Unit 1 (Basics of php)

php introduction

1. PHP is a server-side scripting language created primarily for web development but it is also used as a general-purpose programming language.
2. Unlike client-side languages like JavaScript, which are executed on the user's browser, PHP scripts run on the server.
3. The results are then sent to the client's web browser as plain HTML.
4. It can be integrated with many databases such as Oracle, Microsoft SQL Server, MySQL, PostgreSQL, Sybase, and Informix.
5. It is powerful to hold a content management system like WordPress and can be used to control user access.
6. PHP is dynamically typed, meaning you don't need to declare the data type of a variable explicitly.
7. Websites like *www.facebook.com/* and *www.yahoo.com/* are also built on PHP.

Advantages of php

- **Easy to learn**: PHP has a simple syntax that's similar to HTML, making it easy for beginners to learn.
- **Cost-effective**: PHP is free and open-source, so there's no need to buy licenses or software.
- **Scalable**: PHP can handle increased traffic and user interactions by using caching, load balancing, and code optimization.
- **Platform-independent**: PHP runs on many operating systems, including Windows, Linux, and Unix.
- **Database-friendly**: PHP can connect to databases quickly and securely.
- **Extensive library**: PHP has a large library of pre-written code, so you don't need to start from scratch.
- **Cloud-compatible**: PHP-based applications can run on cloud computing platforms like Amazon Web Services.
- **Integratable**: PHP can integrate with other programming languages and technologies, like Java.

- **Flexible**: PHP can be used with many different operating systems and programming languages.
- **Gives developers more control**: PHP gives developers more control than other programming languages.

The use of $ in php/variables in php

1. All variable names start with a dollar sign ($). This tells PHP that it is a variable name.
2. Variable names can be any length.
3. Variable names can include letters, numbers, and underscores only.
4. Variable names must begin with a letter or an underscore.
5. They cannot begin with a number.
6. Variable names in PHP are case-sensitive, so "$name" is different from "$Name".

---

constants

1. Constants in PHP are identifiers or names that can be assigned any fixed values and cannot change during the execution of a program.
2. They remain constant throughout the program and cannot be changed during the execution.
3. Once a constant is defined, it cannot be undefined or redefined.
4. By convension, constant identifiers are always written in upper case.
5. By default, a constant is always case-sensitive.
6. A constant name must never start with a number.
7. It always starts with a letter or underscores, followed by letter, numbers or underscore.
8. It should not contain any special characters except underscore, as mentioned.

Data types

1. **Integer:** Whole numbers, positive or negative, without any decimal point.

```
5, -23, 100
```

2. **Float (or Double):** Numbers with decimal points or numbers in exponential form.

```
 3.14, -0.5, 2.1e3
```

3. **String:** A sequence of characters enclosed in single or double quotes.

```
"Hello World", 'PHP'
```

4. **Boolean:** Represents two possible states: TRUE or FALSE.

```
 TRUE, FALSE
```

5. **Array:** A collection of values that can be of different data types, stored under a single variable name.

- Indexed Array: An array with numeric indexes.

```
$arr = [1, 2, 3]
```

- `Associative Array: An array where each element is associated with a key (string or integer).`

```
 $arr = ['name' => 'John', 'age' => 30]
```

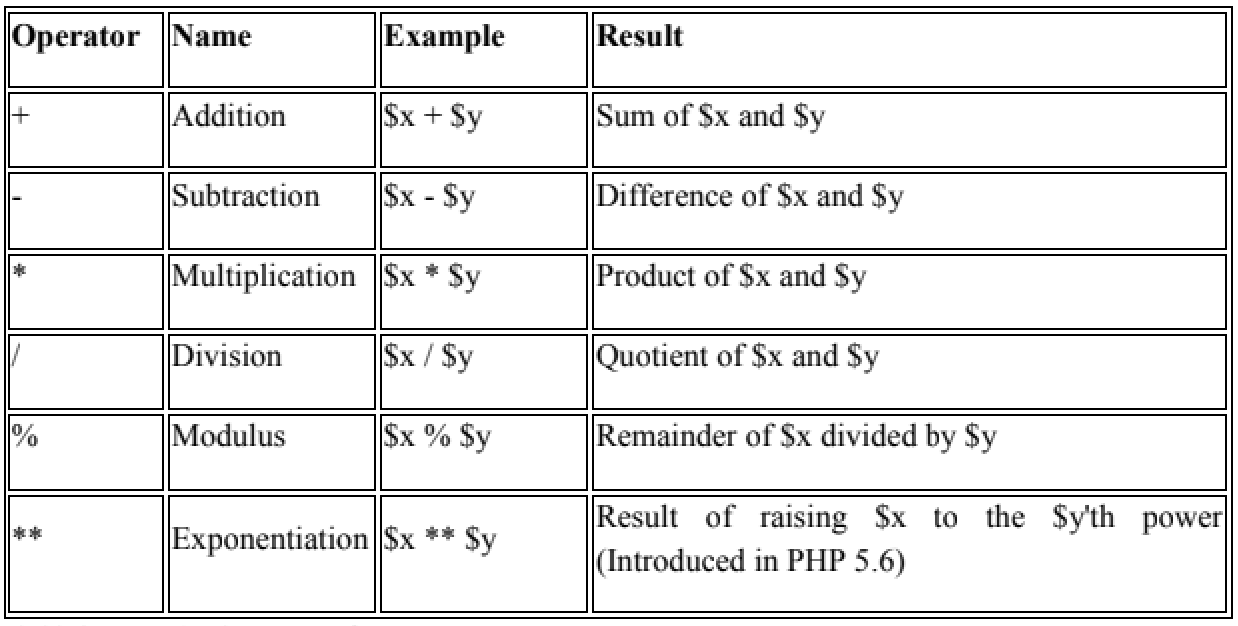
6. **NULL:** Represents a variable with no value or an empty state.

```
$var = NULL;
```

7. **Resource:** A special variable used to store references to external resources like database connections or file handles.

```
$file = fopen("file.txt", "r");
```

Arithmetic Operators



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

Assignment operator

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

Comparison operator

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

---

Increment/Decrement operator

The PHP decrement operators are used to decrement a variable's value.

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

Logical operators

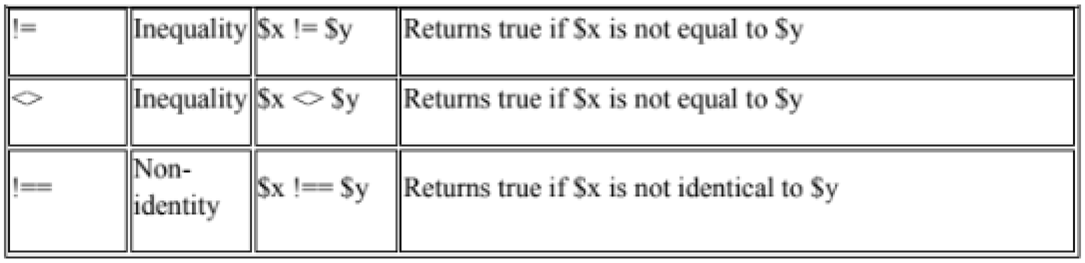
Operator	Name	Example	Result
And	And	$x and $y	True if both $x and $y are true
Or	Or	$x or $y	True if either $x or $y is true
Xor	Xor	$x xor $y	True if either $x or $y is true, but not both
&&	And	$x && $y	True if both $x and $y are true
||	Or	$x || $y	True if either $x or $y is true
!	Not	!$x	True if $x is not true

String operators

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

Array operators

Operator	Name	Example	Result
+	Union	$x + $y	Union of $x and $y
==	Equality	$x == $y	Returns true if $x and $y have the same key/value pairs
===	Identity	$x === $y	Returns true if $x and $y have the same key/value pairs in the same order and of the same types



!=	Inequality	$x != $y	Returns true if $x is not equal to $y
<>	Inequality	$x <> $y	Returns true if $x is not equal to $y
!==	Non-identity	$x !== $y	Returns true if $x is not identical to $y

---

Conditional statements

1. `if` **Statement:** The `if` statement checks a condition, and if the condition is true, it executes the block of code inside the statement.

```
$age = 20;
if ($age >= 18) {
    echo "You are eligible to vote.";
}
```

2. `if...else` **Statement:** The `if...else` statement allows you to define a block of code to be executed if the condition is true, and another block of code to be executed if the condition is false.

```
$age = 15;
if ($age >= 18) {
    echo "You are eligible to vote.";
} else {
    echo "You are not eligible to vote.";
}
```

3. `if...elseif...else` **Statement:** The `if...elseif...else` statement checks multiple conditions. It allows you to specify different blocks of code to execute for different conditions.

```
$age = 30;
if ($age < 18) {
    echo "You are a minor.";
} elseif ($age >= 18 && $age <= 60) {
    echo "You are an adult.";
} else {
    echo "You are a senior citizen.";
}
```

4. **`switch` Statement:** The `switch` statement is used to perform different actions based on different conditions. It's useful when you have many conditions to check for a single variable.

```
$day = 3;
switch ($day) {
    case 1:
        echo "Monday";
        break;
    case 2:
        echo "Tuesday";
        break;
    case 3:
        echo "Wednesday";
        break;
    case 4:
        echo "Thursday";
        break;
    case 5:
        echo "Friday";
        break;
    case 6:
        echo "Saturday";
        break;
    case 7:
        echo "Sunday";
        break;
    default:
        echo "Invalid day";
}
```

5. **Ternary Operator:** The ternary operator is a shorthand for an `if...else` statement. It is used to return one of two values depending on the condition.

```
$age = 20;
echo ($age >= 18) ? "You are eligible to vote." : "You are not
eligible to vote.";
```

6. **Null Coalescing Operator (`??`):** The null coalescing operator returns the value of its left-hand operand if it exists and is not null; otherwise, it returns the value of the right-hand operand.

```
$username = $_GET['user'] ?? 'Guest';
```

```
echo $username; // If 'user' is not set in the query string, it will
print 'Guest'.
```

Comments

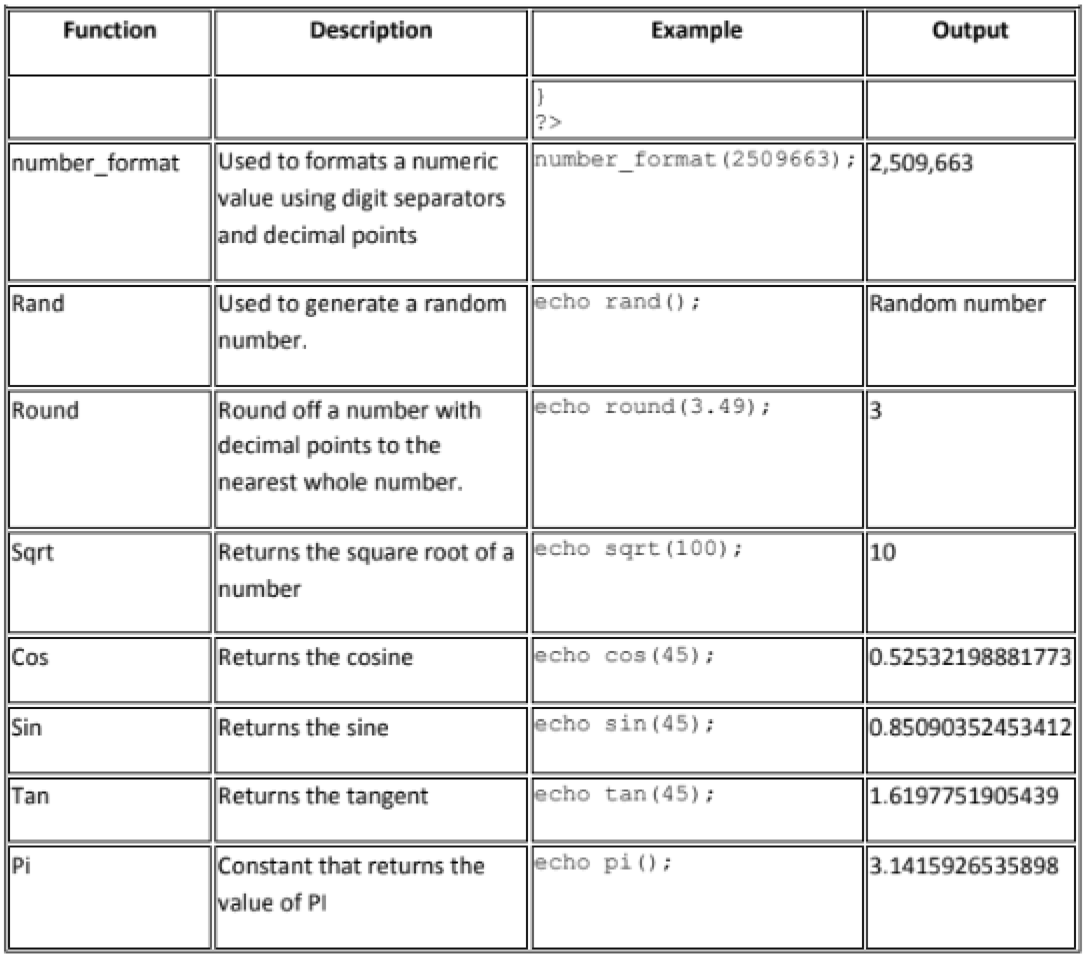
```
# This is a single line commmment
//This is also a single line comment

/*This is a
multiline comment*/
```

PHP string functions

Function	Description	Example	Output
Strtolower	Used to convert all string characters to lower case letters	echo strtolower('Benjamin');	outputs benjamin
Strtoupper	Used to convert all string characters to upper case letters	echo strtoupper('george w bush');	outputs GEORGE W BUSH
Strlen	The string length function is used to count the number of character in a string. Spaces in between characters are also counted	echo strlen('united states of america');	24
Explode	Used to convert strings into an array variable	$settings = explode(';', "host=localhost; db=sales; uid=root; pwd=demo"); print_r($settings);	Array ([0] => host=localhost [1] =>db=sales [2] =>uid=root [3] =>pwd=demo)
Substr	Used to return part of the string. It accepts three (3) basic parameters. The first one is the string to be shortened, the second parameter is the position of the starting point, and the third parameter is the number of characters to be returned.	$my_var = 'This is a really long sentence that I wish to cut short';echosubstr($my_var,0, 12).'...';	This is a re...
str_replace	Used to locate and replace specified string values in a given string. The function accepts three arguments. The first argument is the text to be replaced, the second argument is	echo str_replace ('the', 'that', 'the laptop is very expensive');	that laptop is very expensive

Function	Description	Example	Output
	the replacement text and the third argument is the text that is analyzed.		
Strops	Used to locate the and return the position of a character(s) within a string. This function accepts two arguments	echo strpos('PHP Programing','Pro');	4
sha1	Used to calculate the SHA-1 hash of a string value	echo sha1('password');	5baa61e4c 9b93f3f0 682250b6cf8331b 7ee68fd8
md5	Used to calculate the md5 hash of a string value	echo md5('password');	9f961034ee 4de758 baf4de09ceeb1a75
str_word_count	Used to count the number of words in a string.	echo str_word_count ('This is a really long sentence that I wish to cut short');	12
Ucfirst	Make the first character of a string value upper case	echo ucfirst('respect');	Outputs Respect
Lcfirst	Make the first character of a string value lower case	echo lcfirst('RESPECT');	Outputs rESPECT

Numeric Functions



Function	Description	Example	Output
		`}` `?>`	
number_format	Used to formats a numeric value using digit separators and decimal points	`number_format(2509663);`	2,509,663
Rand	Used to generate a random number.	`echo rand();`	Random number
Round	Round off a number with decimal points to the nearest whole number.	`echo round(3.49);`	3
Sqrt	Returns the square root of a number	`echo sqrt(100);`	10
Cos	Returns the cosine	`echo cos(45);`	0.52532198881773
Sin	Returns the sine	`echo sin(45);`	0.85090352453412
Tan	Returns the tangent	`echo tan(45);`	1.6197751905439
Pi	Constant that returns the value of PI	`echo pi();`	3.1415926535898

Getting the Current Date and Tirme

1. The `date()` function formats a local date and time based on a specified format.

```
date(format, timestamp);
```

2. The `time()` function returns the current Unix timestamp (number of seconds since January 1, 1970, 00:00:00 UTC).

```
echo time(); // Outputs: 1678114416 (Unix timestamp)
```

Date and Time Format Characters

1. Day Formats:

- `d` – Day of the month (01 to 31)

- `D` – A three-letter abbreviation for the day of the week (Mon, Tue, etc.)
- `j` – Day of the month without leading zeros (1 to 31)
- `l` – Full name of the day of the week (Monday, Tuesday, etc.)
- `N` – ISO-8601 numeric representation of the day of the week (1 for Monday, 7 for Sunday)
- `S` – English ordinal suffix for the day of the month (st, nd, rd, th)

2. Month Formats:

- `m` – Numeric representation of the month (01 to 12)
- `M` – A three-letter abbreviation for the month (Jan, Feb, etc.)
- `n` – Numeric representation of the month without leading zeros (1 to 12)
- `F` – Full name of the month (January, February, etc.)
- `t` – Number of days in the given month (28 to 31)

3. Year Formats:

- `Y` – Four-digit representation of the year (2025)
- `y` – Two-digit representation of the year (25)

4. Time Formats:

- `H` – 24-hour format of an hour (00 to 23)
- `i` – Minutes with leading zeros (00 to 59)
- `s` – Seconds with leading zeros (00 to 59)
- `a` – Lowercase am or pm
- `A` – Uppercase AM or PM
- `g` – 12-hour format of an hour without leading zeros (1 to 12)
- `G` – 24-hour format of an hour without leading zeros (0 to 23)

Unit 2 (PHP Form Handling)

Indexed arrays

An indexed array is an array where the elements are accessed using numeric indices (starting from 0 by default).
There are two ways to define an indexed array:

1. Indexed Array with Numeric Index

```
$fruits = array("Apple", "Banana", "Cherry");
echo $fruits[0]; // Outputs: Apple
```

2. Indexed Array with Custom Numeric Index

```
$fruits = array(0 => "Apple", 1 => "Banana", 2 => "Cherry");
echo $fruits[1]; // Outputs: Banana
```

3. Using the `[]` shorthand to create an indexed array

```
$fruits = ["Apple", "Banana", "Cherry"];
echo $fruits[2]; // Outputs: Cherry
```

Multidimensional Arrays in PHP

A **multidimensional array** is an array that contains one or more arrays. You can think of it like a matrix or a table of rows and columns.

```
$students = array(
    array("Alice", 20, "A"),
    array("Bob", 22, "B"),
    array("Charlie", 21, "A")
);

echo $students[1][0]; // Outputs: Bob
```

Accessing and Manipulating Arrays in PHP

1. Accessing elements: You can access array elements using the index or key (for associative arrays).

```
echo $fruits[0]; // For indexed arrays
echo $grades["Math"]["Bob"]; // For associative arrays
```

2. **Modifying elements:** You can modify the value of an array element by assigning a new value to a specific index or key.

```
$fruits[1] = "Mango"; // Changes "Banana" to "Mango"
$grades["Science"]["Alice"] = 95; // Changes Alice's grade in Science
```

3. **Looping through arrays:**Use `foreach` to loop through indexed and associative arrays.

```
// Indexed array
foreach($fruits as $fruit) {
    echo $fruit . "<br>";
}

// Associative array
foreach($grades as $subject => $students) {
    echo $subject . ":<br>";
    foreach($students as $student => $grade) {
        echo "$student - $grade<br>";
    }
}
```

---

built-in array sorting functions

1. sort(): This function sorts the array in ascending order and reindexes the array (the keys are reset to 0, 1, 2, ...).

```
$fruits = ["Banana", "Apple", "Cherry"];
sort($fruits);
print_r($fruits);
// Output: Array ( [0] => Apple [1] => Banana [2] => Cherry )
```

2. rsort(): This function sorts the array in descending order and reindexes the array.

```
$fruits = ["Banana", "Apple", "Cherry"];
rsort($fruits);
print_r($fruits);
// Output: Array ( [0] => Cherry [1] => Banana [2] => Apple )
```

3. asort(): This function sorts the array in ascending order according to the values while maintaining the keys.

```
$fruits = ["a" => "Banana", "b" => "Apple", "c" => "Cherry"];
asort($fruits);
print_r($fruits);
// Output: Array ( [b] => Apple [a] => Banana [c] => Cherry )
```

4. ksort(): This function sorts the array by the keys in ascending order while maintaining the key-value pairs.

```
$fruits = ["a" => "Banana", "c" => "Apple", "b" => "Cherry"];
ksort($fruits);
print_r($fruits);
// Output: Array ( [a] => Banana [b] => Cherry [c] => Apple )
```

5. arsort(): This function sorts the array in descending order according to the values while maintaining the keys.

```
$fruits = ["a" => "Banana", "b" => "Apple", "c" => "Cherry"];
arsort($fruits);
print_r($fruits);
// Output: Array ( [c] => Cherry [a] => Banana [b] => Apple )
```

6. ksort(): This function sorts the array by the keys in ascending order while maintaining the key-value pairs.

```
$fruits = ["a" => "Banana", "c" => "Apple", "b" => "Cherry"];
ksort($fruits);
print_r($fruits);
// Output: Array ( [a] => Banana [b] => Cherry [c] => Apple )
```

Request Method

1. The $_REQUEST method in PHP is a superglobal array that is used to collect data after submitting forms.
2. It can contain data sent via GET, POST, and COOKIE methods, making it a convenient way to access information from various sources in a single array.
3. It combines the values from $_GET, $_POST, and $_COOKIE.
4. It's useful when you don't know exactly whether data will be sent using GET or POST, or when both GET and POST might be involved.
5. The $_REQUEST array will give preference to POST data, then GET, and finally COOKIE data if they share the same key.

6.
   ```
   <form method="get" action="example.php">
       <input type="text" name="name">
       <input type="submit" value="Submit">
   </form>
   ```

7.
   ```
   example.php
   echo $_REQUEST['name'];  // Outputs the value of the 'name'
   parameter
   ```

GET method

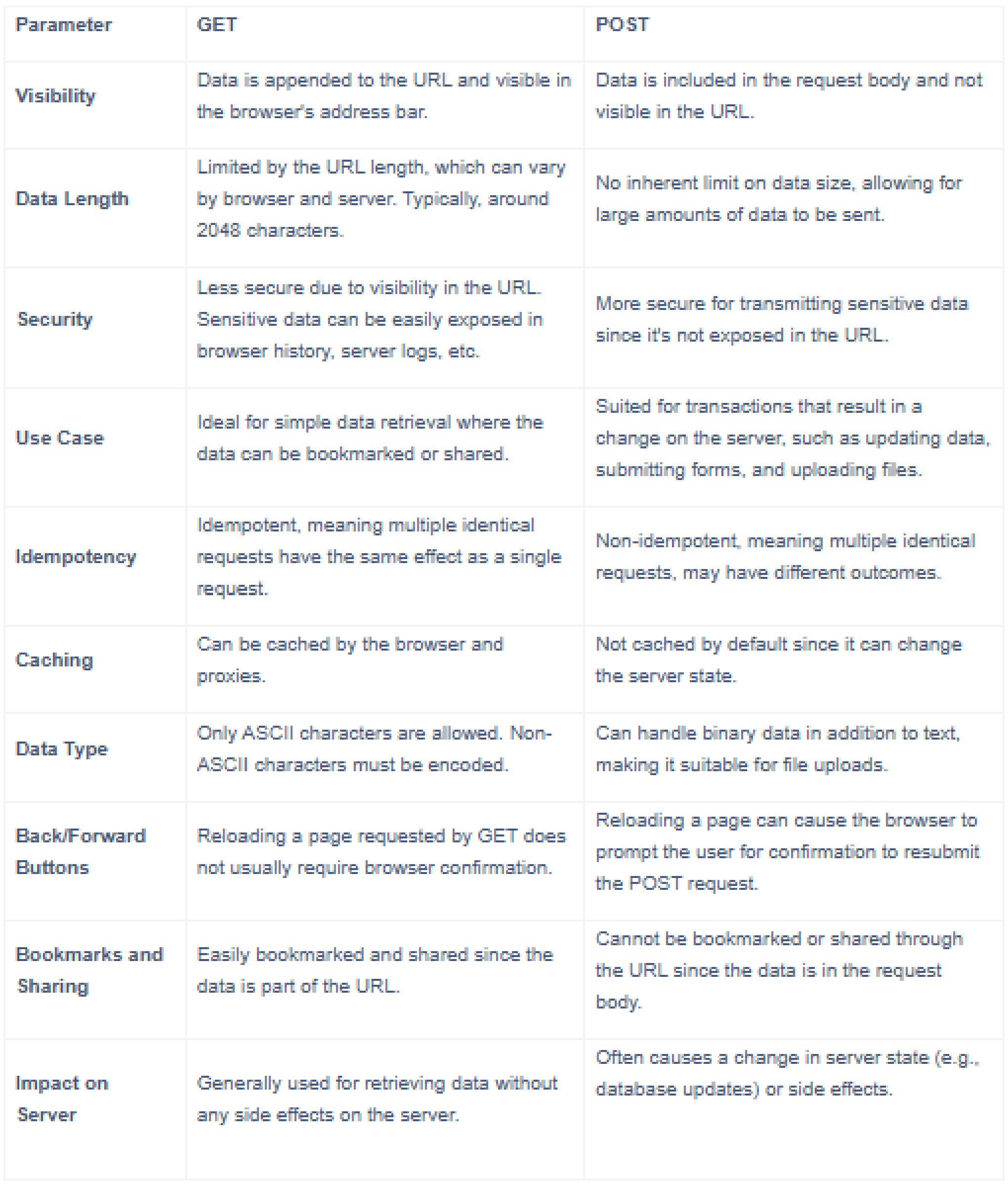
1. The GET method is used in web development to request data from a server.

2. It is one of the most common HTTP request methods.
3. When you enter a website URL or click a link, your browser sends a GET request to the server, asking for the webpage.
4. It Retrieves data like a webpage, image, or API response.
5. Data is visible in the URL (e.g., *example.com/search?q=apple*).
6. It Does not modify server data, only fetches information.
7. It is Cached by browsers to improve speed for repeated requests.
8. It has Limited data length because URLs have size limits.

POST method

1. The POST method is used in web development to send data to a server to create or update resources.
2. Unlike the GET method, POST sends data in the request body, making it more secure and suitable for sensitive information like passwords or form data.
3. Sends data to the server (e.g., submitting a form, uploading a file).
4. Data is not visible in the URL instead it is sent in the request body).
5. Modifies server data like creating a new user or post).
6. It is not cached by browsers it ensures fresh requests.
7. There is no data size limit unlike GET.

Difference in GET and POST



Parameter	GET	POST
Visibility	Data is appended to the URL and visible in the browser's address bar.	Data is included in the request body and not visible in the URL.
Data Length	Limited by the URL length, which can vary by browser and server. Typically, around 2048 characters.	No inherent limit on data size, allowing for large amounts of data to be sent.
Security	Less secure due to visibility in the URL. Sensitive data can be easily exposed in browser history, server logs, etc.	More secure for transmitting sensitive data since it's not exposed in the URL.
Use Case	Ideal for simple data retrieval where the data can be bookmarked or shared.	Suited for transactions that result in a change on the server, such as updating data, submitting forms, and uploading files.
Idempotency	Idempotent, meaning multiple identical requests have the same effect as a single request.	Non-idempotent, meaning multiple identical requests, may have different outcomes.
Caching	Can be cached by the browser and proxies.	Not cached by default since it can change the server state.
Data Type	Only ASCII characters are allowed. Non-ASCII characters must be encoded.	Can handle binary data in addition to text, making it suitable for file uploads.
Back/Forward Buttons	Reloading a page requested by GET does not usually require browser confirmation.	Reloading a page can cause the browser to prompt the user for confirmation to resubmit the POST request.
Bookmarks and Sharing	Easily bookmarked and shared since the data is part of the URL.	Cannot be bookmarked or shared through the URL since the data is in the request body.
Impact on Server	Generally used for retrieving data without any side effects on the server.	Often causes a change in server state (e.g., database updates) or side effects.

PHP validation

PHP validation ensures that user input is correct, secure, and follows required formats before processing. It is divided into **client-side** (JavaScript) and **server-side** (PHP) validation, but **server-side validation is essential** for security.

Types of PHP Validation:

1. **Form Validation** – Checks if required fields are filled.
2. **Data Type Validation** – Ensures correct input type (e.g., numbers for age).
3. **Format Validation** – Uses regex for emails, phone numbers, etc.
4. **Length Validation** – Limits input size to prevent overflow.
5. **Security Validation** – Prevents SQL Injection, XSS attacks.
