Client-side Scripting

Client-side scripting is performed to generate a code that can run on the client end (browser) without needing the server-side processing. Basically, these types of scripts are placed inside an HTML document. The client-side scripting can be used to examine the user's form for the errors before submitting it and for changing the content according to the user input. As I mentioned before, the web requires three elements for its functioning which are, client, database and server.

The effective client-side scripting can significantly reduce the **server load**. It is designed to run as a scripting language utilizing a web browser as a host program. For example, when a user makes a request via browser for a webpage to the server, it just sent the HTML and CSS as plain text, and the browser interprets and renders the web content in the client end. It can also used to create "cookies" that store data on user's computer.

Client-side scripting languages:

HTML

CSS

JavaScript

Server-side Scripting

Server-side scripting is a technique of programming for producing the code which can run software on the server side, in simple words any scripting or programming that can run on the web server is known as server-side scripting. The operations like customization of a website, dynamic change in the website content, response generation to the user's queries, accessing the database, and so on are performed at the server end.

The server-side scripting constructs a communication link between a server and a client (user). Earlier the server-side scripting is implemented by the **CGI (Common Gateway Interface)** scripts. The CGI was devised to execute the scripts from programming languages such as PHP or Perl on the websites.

The server-side involves three parts: server, database, API's and back-end web software developed by the server-side scripting language. When a browser sends a request to the server for a webpage consisting of server-side scripting, the web

server processes the script prior to serving the page to the browser. Here the processing of a script could include extracting information from a database, making simple calculations, or choosing the appropriate content that is to be displayed in the client end. The script is being processed and the output is sent to the browser. The web server abstracts the scripts from the end user until serving the content, which makes the data and source code more secure.

Server-side scripting languages:

PHP

Python

Ruby

Difference between Server-Side Scripting and Client-Side Scripting

Client-side scripting	Server-side scripting
Source code is visible to user.	Source code is not visible to user because it's output of server side is a HTML page.
It usually depends on browser and its version.	In this any server-side technology can be use and it does not depend on client.
It runs on user's computer.	It runs on web server.
There are many advantages link with this like faster. response times, a more interactive application.	The primary advantage is its ability to highly customize, response requirements, access rights based on user.
It does not provide security for data.	It provides more security for data.
It is a technique use in web development in which scripts runs on client's browser.	It is a technique that uses scripts on web server to produce a response that is customized for each client's request.
HTML, CSS and JavaScript are used.	PHP, Python, Java, Ruby are used.

PHP Introduction

PHP stands for **Hypertext Pre-Processor,** that earlier stood for Personal Home Pages. PHP is a server-side scripting language designed specifically for web development. PHP is a loosely typed language; it does not have explicit defined data types. It is open-source which means it is free to download and use. It is very simple to learn and use. The files have the extension ".php".

- PHP is an Interpreted language, hence it doesn't need a compiler.
- To run and execute PHP code, we need a Web server on which PHP must be installed.
- PHP is a server-side scripting language, which means that PHP is executed on the server and the result is sent to the browser in plain HTML.
- PHP is open source and free.
- It can be integrated with many databases such as Oracle, Microsoft SQL Server, MySQL, PostgreSQL, Sybase, Informix.
- It is powerful to hold a content management system like WordPress and can be used to control user access.
- It supports main protocols like HTTP Basic, HTTP Digest, IMAP, FTP, and others.

Characteristics of PHP are -

- Simple and fast
- Efficient
- Secured
- Flexible
- Cross-platform, it works with major operating systems like Windows, Linux, MacOS.

Syntax:

```
<?php

  PHP code goes here

?>
```

PHP Data Types

Data Types define the type of data a variable can store. PHP allows eight different types of data types. All of them are discussed below. There are pre-defined, user-defined, and special data types.

The predefined data types are:

- Boolean
- Integer
- Double
- String

The user-defined (compound) data types are:

- Array
- Objects

The special data types are:

- NULL
- resource

1. PHP Boolean

A Boolean data type can have two possible values, either **True** or **False**.

$a = true;
$b = false;

NOTE: Here the values true and false are not enclosed within quotes, because these are not strings.

2. PHP Integer

An Integer data type is used to store any non-decimal numeric value within the range -2,147,483,648 to 2,147,483,647.

An integer value can be negative or positive, but it cannot have a decimal.

```
$x = -2671;
$y = 7007;
```

3. PHP Float

Float data type is used to store any decimal numeric value. A float (floating point) value can also be either negative or positive.

```
$a = -2671.01;
$b = 7007.70;
```

4. PHP String

String data type in PHP and in general, is a sequence of characters (or anything, it can be numbers and special characters too) enclosed within quotes. You can use single or double quotes.

```
$str1 = "Hello";
$str2 = "What is your Roll No?";
$str3 = "4";
```

5. PHP NULL

NULL data type is a special data type which means **nothing**. It can only have one value, and that is NULL. If you create any variable and do not assign any value to it, it will automatically have NULL stored in it.

Also, we can use NULL value to empty any variable.

```
// holds a null value
$a;
$b = 7007.70;
// we can also assign null value
$b = null;
```

6. PHP ARRAY

Array is a compound data-type that can store multiple values of the same data type. Below is an example of an array of integers. It combines series of data that are related together.

```
$intArray = array (10, 20, 30);

echo "First Element: $intArray[0]\n";
echo "Second Element: $intArray[1]\n";
```

7. PHP OBJECT

Objects are defined as instances of user-defined classes that can hold both values and functions and information for data processing specific to the class. This is an advanced topic and will be discussed in detail in further articles. When the objects are created, they inherit all the properties and behaviors from the class, having different values for all the properties.

Objects are explicitly declared and created from the *new* keyword.

8. PHP RESOURCE

Resources in PHP are not an exact data type. These are basically used to store references to some function call or to external PHP resources. For example, consider a database call. This is an external resource. Resource variables hold special handles to files and database connections.

PHP Variables

Variables in a program are used to store some values or data that can be used later in a program. The variables are also like containers that store character values, numeric values, memory addresses, and strings. PHP has its own way of declaring and storing variables.

There are few rules, that needs to be followed and facts that need to be kept in mind while dealing with variables in PHP:

- Any variables declared in PHP must begin with a dollar sign (**$**), followed by the variable name.

- A variable can have long descriptive names (like $factorial, $even_nos) or short names (like $n or $f or $x)
- A variable name can only contain alphanumeric characters and underscores (i.e., ‘a-z’, ‘A-Z’, ‘0-9 and ‘_’) in their name. Even it cannot start with a number.
- A constant is used as a variable for a simple value that cannot be changed. It is also case-sensitive.
- Assignment of variables is done with the assignment operator, “equal to (=)”. The variable names are on the left of equal and the expression or values are to the right of the assignment operator ‘=’.
- One must keep in mind that variable names in PHP names must start with a letter or underscore and no numbers.

Variable Scopes

Scope of a variable is defined as its extent in program within which it can be accessed, i.e., the scope of a variable is the portion of the program within which it is visible or can be accessed.

Local variables: The variables declared within a function are called local variables to that function and has its scope only in that particular function. In simple words, it cannot be accessed outside that function. Any declaration of a variable outside the function with same name as that of the one within the function is a complete different variable.

```
function local_var()
{
   $num = 50;
   echo "local num = $num \n";
}
```

Global variables: The variables declared outside a function are called global variables. These variables can be accessed directly outside a function. To get access within a function we need to use the “global” keyword before the variable to refer to the global variable.

```
$num = 20;
function global_var()
{
   global $num;
```

```
    echo "Variable num inside function : $num \n";
}
```

Static variable: It is the characteristic of PHP to delete the variable, ones it completes its execution and the memory is freed. But sometimes we need to store the variables even after the completion of function execution. To do this we use static keyword and the variables are then called as static variables. PHP associates a data type depending on the value for the variable.

```
function static_var()
{
    // static variable
    static $num = 5;
    echo $num, "\n";
}
```

PHP Operators

Arithmetic operators

Arithmetic operators are used to perform arithmetic operations on numeric data. The concatenate operator works on strings values too. PHP supports the following operators.

Operator	Name	Description	Example	Output
+	Addition	Summation of x and y	1 + 1;	2
-	Subtraction	Difference between x and y	1 – 1;	0
*	Multiplication	Multiplies x and y	3 * 7;	21
/	Division	Quotient of x and y	45 / 5;	9
%	Php Modulus	Gives reminder of diving x and y	10 % 3;	1
-n	Negation	Turns n into a negative number	-(-5);	5

Operator	Name	Description	Example	Output
x . y	Concatenation	Puts together x and y	"PHP" . " ROCKS";10 . 3;	PHP ROCKS103

Assignment Operators

Assignment operators are used to assign values to variables. They can also be used together with arithmetic operators.

Operator	Name	Description	Example	Output
x = ?	assignment	Assigns the value of x to ?	$x = 5;	5
x += ?	addition	Increments the value of x by ?	$x = 2;$x += 1;	3
X -= ?	subtraction	Subtracts ? from the value of x	$x = 3;$x -= 2;	1
X *=?	multiplication	Multiplies the value of x ? times	$x = 0;$x *=9;	0
X /=?	division	Quotient of x and ?	$x = 6;$x /=3;	2
X %=?	modulus	The reminder of dividing x by?	$x = 3;$x %= 2;	1
X .=?	concatenate	Puts together items	" $x = ‘Pretty’;$x .= ‘ Cool!’;"	Pretty Cool!

Comparison operators

Comparison operators are used to compare values and data types.

Operator	Name	Description	Example	Output
X == y	Equal	Compares x and y then returns true if they are equal	1 == "1";	True or 1
X === y	identical	Compares both values and data types.	1 === "1";	False or 0. Since 1 is integer and “1” is string
X != y, x <> y	PHP Not equal	Compares values of x and y. returns true if the values are not equal	2 != 1;	True or 1
X > y	Greater than	Compares values of x and y. returns true if x is greater than y	3 > 1;	True or 1
X < y	Less than	Compares values of x and y. returns true if x is less than y	2 < 1;	False or 0
X >= y	Greater than or equal	Compares values of x and y. returns true if x is greater than or equal to y	1 >=1	True or 1
X <= y	Less than or equal	Compares values of x and y. returns true if x is greater than or equal to y	8 <= 6	False or 0

Logical operators

When working with logical operators, any number greater than or less than zero (0) evaluates to true. Zero (0) evaluates to false.

Operator	Name	Description	Example	Output
X and y, x && y	And	Returns true if both x and y are equal	1 and 4;True&& False;	True or 1False or 0

Operator	Name	Description	Example	Output
X or y, x || y	Or	Returns true if either x or y is true	6 or 9;0 || 0;	True or 1False or 0
X xor y	Exclusive or, xor	Returns true if only x is true or only y is true	1 xor 1;1 xor 0;	False or 0True or 1
!x	Not	Returns true if x is false and false if x is true	!0;	True or 1

PHP Constants

Constants are either identifiers or simple names that can be assigned any fixed values. They are similar to a variable except that they can never be changed. They remain constant throughout the program and cannot be altered during execution. Once a constant is defined, it cannot be undefined or redefined.

there are two ways to define a constant:

1. Using the define() method.
2. Using the const keyword.

Using define()

The define() function in PHP is used to create a constant as shown below:
Syntax:

define(*name, value, case_insensitive*)

The parameters are as follows:

- *name:* The name of the constant.
- *value:* The value to be stored in the constant.
- *case_insensitive:* Defines whether a constant is case insensitive. By default this value is False, i.e., case sensitive.

Using const Keyword

e can also define constants in PHP using the const keyword. But we can only use the const keyword to define scalar constants, i.e. only integers, floats, booleans and

strings, while define() can be used to define array and resource constants as well, although they are not used oftenly.

```
<?php
   const PI = 3.14;
   echo PI;
?>
```

PHP echo and print Statements

echo and print are more or less the same. They are both used to output data to the screen.

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

1 - echo

- echo is a statement, which is used to display the output.
- echo can be used with or without parentheses.
- echo does not return any value.
- We can pass multiple strings separated by comma (,) in echo.
- echo is faster than print statement.

```
<?php
echo "Multiple ","argument ","string!";
$num1 = 10;
$num2 = 20;
echo $num1."+".$num2."=";
echo $num1 + $num2;
?>
```

2 - print

- print is also a statement, used as an alternative to echo at many times to display the output.
- print can be used with or without parentheses.
- print always returns an integer value, which is 1.

- Using print, we cannot pass multiple arguments.
- print is slower than echo statement.

```
<?php
print "Hello, world!";
?>
```

PHP Decision Making

PHP allows us to perform actions based on some type of conditions that may be logical or comparative. Based on the result of these conditions i.e., either TRUE or FALSE, an action would be performed as asked by the user. It's just like a two-way path.

1. **if Statement**: The *if* statement is used to execute a block of code only if the specified condition evaluates to true.

Syntax :

```
if (condition){
   // if TRUE then execute this code
}
```

2. **if…else Statement**: You can enhance the decision making process by providing an alternative choice through adding an *else* statement to the *if* 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.

Syntax:

```
if (condition) {
   // if TRUE then execute this code
}
else{
   // if FALSE then execute this code
}
```

3. **if…elseif…else Statement**: This allows us to use multiple if…else statments. We use this when there are multiple conditions of TRUE cases.

Syntax:

```
if (condition) {
   // if TRUE then execute this code
}
elseif {
   // if TRUE then execute this code
}
elseif {
   // if TRUE then execute this code
}
else {
   // if FALSE then execute this code
}
```

4. **switch Statement**: The "switch" performs in various cases i.e., it has various cases to which it matches the condition and appropriately executes a particular case block. It first evaluates an expression and then compares with the values of each case. If a case matches then the same case is executed. To use switch, we need to get familiar with two different keywords namely, **break** and **default**.

1. The **break** statement is used to stop the automatic control flow into the next cases and exit from the switch case.
2. The **default** statement contains the code that would execute if none of the cases match.

Syntax:

```
switch(n) {
          case statement1:
           code to be executed if n==statement1;
            break;
        case statement2:
           code to be executed if n==statement2;
            break;
          .....
    default:
      code to be executed if n != any case;
```

PHP Loops

A Loop is an Iterative Control Structure that involves executing the same number of code a number of times until a certain condition is met.

1 - for loop: This type of loops is used when the user knows in advance, how many times the block needs to execute. That is, the number of iterations is known beforehand. These type of loops are also known as entry-controlled loops. There are three main parameters to the code, namely the initialization, the test condition and the counter.

Syntax:

```
for (initialize; condition; increment){

//code to be executed

}
```

- "**initialize**" usually an integer; it is used to set the counter's initial value.
- **"condition"** the condition that is evaluated for each php execution. If it evaluates to true then execution of the for... loop is terminated. If it evaluates to false, the execution of the for... loop continues.
- **"increment"** is used to increment the initial value of counter integer.

2 - while loop: The while loop is also an entry control loop like for loops i.e., it first checks the condition at the start of the loop and if its true then it enters the loop and executes the block of statements, and goes on executing it as long as the condition holds true.

Syntax:

```
while (if the condition is true) {
   // code is executed
}
```

3 - do-while loop: This is an exit control loop which means that it first enters the loop, executes the statements, and then checks the condition. Therefore, a statement is executed at least once on using the do…while loop. After executing once, the program is executed as long as the condition holds true.

Syntax:

```
do {

    //code is executed

} while (if condition is true);
```

4 - foreach loop: This loop is used to iterate over arrays. For every counter of loop, an array element is assigned and the next counter is shifted to the next element.

Syntax:

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

- **"$array_data"** is the array variable to be looped through
- **"$array_value "** is the temporary variable that holds the current array item values.

```
<?php
$animals_list = array("Lion","Wolf","Dog","Leopard","Tiger");
foreach($animals_list as $array_values){
            echo $array_values . "<br>";
}
?>
```

PHP Strings

A string is a sequence of letters, numbers, special characters and arithmetic values or combination of all. The simplest way to create a string is to enclose the string literal (i.e. string characters) in single quotation marks (') and double quotation marks (“).

Single and double quotation marks work in different ways. Strings enclosed in single-quotes are treated almost literally, whereas the strings delimited by the

double quotes replaces variables with the string representations of their values as well as specially interpreting certain escape sequences.

The escape-sequence replacements are:

- \n is replaced by the newline character
- \r is replaced by the carriage-return character
- \t is replaced by the tab character
- \$ is replaced by the dollar sign itself ($)
- \" is replaced by a single double-quote (")
- \\ is replaced by a single backslash (\)

Syntax:

```
<?php
$my_str = 'World';
echo "Hello, $my_str!<br>";      // Displays: Hello World!
echo 'Hello, $my_str!<br>';      // Displays: Hello, $my_str!
?>
```

PHP Strings Function

1. strlen($str)

This function returns the length of the string or the number of characters in the string including whitespaces.

```
<?php

$str = "Welcome to php";
echo "Length of the string is: ". strlen($str);

?>
```

2. str_word_count($str)

This function returns the number of words in the string. This function comes in handly in form field validation for some simple validations.

```
<?php

$str = "Welcome to php";
echo "Number of words in the string are: ". str_word_count($str);

?>
```

3. strrev($str)

This function is used to reverse a string.

```
<?php

$str = "Welcome to php";
echo "Reverse: ". strrev($str);

?>
```

4. strpos($str, $text)

This function is used to find the position of any text/word in a given string. Just like an array, string also assign index value to the characters stored in it, starting from zero.

```
<?php

$str = "Welcome to php";
echo "Position of 'php' in string: ". strpos($str, 'php');

?>
```

5. strtoupper($str)

To convert every letter/character of every word of the string to uppercase, one can use strtoupper() method.

```
<?php

$str = "welcome to php";
```

```
echo strtoupper($str);

?>
```

6. strtolower($str)

This function is used to convert every letter/character of a string to lowercase.

```
<?php

$str = "WELCOME TO STUDYTONIGHT";

echo strtolower($str);

?>
```

7. substr($str, $start, $length)

This function is used to take out a part of the string(substring), starting from a particular position, of a particular length.

The first argument is the string itself, second argument is the starting index of the substring to be exracted and the third argument is the length of the substring to be extracted.

```
<?php

$str = "Welcome to php";

echo substr($str, 11, 12);

?>
```

PHP Arrays

Arrays are complex variables that allow us to store more than one value or a group of values under a single variable name. The arrays are helpful to create a list of

elements of similar types, which can be accessed using their index or key. An array is created using an **array()** function in PHP.

Types of Arrays in PHP

There are three types of arrays that you can create. These are:

- **Indexed array** — An array with a numeric key.
- **Associative array** — An array where each key has its own specific value.
- **Multidimensional array** — An array containing one or more arrays within itself.

Indexed Arrays

An indexed or numeric array stores each array element with a numeric index. All PHP array elements are assigned to an index number by default.

```
<?php
// Define an indexed array
$colors = array("Red", "Green", "Blue");
$colors[0] = "Red";
$colors[1] = "Green";
?>
```

Associative Arrays

These types of arrays are similar to the indexed arrays but instead of linear storage, every value can be assigned with a user-defined key of string type.

```
<?php
// Define an associative array
$ages = array("Peter"=>22, "Clark"=>32, "John"=>28);
$ages["Peter"] = "22";
$ages["Clark"] = "32";
?>
```

Multidimensional Arrays

Multi-dimensional arrays are such arrays that store another array at each index instead of a single element. In other words, we can define multi-dimensional arrays

as an array of arrays, every element in this array can be an array and they can also hold other sub-arrays within. Arrays or sub-arrays in multidimensional arrays can be accessed using multiple dimensions.

```
<?php
// Define a multidimensional array
$contacts = array(
    array(
        "name" => "Peter Parker",
        "email" => "peterparker@mail.com",
    ),
    array(
        "name" => "Clark Kent",
        "email" => "clarkkent@mail.com",
    )
);
// Access nested value
echo "Peter Parker's Email-id is: " . $contacts[0]["email"];
?>
```

1. print_r() Function (imp)

The print_r() function is a built-in function in PHP and is used to print or display information stored in a variable.

Syntax:

print_r($variable)

2. sizeof($arr)

This function returns the size of the array or the number of data elements stored in the array.

```
<?php
$lamborghinis = array("Urus", "Huracan", "Aventador");
echo "Size of the array is: ". sizeof($lamborghinis);
?>
```

3. array_merge($arr1, $arr2)

If you want to combine two different arrays into a single array, you can do so using this function. It doesn't matter whether the arrays to be combined are of same type(indexed, associative etc) or different types, using this function we can combine them into one single array.

```
<?php

$hatchbacks = array(
        "Suzuki" => "Baleno",
        "Skoda" => "Fabia",
        "Hyundai" => "i20",
        "Tata" => "Tigor"
    );
$friends = array("Vinod", "Javed", "Navjot", "Samuel");
$merged = array_merge($hatchbacks, $friends);

print_r($merged);

?>
```

4. array_pop($arr)

This function removes the last element of the array. Hence it can be used to remove one element from the end.

```
<?php

$lamborghinis = array("Urus", "Huracan", "Aventador");
array_pop($lamborghinis);
print_r($lamborghinis);

?>
```

5. array_push($arr, $val)

This function is the opposite of the array_pop() function. This can be used to add a new element at the end of the array.

```
<?php

$lamborghinis = array("Urus", "Huracan", "Aventador");
array_push($lamborghinis, "Estoque");
print_r($lamborghinis);

?>
```

PHP Functions

A **Function in PHP** is a reusable piece or block of code that performs a specific action. It takes input from the user in the form of parameters, performs certain actions, and gives the output. Functions can either return values when called or can simply perform an operation without returning any value.

Advantage of PHP Functions

- Better code organization – PHP functions allow us to group blocks of related code that perform a specific task together.
- Reusability – once defined, a function can be called by a number of scripts in our PHP files. This saves us time of reinventing the wheel when we want to perform some routine tasks such as connecting to the database
- Easy maintenance - updates to the system only need to be made in one place.
- **Easy to understand** - PHP functions separate the programming logic. So it is easier to understand the flow of the application because every logic is divided in the form of functions.

PHP Built-in Functions

A function is a self-contained block of code that performs a specific task.

PHP has a huge collection of internal or built-in functions that you can call directly within your PHP scripts to perform a specific task, like gettype(), print_r(), var_dump, etc.

PHP User-Defined Functions

PHP also allows you to define your own functions. It is a way to create reusable code packages that perform specific tasks and can be kept and maintained separately form main program.

Creating and Invoking Functions

```
function functionName(){
   // Code to be executed
}
```

The declaration of a user-defined function start with the word function, followed by the name of the function you want to create followed by parentheses i.e. () and finally place your function's code between curly brackets {}.

PHP Function Arguments

We can pass the information in PHP function through arguments which is separated by comma.

PHP supports **Call by Value** (default), **Call by Reference**, **Default argument values** and **Variable-length argument list**.

```
function myFunc($oneParameter, $anotherParameter){
   // Code to be executed
}
```

You can define as many parameters as you like. However for each parameter you specify, a corresponding argument needs to be passed to the function when it is called.

PHP Function: Returning Value

A function can return a value back to the script that called the function using the return statement. The value may be of any type, including arrays and objects.

```
<?php
function getSum($num1, $num2){
   $total = $num1 + $num2;
   return $total;
}
echo getSum(5, 10);
?>
```

A function can not return multiple values.

PHP Regular Expressions

Regular Expressions, commonly known as "**regex**" or "**RegExp**", are a specially formatted text strings used to find patterns in text. Regular expressions are one of the most powerful tools available today for effective and efficient text processing and manipulations. For example, it can be used to verify whether the format of data i.e. name, email, phone number, etc. entered by the user was correct or not, find or replace matching string within text content, and so on.

Advantages and uses of Regular expressions:

- Regular expressions help in validation of text strings which are of programmer's interest.
- It offers a powerful tool for analyzing, searching a pattern and modifying the text data.
- It helps in searching specific string pattern and extracting matching results in a flexible manner.
- It helps in parsing text files looking for a defined sequence of characters for further analysis or data manipulation.
- With the help of in-built regexes functions, easy and simple solutions are provided for identifying patterns.
- It effectively saves a lot of development time, which are in search of specific string pattern.

Operators in Regular Expression: Let us look into some of the operators in PHP regular expressions.

Operator	Description
^	It denotes the start of string.
$	It denotes the end of string.
.	It denotes almost any single character.
()	It denotes a group of expressions.
[]	It finds a range of characters for example [xyz] means x, y or z .
[^]	It finds the items which are not in range for example [^abc] means NOT a, b or c.
– (dash)	It finds for character range within the given item range for example [a-z] means a through z.
| (pipe)	It is the logical OR for example x | y means x OR y.

?	It denotes zero or one of preceding character or item range.
*	It denotes zero or more of preceding character or item range.
+	It denotes one or more of preceding character or item range.
{n}	It denotes exactly n times of preceding character or item range for example n{2}.
{n, }	It denotes atleast n times of preceding character or item range for example n{2, }.
{n, m}	It denotes atleast n but not more than m times for example n{2, 4} means 2 to 4 of n.
\	It denotes the escape character.

PHP Include and Require Function

include() Function in PHP

To include a file using the include() function, you simply call the function (as you would any other function) and insert the file path as a parameter.

Syntax: include ('fileName');

require() Function in PHP

the require() function is the same as the include() function - simply call the function and pass the path of the include file.

Syntax: require ('fileName');

The only difference is — the include() statement will only generate a PHP warning but allow script execution to continue if the file to be included can't be found, whereas the require() statement will generate a fatal error and stops the script execution.

The include_once and require_once Statements

If you accidentally include the same file (typically functions or classes files) more than one time within your code using the include or require statements, it may

cause conflicts. To prevent this situation, PHP provides include_once and require_once statements. These statements behave in the same way as include and require statements with one exception.

The include_once and require_once statements will only include the file once even if asked to include it a second time i.e. if the specified file has already been included in a previous statement, the file is not included again.

PHP Superglobals Variables

These are specially-defined array variables in PHP that make it easy for you to get information about a request or its context. The superglobals are available throughout your script. These variables can be accessed from any function, class or any file without doing any special task such as declaring any global variable etc. They are mainly used to store and get information from one page to another etc in an application.

Below is the list of superglobal variables available in PHP:

1. $GLOBALS
2. $_SERVER
3. $_REQUEST
4. $_GET
5. $_POST
6. $_SESSION
7. $_COOKIE
8. $_FILES
9. $_ENV

The **$GLOBALS** Variable

It is a superglobal variable which is used to access global variables from anywhere in the PHP script. PHP stores all the global variables in array $GLOBALS[] where index holds the global variable name, which can be accessed.

```
<?php
$x = 300;
$y = 200;
function multiplication(){
   $GLOBALS['z'] = $GLOBALS['x'] * $GLOBALS['y']; }
multiplication();
echo $z;       ?>
```

The **$_SERVER** Variable

It is a PHP super global variable that stores the information about headers, paths and script locations. Some of these elements are used to get the information from the superglobal variable $_SERVER.

```
<?php
echo $_SERVER['PHP_SELF'];
echo $_SERVER['SERVER_NAME'];
echo $_SERVER['HTTP_HOST'];
echo $_SERVER['HTTP_USER_AGENT'];
?>
```

The **$_POST** Variable

It is a super global variable used to collect data from the HTML form after submitting it. When form uses method post to transfer data, the data is not visible in the query string, because of which security levels are maintained in this method.

The **$_GET** Variable

$_GET is a super global variable used to collect data from the HTML form after submitting it. When form uses method get to transfer data, the data is visible in the query string, therefore the values are not hidden. $_GET super global array variable stores the values that come in the URL.

The $_REQUEST Variable

PHP provides another superglobal variable $_REQUEST that contains the values of both the $_GET and $_POST variables as well as the values of the $_COOKIE superglobal variable.

PHP header() Function

The header() function is an inbuilt function in PHP which is used to send a raw HTTP header. The HTTP functions are those functions which manipulate information sent to the client or browser by the Web server, before any other

output has been sent. The PHP header() function send a HTTP header to a client or browser in raw form. Before HTML, XML, JSON or other output has been sent to a browser or client, a raw data is sent with request (especially HTTP Request) made by the server as header information. HTTP header provide required information about the object sent in the message body more precisely about the request and response.

Syntax:

void header($header, $replace = TRUE, $http_response_code)

- **$header:** This parameter hold the header string. There are two types of header calls. The first header starts with string “HTTP/”, which is used to figure out the HTTP status code to send. The second case of header is the “Location:”. It is mandatory parameter.
- **$replace:** It is optional parameter. It denotes the header should replace previous or add a second header. The default value is True (will replace). If $replace value is False then it force multiple headers of the same type.
- **$http_response_code:** It is an optional parameter. It forces the HTTP response code to the specified value (PHP 4.3 and higher).

```
<?php

// Redirect the browser
header("Location: http://www.geeksforgeeks.org");

exit;

?>
```

PHP File Upload

The global predefined variable **$_FILES** is an associative array containing items uploaded via HTTP POST method. Uploading a file requires HTTP POST method form with enctype attribute set to **multipart/form-data**.

- $_FILES["file"]["name"] — This array value specifies the original name of the file, including the file extension. It doesn't include the file path.

- $_FILES["file"]["type"] — This array value specifies the MIME type of the file.
- $_FILES["file "]["size"] — This array value specifies the file size, in bytes.
- $_FILES["file "]["tmp_name"] — This array value specifies the temporary name including full path that is assigned to the file once it has been uploaded to the server.
- $_FILES["file "]["error"] — This array value specifies error or status code associated with the file upload, e.g. it will be 0, if there is no error.

Once a file has been successfully uploaded, it is automatically stored in a temporary directory on the server. To store this file on a permanent basis, you need to move it from the temporary directory to a permanent location using the PHP's move_uploaded_file() function.

The move_uploaded_file() function moves an uploaded file to a new destination.

Note: This function only works on files uploaded via PHP's HTTP POST upload mechanism.

Note: If the destination file already exists, it will be overwritten.

Syntax

move_uploaded_file(*file*, *dest*)

Parameter Values

Parameter	Description
file	Required. Specifies the filename of the uploaded file
dest	Required. Specifies the new location for the file

PHP Cookies

A cookie is a small text file that lets you store a small amount of data (nearly 4KB) on the user's computer. They are typically used to keeping track of information

such as username that the site can retrieve to personalize the page when user visit the website next time.

Each time the browser requests a page to the server, all the data in the cookie is automatically sent to the server within the request.

The setcookie() function is used to set a cookie in PHP. Make sure you call the setcookie() function before any output generated by your script otherwise cookie will not set.

Syntax : setcookie(*name, value, expire, path, domain, secure*);

The parameters of the setcookie() function have the following meanings:

Parameter	**Description**
name	The name of the cookie.
value	The value of the cookie. Do not store sensitive information since this value is stored on the user's computer.
expires	The expiry date in UNIX timestamp format. After this time cookie will become inaccessible. The default value is 0.
path	Specify the path on the server for which the cookie will be available. If set to /, the cookie will be available within the entire domain.
domain	Specify the domain for which the cookie is available to e.g www.example.com.
secure	This field, if present, indicates that the cookie should be sent only if a secure HTTPS connection exists.

```
<?php
// Setting a cookie
setcookie("username", "John Carter", time()+30*24*60*60);
?>
```

Note : If the expiration time of the cookie is set to 0, or omitted, the cookie will expire at the end of the session i.e. when the browser closes.

Accessing Cookies Values

The PHP $_COOKIE superglobal variable is used to retrieve a cookie value. It typically an associative array that contains a list of all the cookies values sent by the browser in the current request, keyed by cookie name.

```
<?php
// Accessing an individual cookie value
echo $_COOKIE["username"];
?>
```

Removing Cookies

You can delete a cookie by calling the same setcookie() function with the cookie name and any value (such as an empty string) however this time you need the set the expiration date in the past

```
<?php
// Deleting a cookie
setcookie("username", "", time()-3600);
?>
```

Note: You should pass exactly the same path, domain, and other arguments that you have used when you first created the cookie in order to ensure that the correct cookie is deleted.

PHP Sessions

A PHP session variable is used to store information about, or change settings for a user session.

Session variables hold information about one single user, and are available to all pages in one application.

PHP Session Variables

When you are working with an application, you open it, do some changes and then you close it.

This is much like a Session. The computer knows who you are. It knows when you start the application and when you end.

But on the internet there is one problem: the web server does not know who you are and what you do because the HTTP address doesn't maintain state.

A PHP session solves this problem by allowing you to store user information on the server for later use (i.e. username, shopping items, etc). However, session information is temporary and will be deleted after the user has left the website. If you need a permanent storage you may want to store the data in a database.

Sessions work by creating a unique id (UID) for each visitor and store variables based on this UID. The UID is either stored in a cookie or is propagated in the URL.

You can solve both of these issues by using the PHP session. A PHP session stores data on the server rather than user's computer. In a session based environment, every user is identified through a unique number called session identifier or SID. This unique session ID is used to link each user with their own information on the server like emails, posts, etc.

The session IDs are randomly generated by the PHP engine which is almost impossible to guess. Furthermore, because the session data is stored on the server, it doesn't have to be sent with every browser request.

Starting a PHP Session

Before you can store any information in session variables, you must first start up the session. To begin a new session, simply call the PHP session_start() function. It will create a new session and generate a unique session ID for the user.

```
<?php
// Starting session
session_start();
?>
```

Storing and Accessing Session Data

You can store all your session data as key-value pairs in the $_SESSION[] superglobal array. The stored data can be accessed during lifetime of a session.

```
<?php
// Starting session
session_start();
```

```
// Storing session data
$_SESSION["firstname"] = "Peter";
$_SESSION["lastname"] = "Parker";
?>
```

Destroying a Session

If you want to remove certain session data, simply unset the corresponding key of the $_SESSION associative array

```
<?php
// Starting session
session_start();

// Removing session data
   unset($_SESSION["lastname"]);  /// particular session destroy
   session_destroy();  /// all session destroy

?>
```

Before destroying a session with the session_destroy() function, you need to first recreate the session environment if it is not already there using the session_start() function, so that there is something to destroy.

PHP Hidden Fields

A hidden field is similar to a text field. The hidden field in a form are embedded in the HTML source code of the form. The difference is that the user cannot view the hidden field and its content. A hidden field enables the web developer to pass variables with values from one form to another without requiring to re-enter the information.

The Syntax to define a hidden field is as follows:

< INPUT TYPE = “HIDDEN” NAME=” hidden1” VALUE=” PHP MESSAGE” >

Where,

TYPE — specifies that the field is hidden

NAME — specifies the name of the hidden field

VALUE – specifies the value as it appears on the form

PHP File Handling

File handling is needed for any application. For some tasks to be done file needs to be processed. File handling in PHP is similar as file handling is done by using any programming language like C. PHP has many functions to work with normal files.

File Handling Operations

File handling starts with creating a file, reading its content, writing into a file to appending data into an existing file and finally closing the file. Php provides pre-defined functions for all these operations.

1. **Create a File**: fopen()
2. **Open a File**: fopen()
3. **Read a File**: fread()
4. **Write to a File**: fwrite()
5. **Append to a File**: fwrite()
6. **Close a File**: fclose()
7. **Delete a File**: unlink()

1 - fopen() – PHP fopen() function is used to open a file. First parameter of fopen() contains name of the file which is to be opened and second parameter tells about mode in which file needs to be opened, e.g.,

```
<?php
$file = fopen(“demo.txt”,'w');
?>
```

Files can be opened in any of the following modes :

- **“w” –** Opens a file for write only. If file not exist then new file is created and if file already exists then contents of file is erased.
- **“r” –** File is opened for read only.
- **“a” –** File is opened for write only. File pointer points to end of file. Existing data in file is preserved.
- **“w+” –** Opens file for read and write. If file not exist then new file is created and if file already exists then contents of file is erased.
- **“r+” –** File is opened for read/write.

- **“a+” –** File is opened for write/read. File pointer points to end of file. Existing data in file is preserved. If file is not there then new file is created.
- **“x” –** New file is created for write only.

2 - fread() — After file is opened using fopen() the contents of data are read using fread(). It takes two arguments. One is file pointer and another is file size in bytes, e.g.,

```
<?php
$filename = "demo.txt";
$file = fopen( $filename, 'r' );
$size = filesize( $filename );
$filedata = fread( $file, $size );
?>
```

3 - fwrite() – New file can be created or text can be appended to an existing file using fwrite() function. Arguments for fwrite() function are file pointer and text that is to written to file. It can contain optional third argument where length of text to written is specified

```
<?php
$file = fopen("demo.txt", 'w');
$text = "Hello world\n";
fwrite($file, $text);
?>
```

4 - fclose() – file is closed using fclose() function. Its argument is file which needs to be closed

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

Introduction to MYSQL

MySQL is a relational database management system (RDBMS) that runs as a server providing multi-user access to a number of databases.

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

MySQL is fully multithreaded using kernel threads, and provides application programming interfaces (APIs) for many programming languages, including C,C++, Java, Perl, PHP, Python.

MySQL database server offers several advantages:

- MySQL is easy to use, yet extremely powerful, fast, secure, and scalable.
- MySQL runs on a wide range of operating systems, including UNIX or Linux, Microsoft Windows, Apple Mac OS X, and others.
- MySQL supports standard SQL (Structured Query Language).
- MySQL is ideal database solution for both small and large applications.
- MySQL is developed, and distributed by Oracle Corporation.
- MySQL includes data security layers that protect sensitive data from intruders.

Additionally, dynamic websites are dependent on stored information that can be modified quickly and easily; this is the main difference between a dynamic site and a static HTML site. However, PHP doesn't provide a simple, efficient way to store data. This is where a relational database management system like MySQL comes into play. PHP provides native support for it and the database is free, open-source project.

MySQL is a relational database management system (DBMS). Essentially, this means that MySQL allows users to store information in a table-based structure, using rows and columns to organize different pieces of data

Integration of PHP with MySQL

It is possible to execute various commands of MySQL from PHP.PHP provides various built-in functions which you allows you to use MySQL commands from PHP page. Thus you can integrate PHP with MySQL.

Following are the various PHP functions that allows you the facility of integrating PHP with MySQL

Database Connectivity in PHP

In PHP, using the mysqli_connect() function you can easily connect to the mysql. All communication between PHP and the MySQL database server takes place through this connection.

Before you can access data in a database, you must create a connection to the database.

mysqli_connect() function allows you to establish connection of PHP application with MySQL server.

Syntax: MySQLi, Procedural way

```
$link = mysqli_connect("hostname", "username", "password", "database");
```

Syntax: MySQLi, Object Oriented way

```
$mysqli = new mysqli("hostname", "username", "password", "database");
```

Syntax: PHP Data Objects (PDO) way

```
$pdo = new PDO("mysql:host=hostname;dbname=database", "username", "password");
```

Servernave	Indicates the name of MySQL server with which you want to establish connection. It is optional .Default value is localhost:3306
UserName	Indicates the name of user using which you can logs on to MySQL Server. Optional. Default value is the name of the user that owns the server process.
Password	Indicates password of the user using which you can logs on to MySQL Server. It is optional. Default is " ".
Database	Select the database, you can perform operation on which database

Closing the MySQL Database Server Connection

The connection to the MySQL database server will be closed automatically as soon as the execution of the script ends. However, if you want to close it earlier you can do this by simply calling the PHP mysqli_close() function.

```
<?php

    $conn = mysqli_connect("localhost", "root", "", "GFG");

    if(mysqli_connect_error())
        echo "Connection Error.";
    else
        echo "Database Connection Successfully.";

   // Close connection
   mysqli_close($link);
?>
```

1 - mysqli_query()

This function allows you to specify and execute the MySQL command on MySQL Server.

Syntax:

mysqli_query(Query, ConnectionName);

Query - Indicates the MySQL command to be executed.

ConnectionName - Indicates the name of the variable that is used at the time of establish connection with MySQL server using mysqli_connect() function.

Example :-

<?php

$conn=mysqli_connect(“localhost”,”root”,” ”, “Mydatabse”);

$cmd=mysqli_query($conn,“ select * from student ”);

if($cmd)

{

echo “Query Excuted ”;

```
}
Else
{
        echo “Error in executing query.”;
}
?>
```

2 - mysqli_fetch_row()

- This function allows you to retrieve a record from the record set that is returned from executing the MySQL query.
- The record that is returned by this function is in the form of numeric array.
- Numeric array contains index and value associated with that index.
- If there is no record in the record set then it returns false value.

Syntax:

```
mysqli_fetch_row(VariableName);
```

VariableName - indicates the record set that is returned from executing the MySQL command using mysqli_query() function.

Example :-

```
<?php
$conn=mysqli_connect(“localhost”,”root”,” ”, “Mydatabse” );
$sql = " SELECT * FROM student ";
if($result = mysqli_query($conn, $sql)){
   if(mysqli_num_rows($result) > 0){
      echo "<table>";
         echo "<tr>";
            echo "<th>id</th>";
            echo "<th>first_name</th>";
            echo "<th>last_name</th>";
            echo "<th>email</th>";
         echo "</tr>";
```

```
        while($row = mysqli_fetch_row($result)){
            echo "<tr>";
                echo "<td>" . $row[0] . "</td>";
                echo "<td>" . $row[1] . "</td>";
                echo "<td>" . $row[2] . "</td>";
                echo "<td>" . $row[3] . "</td>";
            echo "</tr>";
        }
        echo "</table>";
        mysqli_free_result($result);
    } else{
        echo "No records matching your query were found.";
    }
?>
```

3 - mysqli_fetch_array()

This function allows you to retrieve a record from the record set that is returned from executing the MySQL query.

The record that is returned by this function is in the form of either numeric array, associative array or both.

If there is no record in record set then it will returns false value.

Syntax:

mysqli_fetch_array(VariableName, ResultArrayType)

VariableName :- indicates the record set that is returned from executing the MySQL command using mysql_query() function.

ResultArrayType :- indicates the type of array to be returned. It can have one of the following values. It is optional.

MYSQLI_ASSOC	This type of array contains name of the field and the value associated with that field for current record.
MYSQLI_NUM	This type of array contains index of the field and the value associcated with that index for current record.

MYSQLI_BOTH	It is combination of both Associative array and Numeric array. It is the default type to be returned by this function.

Example :-

```
<?php

$conn=mysqli_connect(“localhost”,”root”,” ”, “Mydatabse” );

$sql = " SELECT * FROM student ";
if($result = mysqli_query($conn, $sql)){
   if(mysqli_num_rows($result) > 0){
      echo "<table>";
         echo "<tr>";
            echo "<th>id</th>";
            echo "<th>first_name</th>";
            echo "<th>last_name</th>";
            echo "<th>email</th>";
         echo "</tr>";
      while($row = mysqli_fetch_array($result)){
         echo "<tr>";
            echo "<td>" . $row[‘id’] . "</td>";
            echo "<td>" . $row[‘name’] . "</td>";
            echo "<td>" . $row[‘lname’] . "</td>";
            echo "<td>" . $row[‘email’] . "</td>";
         echo "</tr>";
      }
      echo "</table>";
      mysqli_free_result($result);
   } else{
      echo "No records matching your query were found.";
   }
?>
```

4 - mysqli_fetch_assoc()

This function allows you to retrieve a record from the record set that is returned from executing the MySQL query in the form of associative array.

Syntax:

mysqli_fetch_assoc(VariableName)

Returns an associative array that corresponds to the fetched row, or FALSE if there are no more rows.

Example :-

```
<?php

$conn=mysqli_connect(“localhost”,”root”,” ”, “Mydatabse” );

$sql = " SELECT * FROM student ";
if($result = mysqli_query($conn, $sql)){
   if(mysqli_num_rows($result) > 0){
      echo "<table>";
         echo "<tr>";
            echo "<th>id</th>";
            echo "<th>first_name</th>";
            echo "<th>last_name</th>";
            echo "<th>email</th>";
         echo "</tr>";
      while($row = mysqli_fetch_assoc($result)){
         echo "<tr>";
            echo "<td>" . $row[‘id’] . "</td>";
            echo "<td>" . $row[‘name’] . "</td>";
            echo "<td>" . $row[‘lname’] . "</td>";
            echo "<td>" . $row[‘email’] . "</td>";
         echo "</tr>";
      }
      echo "</table>";
      mysqli_free_result($result);
   } else{
      echo "No records matching your query were found.";
   }
?>
```

5 - mysqli_num_rows()

This function allows you to retrieve number of records available in the record set.

Syntax:

mysqli_num_rows(ResultVariable);

Result Variable is the variable that holds result returned by mysqli_query() function.

Example :-

```
<?php
        $conn=mysqli_connect(“localhost”,”root”,” ”, “Mydatabse” );

        $sql = " SELECT * FROM student ";
        $result = mysqli_query($conn, $sql);
        $total_record = mysqli_num_rows($result)

       echo "Total Records are : ".$total_record ;
       mysqli_free_result($result);
?>
```

7 - mysqli_error()

This function allows you to retrieve the error text from the most recently executed MySQL function.

If no error encountered while executing the script then it will returns blank string.

If the mysql operation contains more than one error then it will returns error description of the last statement in which error is encountered.

Syntax:

mysqli_error();

Example :-

```
<?php
        $conn=mysqli_connect(“localhost”,”root”,” ”, “Mydatabse” );

        // Check connection
        if (!$conn) {
```

```
    die("Connection failed: " . mysqli_error($con));
}
echo "Connected successfully";
?>
```

8 - mysqli_close()

This function allows you to close the connection that is established using mysqli_connect() function.

Syntax:

mysqli_close(ConnectionName);

ConnectionName :- Indicates the name of the variable that is used at the time of establish connection with MySQL server using mysqli_connect() function.

It returns true is connection is closed successfully otherwise it returns false.

Example :-

```
<?php
    $conn=mysqli_connect("localhost","root"," ", "Mydatabse" );
    mysqli_close($conn);
?>
```

PHP Error handling

When an error occurs, depending on your configuration settings, PHP displays the error message in the web browser with information relating to the error that occurred.

PHP offers a number of ways to handle errors.

We are going to look at three (3) commonly used methods;

1. **Die statements**– the die function combines the echo and exit function in one. It is very useful when we want to output a message and stop the script execution when an error occurs.

2. **Custom error handlers** – these are user defined functions that are called whenever an error occurs.
3. **PHP error reporting** – the error message depending on your PHP error reporting settings. This method is very useful in development environment when you have no idea what caused the error. The information displayed can help you debug your application.

PHP Exception Handling

An exception is a signal that indicates some sort of exceptional event or error has occurred. Exceptions can be caused due to various reasons, for example, database connection or query fails, file that you're trying to access doesn't exist, and so on.

PHP provides a powerful exception handling mechanism that allows you to handle exceptions in a graceful way. As opposed to PHP's traditional error-handling system, exception handling is the object-oriented method for handling errors, which provides more controlled and flexible form of error reporting.

PHP provides following specialized keywords for this purpose.

- **try:** It represent block of code in which exception can arise.
- **catch:** It represent block of code that will be executed when a particular exception has been thrown.
- **throw:** It is used to throw an exception. It is also used to list the exceptions that a function throws, but doesn't handle itself.
- **finally:** It is used in place of catch block or after catch block basically it is put for cleanup activity in PHP code.

Using Throw and Try...Catch Statements

In exception-based approach, program code is written in a try block, an exception can be thrown using the throw statement when an exceptional event occurs during the execution of code in a try block. It is then caught and resolved by one or more catch blocks.

```
<?php
function division($dividend, $divisor){
    // Throw exception if divisor is zero
```

```
    if($divisor == 0){
        throw new Exception('Division by zero.');
    } else{
        $quotient = $dividend / $divisor;
        echo "<p>$dividend / $divisor = $quotient</p>";
    }
}

try{
    division(10, 2);
    division(15, 0);

    // If exception is thrown following line won't execute
    echo '<p>All divisions performed successfully.</p>';
} catch(Exception $e){
    // Handle the exception
    echo "<p>Caught exception: " . $e->getMessage() . "</p>";
}

// Continue execution
echo "<p>Hello World!</p>";
?>
```

Authentication

Authentication is the act of validating that users are whom they claim to be. This is the first step in any security process.

Giving someone permission to download a particular file on a server or providing individual users with administrative access to an application are good examples of authentication.

Complete an authentication process through:

- **Passwords.** Usernames and passwords are the most common authentication factors. If a user enters the correct data, the system assumes the identity is valid and grants access.
- **Authentication apps.** Generate security codes via an outside party that grants access.

- **Biometrics.** A user presents a fingerprint or eye scan to gain access to the system.

In some instances, systems require the successful verification of more than one factor before granting access. This multi-factor authentication (MFA) requirement is often deployed to increase security beyond what passwords alone can provide.

Authorization

Authorization in a system security is the process of giving the user permission to access a specific resource or function. This term is often used interchangeably with access control or client privilege.

In secure environments, authorization must always follow authentication. Users should first prove that their identities are genuine before an organization's administrators grant them access to the requested resources.

Difference between authentication and authorization:

S.NO	Authentication	Authorization
1.	In authentication process, the identity of users are checked for providing the access to the system.	While in authorization process, person's or user's authorities are checked for accessing the resources.
2.	In authentication process, users or persons are verified.	While in this process, users or persons are validated.
3.	It is done before the authorization process.	While this process is done after the authentication process.
4.	It needs usually user's login details.	While it needs user's privilege or security levels.
5.	Authentication determines whether the person is user or not.	While it determines **What permission do user have?**

SQL Injection

SQL Injection (SQLi) is a type of an injection attack that makes it possible to execute malicious SQL statements. These statements control a database server behind a web application. Attackers can use SQL Injection vulnerabilities to bypass application security measures. They can go around authentication and

authorization of a web page or web application and retrieve the content of the entire SQL database. They can also use SQL Injection to add, modify, and delete records in the database.

An SQL Injection vulnerability may affect any website or web application that uses an SQL database such as MySQL, Oracle, SQL Server, or others. Criminals may use it to gain unauthorized access to your sensitive data: customer information, personal data, trade secrets, intellectual property, and more. SQL Injection attacks are one of the oldest, most prevalent, and most dangerous web application vulnerabilities.

Prevent SQL Injection Attacks

1. Validate User Inputs

A common first step to preventing SQL injection attacks is validating user inputs. First, identify the essential SQL statements and establish a whitelist for all valid SQL statements, leaving unvalidated statements out of the query. This process is known as input validation or query redesign.

2. Sanitize Data by Limiting Special Characters

Another component of safeguarding against SQL injection attacks is mitigating inadequate data sanitization. Because SQLi attackers can use unique character sequences to take advantage of a database, sanitizing data not to allow string concatenation is critical.

3. Enforce Prepared Statements and Parameterization

Sadly, input validation and data sanitization aren’t fix-alls. It’s critical organizations also use prepared statements with parameterized queries, also known as variable binding, for writing all database queries.

4. Use Stored Procedures in the Database

Similar to parameterization, using stored procedures also requires variable binding. Unlike the prepared statements approach to mitigating SQLi, stored procedures reside in the database and are called from the web application. Stored procedures are also not immune to vulnerabilities if dynamic SQL generation is used.

5. Limit Read-Access

6. Encryption: Keep Your Secrets Secret

It's best to assume internet-connected applications are not secure. Therefore encryption and hashing passwords, confidential data, and connection strings are of the utmost importance.

7. Continuous Monitoring of SQL Statements

Organizations or third-party vendors should continually monitor all SQL statements of database-connected applications for an application, including documenting all database accounts, prepared statements, and stored procedures.

PHP MySQL Prepared Statements(Prevent Sql Injection)

A prepared statement (also known as parameterized statement) is simply a SQL query template containing placeholder instead of the actual parameter values. These placeholders will be replaced by the actual values at the time of execution of the statement.

The prepared statement execution consists of two stages: prepare and execute.

- **Prepare** — At the prepare stage a SQL statement template is created and sent to the database server. The server parses the statement template, performs a syntax check and query optimization, and stores it for later use.
- **Execute** — During execute the parameter values are sent to the server. The server creates a statement from the statement template and these values to execute it.

Advantages of Using Prepared Statements

A prepared statement can execute the same statement repeatedly with high efficiency, because the statement is parsed only once again, while it can be executed multiple times. It also minimize bandwidth usage, since upon every execution only the placeholder values need to be transmitted to the database server instead of the complete SQL statement.

Prepared statements also provide strong protection against SQL injection, because parameter values are not embedded directly inside the SQL query string. The

parameter values are sent to the database server separately from the query using a different protocol and thus cannot interfere with it. The server uses these values directly at the point of execution, after the statement template is parsed.

Example :

```
<?php
$link = mysqli_connect("localhost", "root", "", "demo");

// Prepare an insert statement
$sql = "INSERT INTO persons (first_name, last_name, email) VALUES (?, ?, ?)";

if($stmt = mysqli_prepare($link, $sql)){
    mysqli_stmt_bind_param($stmt, "sss", $first_name, $last_name, $email);

    $first_name = "Hermione";
    $last_name = "Granger";
    $email = "hermionegranger@mail.com";

    mysqli_stmt_execute($stmt);

    echo "Records inserted successfully.";
} else{
    echo "ERROR: Could not prepare query: $sql. " . mysqli_error($link);
}

// Close statement
mysqli_stmt_close($stmt);

// Close connection
mysqli_close($link);
?>
```

Handling special characters in input

1 - htmlspecialchars() Function

The htmlspecialchars() function converts some predefined characters to HTML entities.

The htmlspecialchars() function converts special characters to HTML entities. This means that it will replace HTML characters like < and > with < and >. This prevents attackers from exploiting the code by injecting HTML or Javascript code (Cross-site Scripting attacks) in forms.

The predefined characters are:

- & (ampersand) becomes &
- " (double quote) becomes "
- ' (single quote) becomes '
- < (less than) becomes <
- > (greater than) becomes >

```
<?php
$str = "This is some <b>bold</b> text.";
echo htmlspecialchars($str);
?>
```

2 - mysqli_real_escape_string() Function

The mysqli_real_escape_string() function is an inbuilt function in PHP which is used to escape all special characters for use in an SQL query. It is used before inserting a string in a database, as it removes any special characters that may interfere with the query operations.

When simple strings are used, there are chances that special characters like backslashes and apostrophes are included in them (especially when they are getting data directly from a form where such data is entered).

```
<?php

mysqli_real_escape_string(connection string , variable name);
?>
```

3 - addslashes() Function

The **addslashes()** function is an inbuilt function in PHP and it returns a string with backslashes in front of predefined characters. It does not take any specified characters in the parameter.

The predefined characters are:

1. single quote (‘)
2. double quote (“)
3. backslash (\)
4. NULL

Syntax:

addslashes($string)

Note: The addslashes() function is different from addcslashes() function accepts specified characters before which be want to add slashes but the addslashes() function doesnot accepts any character in parameters, rather it adds slashes before some specified characters.

4 - htmlentities() Function

The htmlentities() function is an inbuilt function in PHP which is used to transform all characters which are applicable to HTML entities. This function converts all characters that are applicable to HTML entity.

Syntax

string htmlentities($string, $flags, $encoding, $double_encode)

Parameters: This function accepts four parameters as mentioned above and described below:

- **$string:** This parameter is used to hold the input string.
- **$flags:** This parameter is used to hold the flags. It is combination of one or two flags, which tells how to handle quotes.
- **$encoding:** It is an optional argument which specifies the encoding which is used when characters are converted. If encoding is not given then it is converted according to PHP default version.
- **$double_encode:** If double_encode is turned off then PHP will not encode existing HTML entities. The default is to convert everything.

```
<?php
// String convertable to htmlentities
$str = '<a href="https://www.geeksforgeeks.org">GeeksforGeeks</a>';
```

```
// It will convert htmlentities and print them
echo htmlentities( $str );
?>
```