field below:</b></p>
<form>
First name: <input type="text" id="txt">
</form>
<p>Suggestions: <span id="txtHint"></span></p>
```

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

Hint.php

```
<?php
$q = $_REQUEST["q"];
$hint = "";
if ($q !== "") {
    $con = mysqli_connect("localhost","root","","school");
    $sql = "select * from students where name like '%$q%'";
    $res = mysqli_query($con,$sql);
    while($row = mysqli_fetch_assoc($res)) {
        if ($hint === "") {
            $hint = $row['name'];
        } else {
            $hint .= ", ".$row['name'];
        }

    }
}
echo $hint === "" ? "no suggestion" : $hint;
?>
```

❖ JSON:

- What is JSON?
  - JSON stands for **J**ava**S**cript **O**bject **N**otation
  - JSON is a text format for storing and transporting data
  - JSON is "self-describing" and easy to understand
  - JSON is a lightweight data-interchange format
  - JSON is plain text written in JavaScript object notation
  - JSON is used to send data between computers
  - JSON is language independent *
  - **Example of JSON String:**

    '{"name":"John", "age":30, "car":null}'
- JSON Resource Types :
  - In JSON, values must be one of the following data types:
  - a string
  - a number
  - an object (JSON object)
  - an array
  - a boolean
  - *null*
  - JSON Strings Example:
    - {"name":"John"}
  - JSON Numbers:
    - {"age":30}
  - JSON Objects:
    - {"employee":{"name":"John", "age":30, "city":"New York"} }
  - JSON Arrays:
    - { "employees":["John", "Anna", "Peter"] }
  - JSON Booleans:
    - {"sale":true}
  - JSON null:
    - {"middlename":null}

Json_encode() function:

- The json_encode() function is used to encode a value to JSON format.
- Syntax:
- json_encode(value, options, depth)

- Example:

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

Json_decode() function:

- The json_decode() function is used to decode or convert a JSON object to a PHP object.
- Syntax:
- json_decode(*string, assoc, depth, options)*
- *Example:*

```
<?php
$jsonobj = '{"Peter":35,"Ben":37,"Joe":43}';
var_dump(json_decode($jsonobj, true));
?>
```

UNIT – 3

(Introduction of SQL)

Introduction of MySQL:

- What is MySQL?
    - MySQL is a database system used on the web
    - MySQL is a database system that runs on a server
    - MySQL is ideal for both small and large applications
    - MySQL is very fast, reliable, and easy to use
    - MySQL uses standard SQL
    - MySQL compiles on a number of platforms
    - MySQL is free to download and use
    - MySQL is developed, distributed, and supported by Oracle Corporation

The data in a MySQL database are stored in tables. A table is a collection of related data, and it consists of columns and rows.

MYSQL using phpmyadmin:

- phpMyAdmin?
    - phpMyAdmin is a fully capable web app, written in php, that will allow you to manage and edit databases easily. It provides a robust interface with lots of options for creating databases and tables, managing database accounts, manually adding data to a database, and so much more!
- What can phpMyAdmin do?
    - phpMyAdmin is a fully features database management tool. It can do so much. Here is a short list of just some of the things the web app can do:
        - create, edit, and delete databases
        - manage user accounts, and all their permissions
        - run custom SQL commands right in the app
        - manually add data to database tables
        - and so much more!

SQL DML Commands:

- DML(Data Manipulation Language): ?
  - The SQL commands that deals with the manipulation of data present in the database belong to DML or Data Manipulation Language and this includes most of the SQL statements. It is the component of the SQL statement that controls access to data and to the database. Basically, DCL statements are grouped with DML statements.
    - **List of DML commands:**
      - INSERT : It is used to insert data into a table.
      - UPDATE: It is used to update existing data within a table.
      - DELETE : It is used to delete records from a database table.
      - SELECT : It is used to select records from a database table.
- **DML command Syntax?**
  - **Insert:**
    - Insert into table_name (field_name) values (‘ value ‘);
  - **Update:**
    - Updtae table_name set field_name = “ value ” where column_name = ‘ values ’;
  - **Delete:**
    - Delete from table_name where column_name = “ value ”;
  - **Select:**
    - Select * from table_name;

php mysql connectivity:

- **Connection to MySQL?**

Before we can access data in the MySQL database, we need to be able to connect to the server:

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

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

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

- **PHP mysqli Functions:**
  1) **mysqli_connect() :** Open a connection to a MySQL Server
     Example:

```php
<?php
$con = mysqli_connect("localhost","root","","school");
if($con){
        echo "Successfully Connected";
}else{
        echo mysqli_connect_error();
}
?>
```

  2) **mysqli_close():** Close MySQL connection.
     Example:

```php
<?php
$con = mysqli_connect("localhost","root","","school");
if($con){
        echo "Successfully Connected";
}else{
```

```
        echo mysqli_connect_error();
}
$sql = "select * from students";
$res = mysqli_query($con,$sql);
$row = mysqli_fetch_assoc($res);
echo '<pre>';
print_r($row);
mysqli_close($con);
$sql = "select * from students";
$res = mysqli_query($con,$sql);
$row = mysqli_fetch_assoc($res);
echo '<pre>';
print_r($row);
?>
```

3) **mysqli_error():** Returns the text of the error message from previous MySQL operation.
   - Example:

```
<?php
$con = mysqli_connect("localhost","root","","school");
$sql = "select * from students1";
$res = mysqli_query($con,$sql);
if(!$res){
        echo mysqli_error($con);
}else{
        $row = mysqli_fetch_assoc($res);
        echo '<pre>';
        print_r($row);
}
?>
```

4) **mysqli_errno():** Returns the numerical value of the error message from previous MySQL operation.

   Example:

```
<?php
$con = mysqli_connect("localhost","root","","school");
$sql = "select * from students1";
$res = mysqli_query($con,$sql);
if(!$res){
	echo mysqli_errno($con);
}else{
	$row = mysqli_fetch_assoc($res);
	echo '<pre>';
	 print_r($row);
}
?>
```

5) mysqli_select_db(): Select a MySQL database.

Example:

```
<?php
$con = mysqli_connect("localhost","root","","school");
$sql = "select * from students1";
$res = mysqli_query($con,$sql);
mysqli_select_db($con,'temp');
$sql = "select * from admin";
$res = mysqli_query($con,$sql);
if(!$res){
                                        echo mysqli_error($con);
}else{
                                        $row                    =
mysqli_fetch_assoc($res);
                                        echo '<pre>';
                                        print_r($row);
}
?>
```

6) **mysqli_query():** Send a MySQL query.

Example:

```
<?php
$con = mysqli_connect("localhost","root","","school");
if($con){
        echo "Successfully Connected";
}else{
        echo mysqli_connect_error();
}
$sql = "select * from students";
$res = mysqli_query($con,$sql);
$row = mysqli_fetch_assoc($res);
echo '<pre>';
print_r($row);
?>
```

7) **mysqli_fetch_array ():** Fetch a result row as an associative array, a numeric array, or both.

Example:

```
<?php
$con = mysqli_connect("localhost","root","","school");
if($con){
        echo "Successfully Connected";
}else{
        echo mysqli_connect_error();
}
$sql = “select * from students”;
$res = mysqli_query($con,$sql);
$row = mysqli_fetch_array($res);
echo ‘<pre>’;
print_r($row);
?>
```

8) **mysqli_num_rows() :** Get number of rows in result.

Example:

```
<?php
```

```
$con = mysqli_connect("localhost","root","","school");
$sql = "select * from students";
$res = mysqli_query($con,$sql);
if(!$res){
                                        echo mysqli_error($con);
}else{
                                        $cnt                    =
mysqli_num_rows($res);
                                        echo $cnt;
}
?>
```

9) **mysqli_affected_rows():** Get number of affected rows in previous MySQL operation.

Example:

```
<?php
$con = mysqli_connect("localhost","root","","school");
$sql = “update students set name=‘abc’ where id>2";
$res = mysqli_query($con,$sql);
if(!$res){
      echo mysqli_error($con);
}else{
      $cnt = mysqli_affected_rows($con);
      echo $cnt;
}
?>
```

10) **mysqli_fetch_assoc():**Fetch a result row as an associative array.

Example:

```
<?php
$con = mysqli_connect("localhost","root","","school");
if($con){
   echo "Successfully Connected";
```

```
}else{
    echo mysqli_connect_error();
}
$sql = "select * from students";
$res = mysqli_query($con,$sql);
$row = mysqli_fetch_assoc($res);
echo '<pre>';
print_r($row);
?>
```

11) **mysqli_fetch_field():** Get column information from a result and return as an object.

Example:

```
<?php
$con = mysqli_connect("localhost","root","","school");
$sql = "select * from students";
$res = mysqli_query($con,$sql);
if(!$res){
    echo mysqli_error($con);
}else{
    $row = mysqli_fetch_field($res);
    echo '<pre>';
    print_r($row);
}
?>
```

12) **mysqli_fetch_object():** Fetch a result row as an object.

Example:

```
<?php
$con = mysqli_connect("localhost","root","","school");
if($con){
    echo "Successfully Connected";
}else{
    echo mysqli_connect_error();
```

```
}
$sql = "select * from students";
$res = mysqli_query($con,$sql);
$row = mysqli_fetch_object($res);
echo '<pre>';
print_r($row);
?>
```

13) **mysqli_fetch_row():** Get a result row as an enumerated array.

Example:

```
<?php
$con = mysqli_connect("localhost","root","","school");
if($con){
   echo "Successfully Connected";
}else{
   echo mysqli_connect_error();
}
$sql = "select * from students";
$res = mysqli_query($con,$sql);
$row = mysqli_fetch_row($res);
echo '<pre>';
print_r($row);
?>
```

14) **mysqli_insert_id():** Get the ID generated in the last query.

Example:

```
<?php
$con = mysqli_connect("localhost","root","","school");
$sql = "insert into students(name,email,contact,gender,age)
values ('raj','raj@gmail.com',123,'male',12)";
$res = mysqli_query($con,$sql);
if(!$res){
   echo mysqli_error($con);
}else{
```

```
    $id = mysqli_insert_id($con);
    echo $id;
}
?>
```

15) **mysqli_num_fields():** Get number of fields in result.

Example:

```
<?php
$con = mysqli_connect("localhost","root","","school");
$sql = "select * from students";
$res = mysqli_query($con,$sql);
if(!$res){
    echo mysqli_error($con);
}else{
    $id = mysqli_num_fields($res);
    echo $id;
}
?>
```

16) **mysqli_field_seek():**Set result pointer to a specified field offset.

Example:

```
<?php
$con = mysqli_connect("localhost","root","","school");
$sql = "select * from students";
$res = mysqli_query($con,$sql);
if(!$res){
    echo mysqli_error($con);
}else{
    mysqli_field_seek($res, 1);
    $fieldinfo = mysqli_fetch_field($res);
    echo '<pre>';print_r($fieldinfo);
}
?>
```

17) **mysqli_data_seek():** adjusts the result pointer to an arbitrary row in the result-set.

Example:

```
<?php
$con = mysqli_connect("localhost","root","","school");
$sql = "select * from students";
$res = mysqli_query($con,$sql);
if(!$res){
    echo mysqli_error($con);
}else{
    mysqli_data_seek($res,14);
   $row=mysqli_fetch_row($res);
    echo '<pre>';print_r($row);
}
?>
```

UNIT – 4

(jQuery)

❖ What is jQuery?

- jQuery is a JavaScript library. That means, it is built using JavaScript, it accepts Javascript methods and functions. But, jQuery can do more in a single line of code than JavaScript can do in a whole function. This shortens development time and let's face it, makes our jobs easier.
- **jQuery look like this:**

```
$(document).ready(function(){
        //code here
});
```

❖ Syntax of jQuery:

- The jQuery syntax is tailor-made for selecting HTML elements and performing some action on the element(s).
- **Basic syntax is: $(selector).action()**
  - A $ sign to define/access jQuery
  - A (selector) to "query (or find)" HTML elements
  - A jQuery action() to be performed on the element(s)
- **Examples:**
  - $(this).hide() - hides the current element.
  - $("p").hide() - hides all <p> elements.
  - $(".test").hide() - hides all elements with class="test".
  - $("#test").hide() - hides the element with id="test**".**

❖ What is selector?

- A jQuery Selector is a function which makes use of expressions to find out matching elements from a DOM based on the given criteria. Simply you can say, selectors are used to select one or more HTML elements using jQuery. Once an element is selected then we can perform various operations on that selected element.
- How to Use Selectors?
  - The selectors are very useful and would be required at every step while using jQuery. They get the exact element that you want from your HTML document.

- **The $() function:**
  - jQuery selectors start with the dollar sign and parentheses − $(). The factory function $() makes use of following three building blocks while selecting elements in Tag Name , Ta Id, Tag Class.
  - **Tag Name:**
    - Represents a tag name available in the DOM. For example $('p') selects all paragraphs <p> in the document.
  - **Tag ID:**
    - Represents a tag available with the given ID in the DOM. For example $('#some-id') selects the single element in the document that has an ID of some-id.
  - **Tag Class:**
    - Represents a tag available with the given class in the DOM. For example $('.some-class') selects all elements in the document that have a class of some-class.

All the above items can be used either on their own or in combination with other selectors. All the jQuery selectors are based on the same principle except some tweaking.

- **<u>The element Selector (Tag Name):</u>**
  - The jQuery element selector selects elements based on the element name.
  - You can select all <p> elements on a page like this.
  - **Example:**

```
<script>
    $(document).ready(function(){
        $("button").click(function(){
            $("p").hide();
        });
    });
</script>
<h2>This is a heading</h2>
```

```
<p>This is a paragraph.</p>
<p>This is another paragraph.</p>
<button>Click me to hide paragraphs</button>
```

- **The #id Selector (Tag ID):**
  - The jQuery #id selector uses the id attribute of an HTML tag to find the specific element.
  - An id should be unique within a page, so you should use the #id selector when you want to find a single, unique element.To find an element with a specific id, write a hash character, followed by the id of the HTML element:
  - **Example:**

```
<script>
    $(document).ready(function(){
        $("button").click(function(){
            $("#test").hide();
        });
    });
</script>
    <h2>This is a heading</h2>
    <p>This is a paragraph.</p>
    <p id="test">This is another paragraph.</p>
    <button>Click me</button>
```

- **The .class Selector (Tag Class):**
  - The .class selector selects all elements with the specific class.
  - The class refers to the class attribute of an HTML element.
  - The class attribute is used to set a particular style for several HTML elements.Select all elements with class Name.
  - **Ex.**

```
<script>
    $(document).ready(function(){
        $(".intro").css("background-color","yellow");
    });
```

```
    </script>
  <h1>Welcome to My Homepage</h1>
  <p class="intro">My name is Donald.</p>
  <p>I live in Duckburg.</p>
  <div class="intro">My name is Dolly.</div>
  <p>I live in Duckburg.</p>
```

❖ **jQuery Events:**

- jQuery events are the actions that can be detected by your web application. They are used to create dynamic web pages. An event shows the exact moment when something happens.
- **These are some examples of Events:**
  - A mouse click
  - An HTML form submission
  - A web page loading
  - A keystroke on the keyboard
  - Scrolling of the web page etc.

Mouse Events

- click
- dblclick
- mouseenter
- mouseleave

Form Events

- submit
- change
- blur
- focus

Document/Window

- load
- unload
- scroll
- resize

Keyboard Events

- keyup
- keydown
- keypress

❖ **jQuery Events:**

1) **.click() :** Bind an event handler to the “click” JavaScript event, or trigger that event on an element.

Ex:

```
$(document).ready(function(){
    $("p").click(function(){
        alert("The paragraph was clicked.");
    });
});
```

2) **.dblclick() :** Bind an event handler to the “dblclick” JavaScript event, or trigger that event on an element.

Ex:

```
$(document).ready(function(){
    $("p").dblclick(function(){
        alert("The paragraph was double-clicked.");
    });
});
```

3) **.keypress():** Bind an event handler to the “keypress” JavaScript event, or trigger that event on an element.

Ex:

```
i = 0;
$(document).ready(function(){
    $("input").keypress(function(){
        $("span").text(i += 1);
    });
});
```

4) **.keydown():** Bind an event handler to the “keydown” JavaScript event, or trigger that event on an element.

Ex:

```
i = 0;
$(document).ready(function(){
    $("input"). keydown(function(){
```

```
            $("span").text(i += 1);
        });
    });
```

5) **.keyup():** Bind an event handler to the “keyup” JavaScript event, or trigger that event on an element.

Ex:

```
i = 0;
$(document).ready(function(){
        $("input"). keyup(function(){
            $("span").text(i += 1);
        });
});
```

6) **.submit():** Bind an event handler to the “submit” JavaScript event, or trigger that event on an element.

Ex:

```
$(document).ready(function(){
        $("form").submit(function(){
            alert("Submitted");
        });
});
```

7) **.change():** Bind an event handler to the “change” JavaScript event, or trigger that event on an element.

Ex:

```
$(document).ready(function(){
        $("input").change(function(){
            alert("The text has been changed.");
        });
});
```

8) **.focus():** Bind an event handler to the “focus” JavaScript event, or trigger that event on an element.

Ex:

```
$(document).ready(function(){
```

```
        $("input").focus(function(){
            $("span").css("display", "inline").fadeOut(2000);
        });
    });
```

9) **.blur():** Bind an event handler to the "blur" JavaScript event, or trigger that event on an element.

Ex:

```
    $(document).ready(function(){
        $("input").blur(function(){
            alert("This input field has lost its focus.");
        });
    });
```

10) **.load() :** Bind an event handler to the "load" JavaScript event.

Note: The load() method was deprecated in jQuery version 1.8 and removed in version 3.0. Use the on() or trigger() method instead.

Ex:

```
$(document).ready(function(){
        $("img").load(function(){
            alert("Image loaded.");
    });
 });
```

11) **.resize() :** Bind an event handler to the "resize" JavaScript event, or trigger that event on an element.

Ex:

```
 var x = 0;
 $(document).ready(function(){
        $(window).resize(function(){
            $("span").text(x += 1);
    });
 });
```

12) **.scroll() :** Bind an event handler to the "scroll" JavaScript event, or trigger that event on an element.

Ex:

```
var x = 0;
 $(document).ready(function(){
        $("div").scroll(function(){
            $("span").text( x+= 1);
        });
 });
```

13). **.unload() :** Bind an event handler to the "unload" JavaScript event.

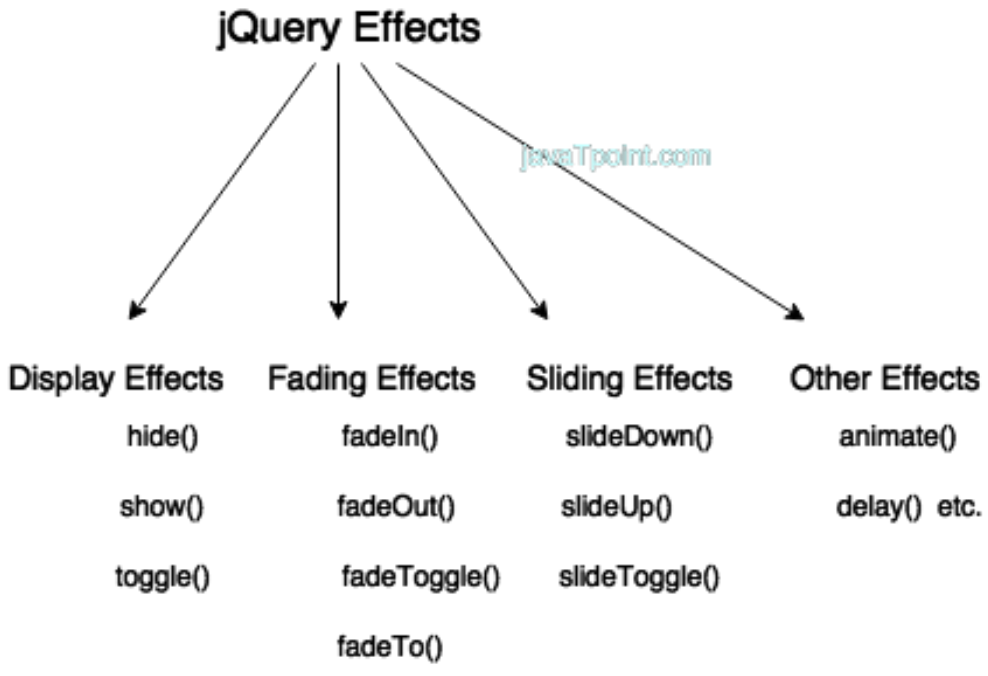
Note: The unload() method was deprecated in jQuery version 1.8 and removed in version 3.0. Use the on() or trigger() method instead.

Ex:

```
$(document).ready(function(){
        $(window).unload(function(){
            alert("Goodbye!");
        });
 });
```

❖ **jQuery Effects:**

- jQuery enables us to add effects on a web page. jQuery effects can be categorized into fading, sliding, hiding/showing and animation effects. jQuery provides many methods for effects on a web page.



- **hide, show and toggle:**
  - **The show()** method shows the hidden, selected elements.
  - **The hide()** method hidden the show, selected elements.
  - **The toggle()** method hidden the show and show the hidden selected elements.
  - **Example:**

    Script

```
<script>
    $(document).ready(function(){
      $(".btn1").click(function(){
            $("p").hide();
      });
      $(".btn2").click(function(){
      $("p").show();
      });
      $(".btn3").click(function(){
      $("p").toggle('slow');
      });
    });
</script>
```

    Html

```
<p>This is a paragraph.</p>
<button class="btn1">Hide</button>
<button class="btn2">Show</button>
<button class="btn3">Toggle</button>
```

- **jQuery Fade:**
  - With jQuery you can fade an element in and out of visibility.

    jQuery has the following fade methods:
    - fadeIn()
    - fadeOut()
    - fadeToggle()
    - fadeTo()

- **fadeIn() Method:**
  - The fadeIn() method gradually changes the opacity, for selected elements, from hidden to visible (fading effect).
  - **Example:**

    Script:

```
$(document).ready(function(){
   $("button").click(function(){
   $("#div1").fadeIn();
   $("#div2").fadeIn("slow");
         $("#div3").fadeIn(3000);
   });
});
```

    Html:

```
<button>Click to fadeTo boxes</button><br><br>
<div id="div1" style="width:80px;height:80px;background-color:red; display: none;"></div>
<div id="div2" style="width:80px;height:80px;background-color:green; display: none;"></div>
<div id="div3" style="width:80px;height:80px;background-color:blue; display: none;"></div>
```

- **fadeOut()**
  - the jQuery fadeOut() method is used to fade out a visible element.
  - **Example:**

    Script:

```
$(document).ready(function(){
  $("button").click(function(){
    $("#div1").fadeOut();
    $("#div2").fadeOut("slow");
    $("#div3").fadeOut(3000);
  });
});
```

HTML:

```
<button>Click to fadeToggle boxes</button><br><br>
<div id="div1" style="width:80px;height:80px;background-color:red;"> </div>
<div id="div2" style="width:80px;height:80px;background-color:green;"> </div>
<div id="div3" style="width:80px;height:80px;background-color:blue;"> </div>
```

- **.fadeToggle():**
  - The jQuery fadeToggle() method toggles between the fadeIn() and fadeOut() methods.
  - **Example:**

    Script:

```
$(document).ready(function(){
  $("button").click(function(){
    $("#div1"). fadeToggle();
    $("#div2"). fadeToggle("slow");
    $("#div3"). fadeToggle(3000);
  });
});
```

    Html:

```
<button>Click to fadeToggle boxes</button><br><br>
<div id="div1" style="width:80px;height:80px;background-color:red;"> </div>
<div id="div2" style="width:80px;height:80px;background-color:green;"> </div>
<div id="div3" style="width:80px;height:80px;background-color:blue;"> </div>
```

- **.fadeTo():**
  - The jQuery fadeTo() method allows fading to a given opacity (value between 0 and 1)..
  - **Example:**

Script:

```
$(document).ready(function(){
 $("button").click(function(){
    $("#div1"). fadeTo("slow", 0.15);
    $("#div2"). fadeTo("slow", 0.4);
    $("#div3"). fadeTo("slow", 0.7);
  });
});
```

Html:

```
<button>Click to fadeTo boxes</button><br><br>
<div id="div1" style="width:80px;height:80px;background-
color:red;"> </div>
<div id="div2" style="width:80px;height:80px;background-
color:green;"> </div>
<div id="div3" style="width:80px;height:80px;background-
color:blue;"> </div>
```

❖ **jQuery slides:**

- With jQuery you can create a sliding effect on elements.
  - jQuery has the following slids methods:
  - slideDown()
  - slideUp()
  - slideToggle()

➢ **slideDown():**

- The jQuery slideDown() method is used to slide down an element. The optional speed parameter specifies the duration of the effect. It can take the following values: "slow", "fast", or milliseconds.
  - **Example:**

Script:

```
$(document).ready(function(){
        $("#flip").click(function(){
        $("#panel").slideDown("slow");
        });
```

```
});
```

Css:

```
#flip {
    padding: 5px;
}
#panel {
    padding:50px;
    background-color: #e5eecc;
    border: solid 1px #c3c3c3;
    text-align: center;
    display:none;
}
```

Html:

```
<div id="flip">Click to slide down panel</div>
<div id="panel">Hello world!</div>
```

- **.slideUp():**
  - The jQuery slideUp() method is used to slide up an element.
  - **Example:**

    Script:

```
$(document).ready(function(){
    $("#flip").click(function(){
    $("#panel").slideUp("slow");
    });
});
```

CSS:

```
#flip {
    padding: 5px;
}
#panel {
    padding:50px;
    background-color: #e5eecc;
```

```
        border: solid 1px #c3c3c3;
        text-align: center;
    }
```

HTML:

```
<div id="flip">Click to slide up panel</div>
  <div id="panel">Hello world!</div>
```

- **.slideToggle()**
  - The jQuery slideToggle() method toggles between the slideDown() and slideUp() methods.
  - **Example:**

Script:

```
$(document).ready(function(){
    $("#flip").click(function(){
    $("#panel").slideToggle("slow");
    });
});
```

CSS:

```
#flip {
    padding: 5px;
}
#panel {
    padding:50px;
    background-color: #e5eecc;
    border: solid 1px #c3c3c3;
    text-align: center;
}
```

HTML:

```
<div id="flip">Click to slide toggle panel</div>
<div id="panel">Hello world!</div>
```

❖ **jQuery Methods:**

➢ **.css():**

- The .css() method sets or returns one or more style properties for the selected elements.
- **Example:**

```
<script>
$(document).ready(function(){
  $("button").click(function(){
    alert("Background color = " + $("p").css("background-color"));
  });
});
</script>
<p style="background-color:#0000ff">This is a paragraph.</p>
<button>Click</button>
```

➢ **.height():**

- Get the current computed height for the first element in the set of matched elements or set the height of every matched.
- **Example:**

```
<script>
$(document).ready(function(){
  $("button").click(function(){
    alert("Height of div: " + $("#div1").height());
  });
});
</script>
<div id=”div1”
style="height:100px;width:300px;padding:10px;margin: 3px; border:
1px solid blue;background-color:lightblue;"></div>
<br>
<button>Click</button>
```

- **.width():**
  - Get the current computed width for the first element in the set of matched elements or set the width of every matched element.
  - **Example:**

```
<script>
$(document).ready(function(){
  $("button").click(function(){
    alert(“Width of div: " + $("#div1").width());
  });
});
</script>
<div id=”div1”
style="height:100px;width:300px;padding:10px;margin:
3px;border:1px solid blue;background-color:lightblue;"></div><br>
<button>Display the height of div</button>
```

- **.innerWidth():**
  - Get the current computed inner width for the first element in the set of matched elements, including padding but not border.
  - **Example:**

```
<script>
$(document).ready(function(){
  $("button").click(function(){
    alert(“innerwidth of div: " + $("div").innerwidth());
  });
});
</script>
<div
style="height:100px;width:300px;padding:10px;margin:3px;border:1
px solid blue;background-color:lightblue;"></div><br>
<button>Click</button>
```

- **.innerHeight():**
  - Get the current computed height for the first element in the set of matched elements, including padding but not border.
  - **Example:**

```
<script>
$(document).ready(function(){
  $("button").click(function(){
    alert(“innerheight of div: " + $("div"). innerHeight());
  });
});
</script>
<div
style="height:100px;width:300px;padding:10px;margin:3px;border:1
px solid blue;background-color:lightblue;"></div><br>
<button>Click</button>
```

- **.outerWidth():**
  - Get the current computed outer width (including padding, border, and optionally margin) for the first element in the set of matched elements.
  - **Example:**

```
<script>
$(document).ready(function(){
  $("button").click(function(){
    alert(“outerWidth of div: " + $("#div1"). outerWidth());
  });
});
</script>
<div id=”div1”
style="height:100px;width:300px;padding:10px;margin:
3px;border:1px solid blue;background-color:lightblue;"></div><br>
<button>Click</button>
```

- **.outerHeight():**
  - Get the current computed outer height (including padding, border, and optionally margin) for the first element in the set of matched elements.
  - **Example:**

```
<script>
$(document).ready(function(){
  $("button").click(function(){
    alert(“outerHeight of div: " + $("div"). outerHeight());
  });
});
</script>
<div
style="height:100px;width:300px;padding:10px;margin:3px;border:1
px solid blue;background-color:lightblue;"></div><br>
<button>Click</button>
```

- **.html():**
  - The html() method sets or returns the content (innerHTML) of the selected elements.
  - When this method is used to return content, it returns the content of the FIRST matched element.
  - When this method is used to set content, it overwrites the content of ALL matched elements.
  - **Example:**

```
<script>
$(document).ready(function(){
  $("button").click(function(){
    $("p").html("Hello <b>world!</b>");
  });
});
```

```
</script>
<button>Click</button>
<p>This is a paragraph.</p>
<p>This is another paragraph.</p>
```

- **.text():**
  - The text() method sets or returns the text content of the selected elements.
  - When this method is used to return content, it returns the text content of all matched elements (HTML markup will be removed).
  - When this method is used to set content, it overwrites the content of ALL matched elements.
  - **Example:**

```
<script>
$(document).ready(function(){
  $("button").click(function(){
    $("p").text("Hello <b>world!</b>");
  });
});
</script>
<button>Click</button>
<p>This is a paragraph.</p>
<p>This is another paragraph.</p>
```

- **.append():**
  - The append() method inserts specified content at the end of the selected elements.
  - **Example:**

```
<script>
$(document).ready(function(){
  $("#btn1").click(function(){
    $("p").append(" <b>Appended text</b>.");
  });
});
```

```
</script>
<button>Click</button>
<p>This is a paragraph.</p>
<p>This is another paragraph.</p>
```

- **.prepend():**
  - The prepend() method inserts specified content at the beginning of the selected elements.
  - **Example:**

```
<script>
$(document).ready(function(){
  $("#btn1").click(function(){
    $("p"). prepend(" <b>Appended text</b>.");
  });
});
</script>
<button>Click</button>
<p>This is a paragraph.</p>
<p>This is another paragraph.</p>
```

- **.after():**
  - The after() method inserts specified content after the selected elements.
  - **Example:**

```
<script>
$(document).ready(function(){
  $("button").click(function(){
    $("p").after("<p>Hello world!</p>");
  });
});
</script>
<button>Click</button>
```

```
<p>This is a paragraph.</p>
<p>This is another paragraph.</p>
```

- **.before():**
  - The before() method inserts specified content in front of (before) the selected elements.
  - **Example:**

```
<script>
$(document).ready(function(){
  $("button").click(function(){
    $("p").before("<p>Hello world!</p>");
  });
});
</script>
<button>Click</button>
<p>This is a paragraph.</p>
 <p>This is another paragraph.</p>
```

- **.addClass():**
  - The addClass() method adds one or more class names to the selected elements.
  - This method does not remove existing class attributes, it only adds one or more class names to the class attribute.
  - **Example:**

```
<script>
$(document).ready(function(){
  $("button").click(function(){
    $("p:first").addClass("intro");
  });
});
</script>
        <style>
```

```
        .intro {
            font-size: 150%;
            color: red;
        }
    </style>
<h1>This is a heading</h1>
<p>This is a paragraph.</p>
<p>This is another paragraph.</p>
<button>Click</button>
```

- **.removeClass():**
  - The removeClass() method removes one or more class names from the selected elements.
  - **Example:**

```
<script>
$(document).ready(function(){
  $("button").click(function(){
    $("p").removeClass("intro");
  });
});
</script>
    <style>
        .intro {
            font-size: 150%;
            color: red;
        }
    </style>
<h1>This is a heading</h1>
<p class="intro">This is a paragraph.</p>
<p class="intro">This is another paragraph.</p>
<button>Remove the "intro" class from all p elements</button>
```

- **.toggleClass():**
  - The toggleClass() method toggles between adding and removing one or more class names from the selected elements.
  - This method checks each element for the specified class names. The class names are added if missing, and removed if already set - This creates a toggle effect.
  - **Example:**

```
<script>
$(document).ready(function(){
  $("button").click(function(){
    $("p:first"). toggleClass(“main");
  });
});
</script>
        <style>
            .main {
                font-size: 150%;
                color: red;
            }
        </style>
<h1>This is a heading</h1>
<p>This is a paragraph.</p>
<p>This is another paragraph.</p>
<button>Click</button>
```

- **.remove():**
  - The remove() method removes the selected elements, including all text and child nodes.
  - This method also removes data and events of the selected elements.
  - **Example:**

```
<script>
```

```
$(document).ready(function(){
  $("button").click(function(){
    $("p").remove();
  });
});
</script>

<p>This is a paragraph.</p>
<p>This is another paragraph.</p>
<button>Remove all p elements</button>
```

- **.empty():**
  - The empty() method removes all child nodes and content from the selected elements.
  - **Example:**

```
<script>
$(document).ready(function(){
  $("button").click(function(){
    $("div").empty();
  });
});
</script>
        <div style="height:100px;background-color:yellow">
                This is some text
                <p>This is a paragraph inside the div.</p>
        </div>
<p>This is a paragraph outside the div.</p>
<button>Remove content of the div element</button>
```

UNIT – 5

(OOP – Object Oriented Programming)

❖ Concept of OOP:

➢ What is OOP ?

- Object Oriented Programming(OOP) is a programming technique that uses objects of data consisting of data fields and methods together with their interactions to design computer programs.
- In object oriented programming that uses objects for main head.
- In programming technique that can be used by the class which defined by user and can be accessed by object as user want.

➢ What is Class ?

- Classes are the data type based on which objects are created.
- Class is a group or block which can stored different type of the data like variables, constants, structures, methods etc.
- A class is thus a collection of objects of similar type. For example, mango, apple, and orange are members of the class fruit.

➢ What is Property?

- Data members declared inside class are called properties. Property is sometimes referred to as attribute or field.
- In PHP, a property is qualified by one of the access specifier keywords, public, private or protected.
- Name of property could be any valid label in PHP. Value of property can be different for each instance of class.
- That's why it is sometimes referred as instance variable.
- Inside any instance method, property can be accessed by calling object's context available as a pesudo-variable $this.
- If a property is declared as public, it is available to object with the help of -> operator.
- If a property is defined with static keyword, its value is shared among all objects of the class and is accessed using scope resolution operator (::) and name of class.
- **Example:**

```
<?php
class myclass{
```

```
  private $fname="Kiran";
  public $mname="Pratap";
  static $lname="Singh";
  function dispdata(){
    echo $this->fname;
    echo $this->mname;
    echo myclass::$lname;
  }
}
$obj=new myclass();
$obj->dispdata();
?>
```

- **Visibility:**
  - In OOP PHP we have three visibility levels for properties and methods of a class: public, protected, and private.
  - Visibility is declared using a visibility keyword to declare what level of visibility a property or method has.
  - The three levels define whether a property or method can be accessed outside of the class, and in classes that extend the class.
  - **Example (public):**

```
<?php
class Fruit {
  public $name;
  protected $color;
  private $weight;
}
$mango = new Fruit();
$mango->name = 'Mango'; // OK
echo $mango->name;
// $mango->color = 'Yellow'; // ERROR
// $mango->weight = '300'; // ERROR
?>
```

- **Constructor:**
  - A constructor allows you to initialize an object's properties upon creation of the object.
  - If you create a __construct() function, PHP will automatically call this function when you create an object from a class.
  - It is called constructor because it construct the values of data member of the class.
  - Notice that the construct function starts with two underscores (__)!
  - We see in the example below, that using a constructor saves us from calling the set_name() method which reduces the amount of code:
  - Example:

```
<?php
class Fruit {
  public $name;
  public $color;
  function __construct($name) {
    $this->name = $name;
  }
  function get_name() {
    return $this->name;
  }
}
$apple = new Fruit("Apple");
echo $apple->get_name();
?>
```

- **Destructor:**
  - A destructor is called when the object is destructed or the script is stopped or exited.

- If you create a __destruct() function, PHP will automatically call this function at the end of the script.
- A destructor never takes any argument nor does it return any value.
- It will be invoked implicitly by the compiler upon exit from the program to clean up storage that is no longer accessible.
- It is a good practice to declare destructor in a program since it releases memory space for future use.
- Notice that the destruct function starts with two underscores (__)!
- The example below has a __construct() function that is automatically called when you create an object from a class, and a __destruct() function that is automatically called at the end of the script:
- Example:

```
<?php
class Fruit {
  public $name;
  public $color;
  function __construct($name) {
    $this->name = $name;
  }
  function __destruct() {
    echo "The fruit is {$this->name}.";
  }
}
$apple = new Fruit("Apple");
?>
```

- **Inheritance:**
  - Inheritance in OOP = When a class derives from another class.
  - The class reusability is called as inheritance.

- The child class will inherit all the public and protected properties and methods from the parent class. In addition, it can have its own properties and methods.
- An inherited class is defined by using the extends keyword.
- The common features we defined in a class called base which is used in all the derived class.
- This type is worked as parent and child class concept.
- Example:

```
<?php
class Fruit {
  public $name;
  public $color;
  public function __construct($name, $color) {
    $this->name = $name;
    $this->color = $color;
  }
  public function intro() {
    echo "The fruit is {$this->name} and the color is {$this->color}.";
  }
}
// Strawberry is inherited from Fruit
class Strawberry extends Fruit {
  public function message() {
    echo "Am I a fruit or a berry? ";
  }
}
$strawberry = new Strawberry("Strawberry", "red");
$strawberry->message();
$strawberry->intro();
?>
```

- **Define a Base class:**
  - The parent class is also called a base class or super class.

- The general form of defining a base class is:
- Syntax:

```
class baseClassName{
        ..... ..... // members of base class .....
}
```

- Example:

```
class Fruits
{
        //base class members here
}
```

➢ **Define a Derived class:**

- A derived class can be defined by specifying its relationship with the base class in addition to its own details.
- The general form of defining a derived class is:
- Syntax:

```
class derivedClassName extends baseClassName{
        ..... ..... // members of derived class .....
}
```

- Example:

```
class Apple extends Fruits{
        //derived class members here
}
```

➢ **Class Constants:**

- Constants cannot be changed once it is declared.
- Class constants can be useful if you need to define some constant data within a class.
- A class constant is declared inside a class with the const keyword.
- Class constants are case-sensitive. However, it is recommended to name the constants in all uppercase letters.

- We can access a constant from outside the class by using the class name followed by the scope resolution operator (::) followed by the constant name, like here:
- Example:

```
<?php
class Goodbye {
    const MESSAGE = "Thank you for visiting www.cdmi.in!";
}
echo Goodbye::MESSAGE;
?>
```

- **Scope Resolution Operator(::) :**
  - In PHP, the double colon :: is defined as Scope Resolution Operator.
  - It used when we want to access constants, properties and methods defined at class level.
  - When referring to these items outside class definition, name of class is used along with scope resolution operator.
  - This operator is also called Paamayim Nekudotayim, which in Hebrew means double colon.
  - Example:
  -

```
<?php
class A{
  const PI=3.142;
  static $x=10;
}
echo A::PI;
echo A::$x;
$var='A';
echo $var::PI;
echo $var::$x;
?>
```

- **Autoloading Classes:**
  - Autoloading allows you to automatically load classes using an autoloader so you don't have to manually include the files containing the class definitions.
  - whenever you need to use a class, you need to do the following:
    - require_once ('/path/to/classes/first.php');
    - require_once ('/path/to/classes/second.php');
    - require_once ('/path/to/classes/third.php');

    ```
    $First = new First();
    $Second = new Second();
    $Third = new Third();
    ```
  - With autoloading, you can use an autoloader instead of multiple require_once() statements.
  - It also eliminates the need to update these require statements whenever you add, rename, or change the location of your classes. That's a big plus for maintainability.
  - Example:

```
//create file first.php
class First{
 function test(){
    echo "this is test function from first.php file";
 }
}
//create another file autoload_example.php
spl_autoload_register(function ($class_name) {
  include $class_name . '.php';
});
$obj = new First();
$obj->test();
```

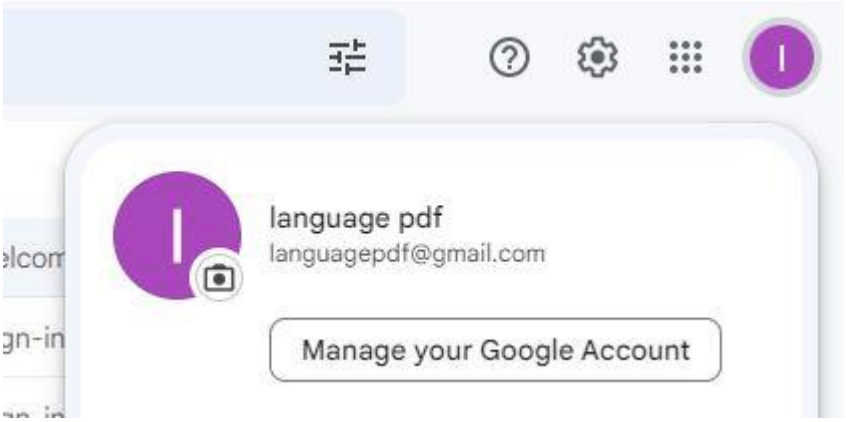
- **Mysql Database handling with OOP:**

```
<?php
class DBExample{
  var $con;
  function __construct($host,$user,$pass,$dbname){
    $this->con = mysqli_connect($host,$user,$pass,$dbname);
  }
  function query($sql,$type){
    switch($type){
      case 1:
        $res = mysqli_query($this->con,$sql) or
die(mysqli_error($this->con));
        return mysqli_insert_id($this->con);
      case 2:
        $res = mysqli_query($this->con,$sql) or
die(mysqli_error($this->con));
        return mysqli_affected_rows($this->con);
      case 3:
        $res = mysqli_query($this->con,$sql) or
die(mysqli_error($this->con));
        $row = mysqli_fetch_assoc($res);
        return $row;
      case 4:
        $res = mysqli_query($this->con,$sql) or
die(mysqli_error($this->con));
        $rows = array();
        while($row = mysqli_fetch_assoc($res)){
          $rows[] = $row;
        }
        return $rows;
    }
  }
```

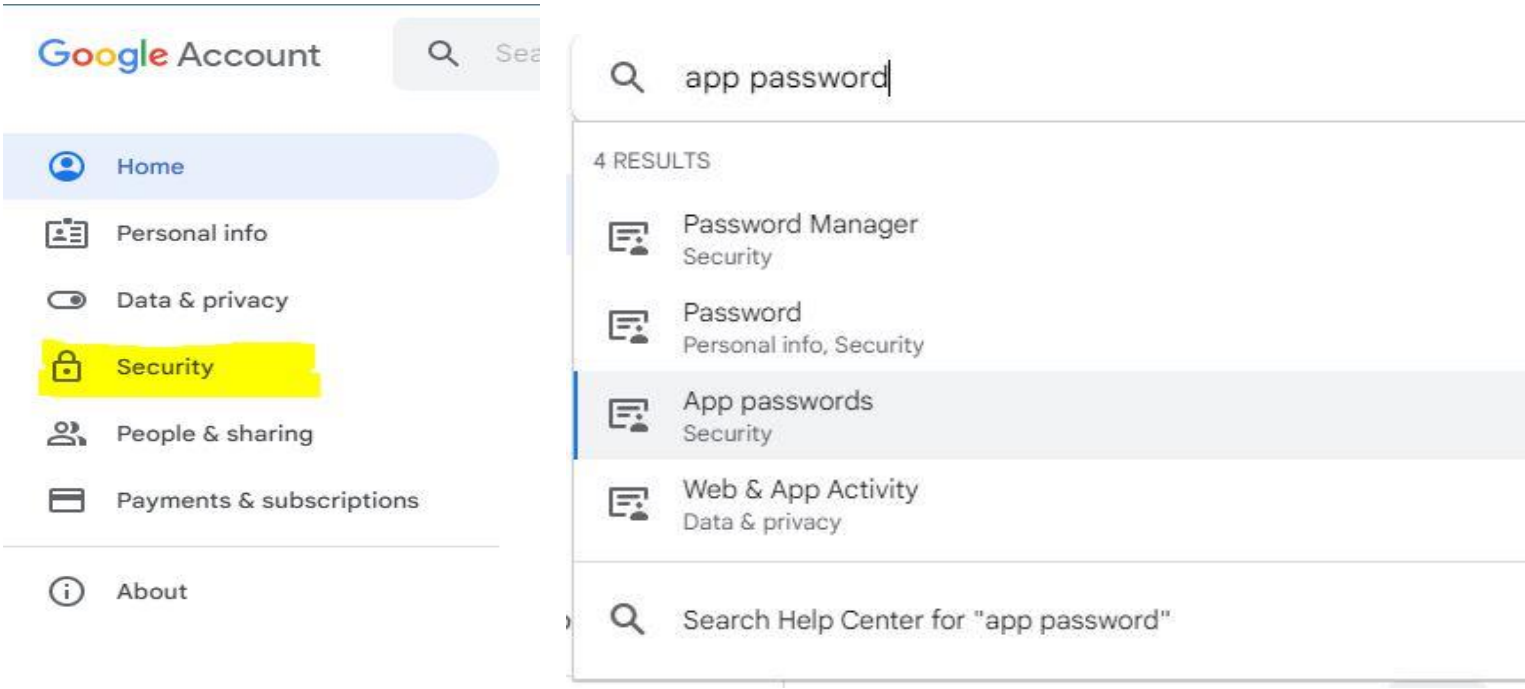
```
}
$obj = new DBExample("localhost","root","","school");
//insert
//$sql = "insert into students (name,email,contact,gender,city, age)
values
('mahesh','mahesh@gmail.com','9876543210','male','surat',23)";
// echo $obj->query($sql,1);

//update
// $sql = "update students set name='mahesh
patel',email='mahesh123@gmail.com' where id=10";
// echo $obj->query($sql,2);

//delete
// $sql = "delete from students where id=10";
// echo $obj->query($sql,2);

//select one
// $sql = "select * from students where id=9";
// echo '<pre>';
// $output = $obj->query($sql,3);
// print_r($output);

//select all
// $sql = "select * from students";
// echo '<pre>';
// $output = $obj->query($sql,4);
// print_r($output);
?>
```

Ch-2 (mail-sending)

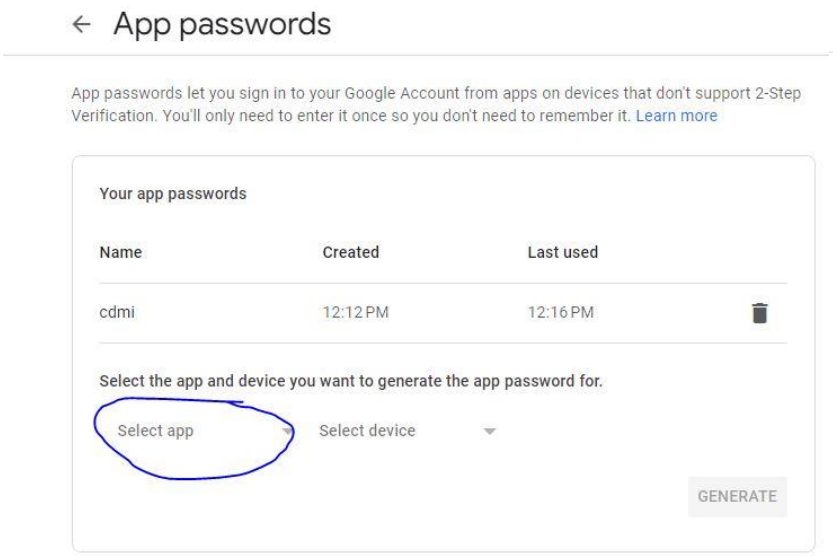
1. Download phpmailer (phpmailer release) in version 5.2.24.
2. Click on Source code and download Source code.
3. Extract zip file and copy code in your server folder.
4. Select smtp.phps file and rename it smtp.php
5. Set gmail configuration in smtp.php
    1. Open smtp.php file and set Host name : smtp.gmail.com
    2. Set port number 25 , 465 , 587
    3. Set username: your email address
    4. Set password: set app password
    5. Set setfrom : your email address
    6. Set addReplyTo: your email address
    7. Set addAddress: set receiver email address
    8. Set msgHtml : with any text
    9. Set addAttachment: with any image or pdf filel to be send
5. gmail account configuration
    a. goto your gmail account and select account management

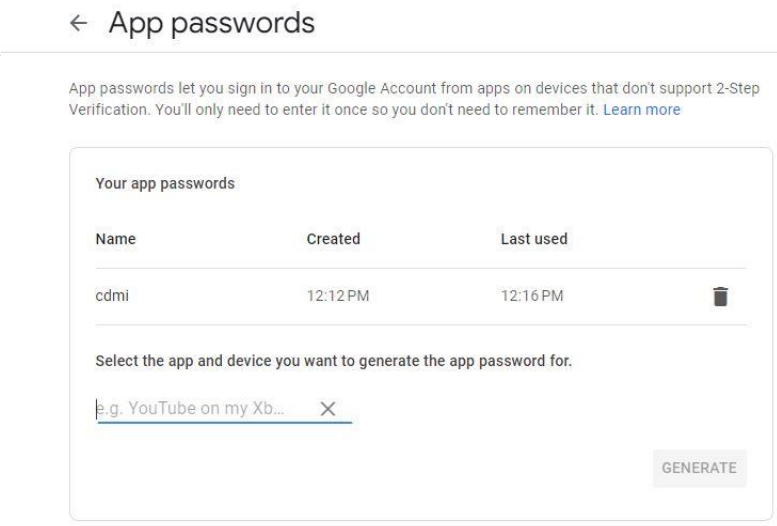


b. select security and goto search-bar and search **app-password**



c. select app password and select app with other and assign name and click to generate.





d. Copy password and past to app password in smtp.php file then run it