XAMPP Server

XAMPP Server

- XAMPP stands for Cross-Platform (X), Apache (A), MySQL (M), **PHP** (P) and **Perl**(P).
- It is a simple, light-weighted Apache server that makes it extremely easy for developers to create a local http server
- Free software

Features and Requirements of XAMPP

- Requires only one .exe or .zip file for configurations (no configuration of various components of server is required)
- It regularly updates latest release of Apache/MySQL/PHP/Perl
- Installing XAMPP takes less time than installing individual components
- Self contained and multiple instances of XAMPP can exist on same computer

Starting XAMPP Server

- Open XAMPP control panel
- Start Apache, MySQL services(required services)
- Minimize the Control Panel
- **Save your folder(containing multiple webpages) in htdocs folder inside xampp folder** (.html/.php/.js/.css)
- Check if the XAMPP server has started properly by
  - Create a web page
  - Go to Web browser (type : **localhost/sample**)
  - View the page

Basics of PHP

What is PHP?

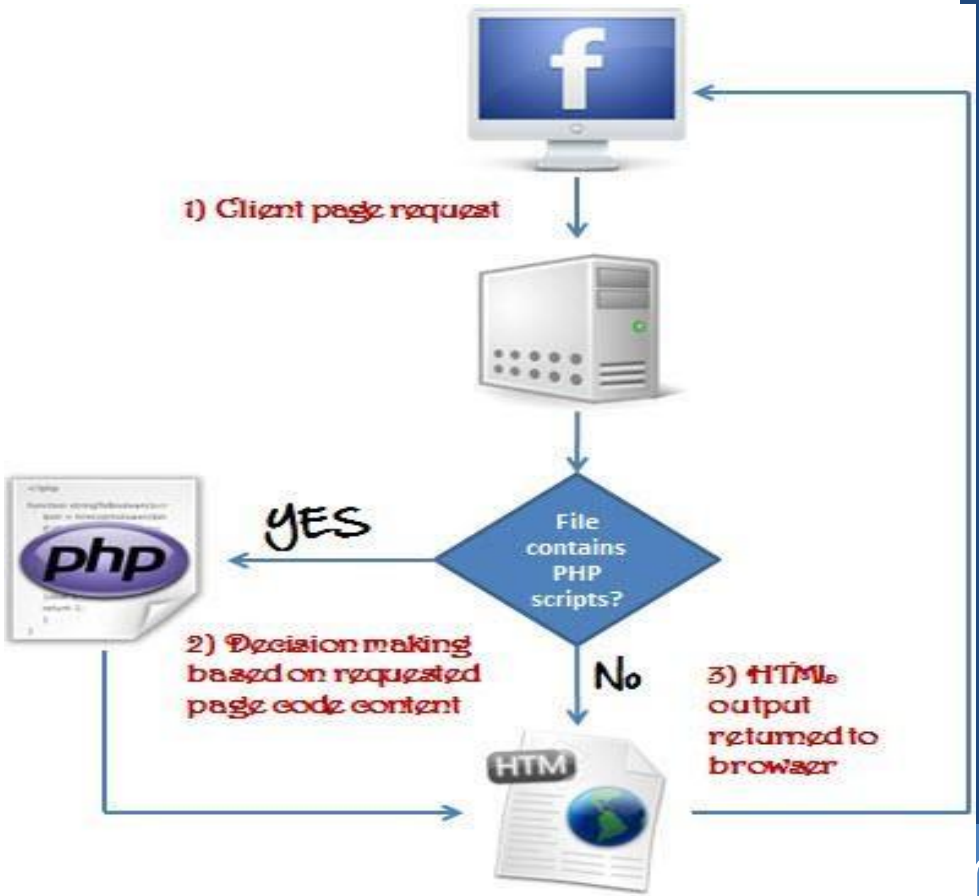
- PHP is an acronym for "PHP: Hypertext Preprocessor"
- PHP is a widely-used, open source scripting language
- PHP scripts are executed on the server

What is a PHP File?

- PHP files can contain text, HTML, CSS, JavaScript, and PHP code

- PHP files have extension

  ".**php**"



What Can PHP Do?

- Generate dynamic page content
- Create, open, read, write, delete, and close files on the server
- Collect form data
- Send and receive cookies
- Add, delete, modify data in your database
- To control user-access
- Data encryption

With PHP can generate output HTML, images, PDF files, and even Flash movies.

Why PHP?

- PHP runs on various platforms (Windows, Linux, Unix, Mac OS X, etc.)
- PHP is compatible with almost all servers used today (Apache, IIS etc.)
- PHP supports a wide range of databases
- PHP is free. Download it from the official PHP resource: www.php.net
- PHP is easy to learn and runs efficiently on the server side

Install PHP

- To install PHP, suggest you to install AMP (Apache, MySQL, PHP) software stack.
- It is available for all operating systems. There are many AMP options available in the market that are given below:
- **WAMP** for Windows

  (http://www.wampserver.com/en/)
- **LAMP** for Linux

  (http://csg.sph.umich.edu/abecasis/LAMP/download/)
- **MAMP** for Mac

  (https://www.mamp.info/en/downloads/)
- **XAMPP** (Cross, Apache, MySQL, PHP, Perl) for Cross Platform

  (https://www.apachefriends.org/download.html)

Example of PHP

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

<h1>My first PHP page</h1>

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

</body>
</html>
```

In PHP, all keywords (e.g. if, else, while, echo, etc.), classes, functions, and user-defined functions are NOT case-sensitive. However; all variable names are case-sensitive.

PHP echo

- PHP echo is a language construct not a function, so you don't need to use parenthesis with it.
- But if you want to use more than one parameters, it is required to use parenthesis.
- **Syntax:**

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

- PHP echo statement can be used to print string, multi line strings, escaping characters, variable, array etc.

Example: PHP echo

```
<?php
echo "Hello by PHP echo";
                                        //print string

echo "Hello by PHP echo
this is multi line
text printed by
PHP echo statement
";                                      // multi line string

echo "Hello escape \"sequence\" characters";      //escape characters

$msg="Hello JavaTpoint PHP";
echo "Message is: $msg";                //print variable value

?>
```

Examples

```
echo " Hello", "World";

echo "Hello"."World";
Echo "<ul>";

echo "<li> Hello India </li>";
Echo "</ul>";
echo 500;
```

PHP print

- Like PHP echo, PHP print is a language construct, so you don't need to use parenthesis with the argument list. Unlike echo, it always returns 1.
- **Syntax:**

int print(string $arg)

- PHP print statement can be used to print string, multi line strings, escaping characters, variable, array etc.

Example: PHP print

```
<?php
print "Hello by PHP echo";          //print string

print "Hello by PHP echo
this is multi line
text printed by
PHP echo statement                  // multi line string
";

print "Hello escape \"sequence\" characters";       //escape characters

$msg="Hello PHP";
print "Message is: $msg";                  //print variable value

?>
```

PHP echo and print Statements

- echo and print are more or less the same.
- Used to output data to the screen.

- **Differences:**
  - 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.

PHP Variables

- 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 can only contain alpha-numeric characters and underscores (A-z, 0-9, and _)
- A variable name **cannot start with a number**
- Variable names **are case-sensitive** ($age and $AGE are two different variables)

PHP has three different variable scopes:

- local
- global
- static

How to write a Variable Name :

Write Way

- $firstname
- $_firstname
- $first_name
- $first-name
- $firstName
- $firstname99

Wrong Way

- $first name
- $99firstname
- $first%name

$age $AGE

Not same

Example: Variables

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

echo $txt;
echo "<br>";
echo $x;
echo "<br>";
echo $y;
?>
</body> </html>
```

Example: Variables

```
<?php
$txt = ”K J Somaiya College of Engineering";
echo ”Name of my College is $txt";
?>
```

Same as

```
<?php
$txt = ”K J Somaiya College of Engineering ";
echo ”Name of my College is " . $txt . ”!!";
?>
```

What will be the output of

```
<?php
$x =15;
$y =41;
echo $x + $y;
?>
```

Output:

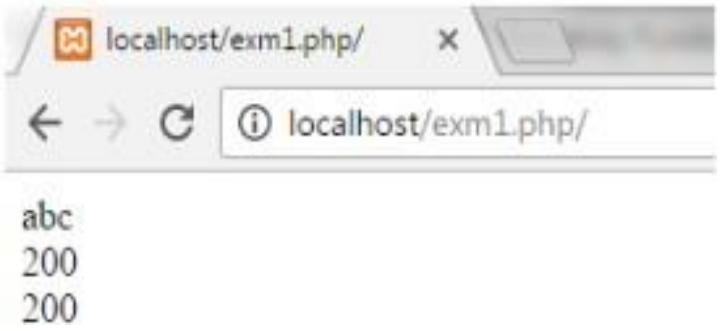
1541 Or 56

PHP $ and $$ Variables

- The **$var** (single dollar) is a normal variable with the name var that stores any value like string, integer, float, etc.
- The **$$var** (double dollar) is a reference variable that stores the value of the $variable inside it.

```
<?php
$x = "abc";
$$x = 200;
echo $x."<br/>";
echo $$x."<br/>";
echo $abc;
?>
```

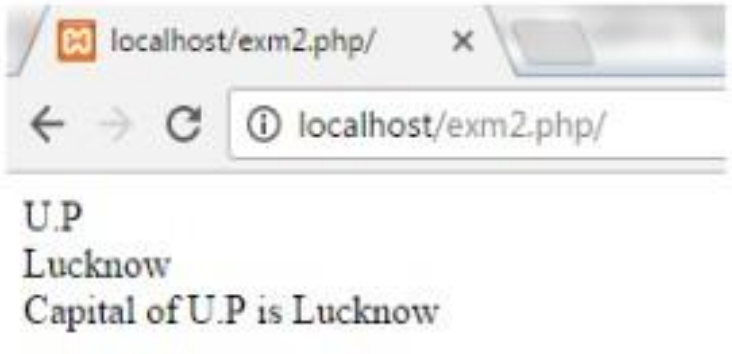
Output:



Example:

```
<?php
 $x="U.P";
$$x="Lucknow";
echo  $x.  "<br>";
echo $$x. "<br>";
echo "Capital of $x is " . $$x;
?>
```

Output:

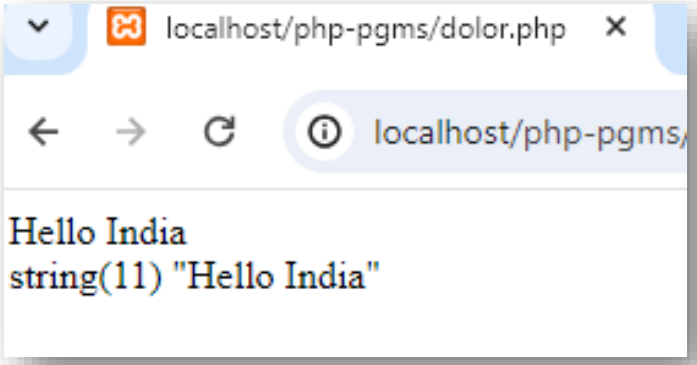


Data types

$x = "Hello World";	String
$x = 25;	Integer
$x = 30.50;	Float
$x = true;	Boolean
$x = array("HTML","CSS","JS");	Array
$x = new MyClass();	Object
$x = null;	Null

Var_dump()-
Return datatype of a variable

```
<?php
$x="Hello India";
echo $x. "<br>";
var_dump($x);
?>
```



PHP Constants

- A constant is an identifier (name) for a simple value.
- The value cannot be changed during the script.
- A valid constant name starts with a letter or underscore (**no $ sign before the constant name**).
- PHP constants can be defined by 2 ways:
  - **Using define() function**

    Syntax:

    define(*name, value, case-Insensitive)*

    -1.name of the variable

    -2. value-value of the variable

    -3. fix the case sensitivity of variable (true or false)

    default value is false

  - **Using const keyword**
    - It is always case sensitive.
    - It is a language construct not a function.
    - It is bit faster than define().

Example: Constants

```
<?php
define(“College", ”K J Somaiya College of Engineering ");
echo College;

define(“College", ”K J Somaiya College of Engineering “, true);
echo College;

const MESSAGE=" K J Somaiya College of Engineering ";
echo MESSAGE;

?>
```

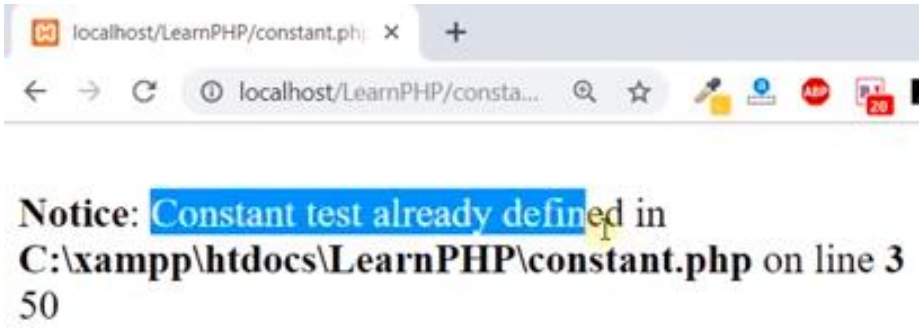
Php Constants

```
define(num, 500, true);

define(_num, 500, true);
```

Can't use $ sign with constant variable name.

Constant Variables are Global Variables.

```php
<?php
define("test",50);
define("test",10);

echo test;

?>
```



PHP Operators

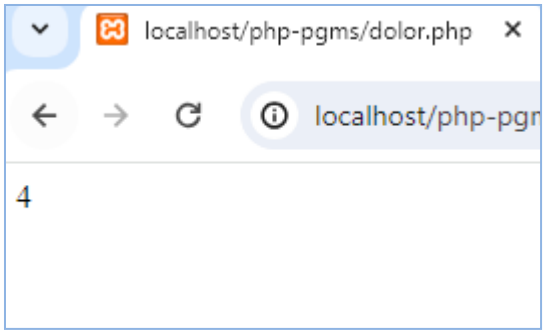
- Operators are used to perform operations on variables and values.
  - Arithmetic operators
  - Assignment operators
  - Comparison operators
  - Increment/Decrement operators
  - Logical operators
  - String operators
  - Array operators

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

Arithmetic Operator: Example

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

<?php
$x = 10;
$y = 6;

echo $x - $y;
?>

</body>
</html>
```



Assignment Operators

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

Assignment Operator: Example

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

<?php
$x = 20;
$x += 100;

echo $x;
?>

</body>
</html>
```

Comparison Operators

Operator	Description	
==	Equal to	$x == $y
===	Equal value and equal type	$x === $y
!=	Not equal	$x != $y
<>	Not equal	$x <> $y
!==	Not equal value or not equal type	$x !== $y
>	Greater than	$x > $y
<	Less than	$x < $y
>=	Greater than or equal to	$x >= $y
<=	Less than or equal to	$x <= $y

Operator	Description	
<=>	Spaceship	$x <=> $y

It returns -1, 0 or 1 respectively less than, equal to, or greater than

Comparison Operator: Example

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

<?php
$x = 100;
$y = "100";

Echo $x === $y;
?>
</body> </html>
```

Output:

(blank)

Increment / Decrement Operator

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

Increment / Decrement Operator: Example

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

<?php
$x = 10;
echo ++$x;
?>

</body>
</html>
```

Logical Operator

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

Logical Operator: Example

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

<?php
$x = 100;
$y = 50;

if ($x == 100 && $y == 50) {
    echo "Hello world!";
}
?>

</body>
</html>
```

Comments in PHP

```
<html>
<body>
<?php
// This is a single-line comment

# This is also a single-line comment

/*
This is a multiple-lines comment block
that spans over multiple
lines
*/

// You can also use comments to leave out parts of a code line
$x = 5 /* + 15 */ + 5;
echo $x;
?>  </body> </html>
```

PHP Data Types

- String
- Integer
- Float (also called double)
- Boolean

Scalar Data types

- Array
- Object

Compound Data Types

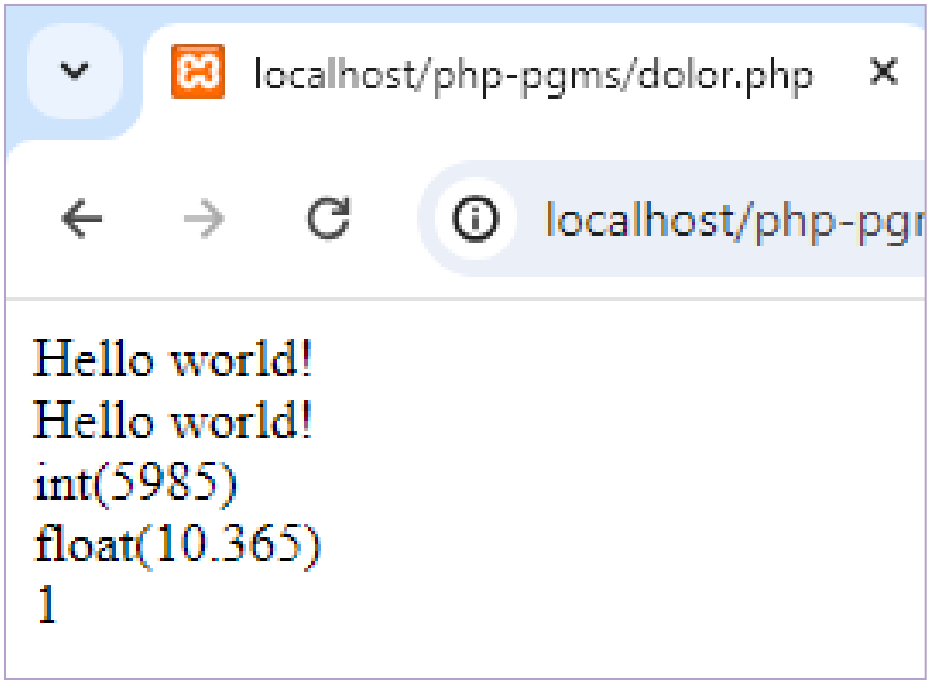
- NULL
- Resource

Special Data types

The PHP var_dump() function returns the data type and value

String, Integer, Float and Boolean: Example

```
<?php
$x = "Hello world!";
$y = 'Hello world!';
echo $x;  echo "<br>";
echo $y;
$x = 5985;
var_dump($x);
echo "<br>";
$x = 10.365;
var_dump($x);
echo "<br>";
$m=true;
echo $m;
?>
```



Array: Example

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

<?php
$animals = array(”Elephant ",”Dog","Tiger");
var_dump($animals);
?>

</body>
</html>
```

Output:

array(3) { [0]=> string(8) ”Elephant" [1]=> string(3) ”Dog" [2]=> string(5) "Tiger" }

Object: Example

- An object is a data type which stores data and information on how to process that data.

```
<?php
class Car {
   function Car() {
      $this->model = "VW";
   }
}
// create an object
$c1 = new Car();

// show object properties
echo $c1->model;
?>
```

Output:
VW

NULL :Example

- 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.
- **Tip: If a variable is created without a value, it is automatically assigned a value of NULL.**

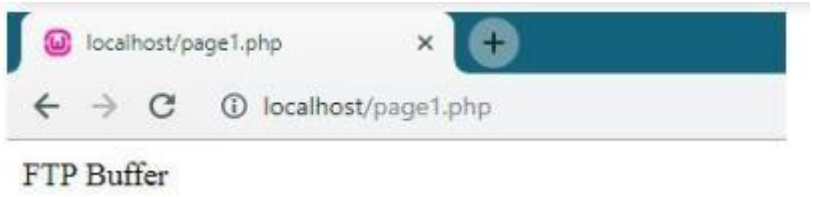
```
<!DOCTYPE html>
<html>
<body>

<?php
$x = "Hello world!";
$x = null;
var_dump($x);
?>

</body>
</html>
```

Output:
null

Resource: Example

- It refers the external resources like database connection, FTP connection, file pointers, etc.
- In simple terms, a resource is a special variable which carrying a reference to an external resource.
- Example:

```
<?php
   $conn = ftp_connect("127.0.0.1") or die("Could not connect");
   echo get_resource_type($conn);
?>
```



Conditional Statement

- **if statement** - executes some code if one condition is true
- **if...else statement** - executes some code if a condition is true and another code if that condition is false
- **if...elseif....else statement** - executes different codes for more than two conditions
- **switch statement** - selects one of many blocks of code to be executed

if statement

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

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

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

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

</body>
</html>
```

```
$x = 15;

If($x > 10){
    echo “X is Greater”;
}
```

if...else Statement

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

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

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

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

</body>
</html>
```

```
$x = 15;

If($x > 30){
    echo “X is Greater”;
}else{
    echo “X is Smaller”;
  }
```

if...elseif....else Statement

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

<?php
$t = date("H");
echo "<p>The hour (of the server) is " . $t;
echo ", and will give the following message:</p>";

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

</body>
</html>
```

```
If(Condition 1){
    Statement 1
}elseif (Condition 2){
    Statement 2
} else {
    Default Statement
}
```

switch Statement

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

<?php
$favcolor = "red";

switch ($favcolor) {
    case "red":
        echo "Your favorite color is red!";
        break;
    case "blue":
        echo "Your favorite color is blue!";
        break;
    case "green":
        echo "Your favorite color is green!";
        break;
    default:
        echo "Your favorite color is neither red, blue, nor green!";
}
?>

</body> </html>
```

PHP Loops

- **while** - loops through a block of code as long as the specified condition is true
- **do...while** - loops through a block of code once, and then repeats the loop as long as the specified condition is true
- **for** - loops through a block of code a specified number of times
- **foreach** - loops through a block of code for each element in an array

while Loop

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

<?php
$x = 1;

while($x <= 5) {
  echo "The number is: $x <br>";
  $x++;
}
?>

</body>
</html>
```

do...while Loop

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

<?php
$x = 1;

do {
    echo "The number is: $x <br>";
    $x++;
} while ($x <= 5);
?>

</body>
</html>
```

for Loop

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

<?php
for ($x = 0; $x <= 10; $x++) {
  echo "The number is: $x <br>";
}
?>

</body>

  </html>
```

Types of array in PHP

Indexed Array

```
$colors = array(
    "red",
    "green",
    "blue"
);
```

Associative Array

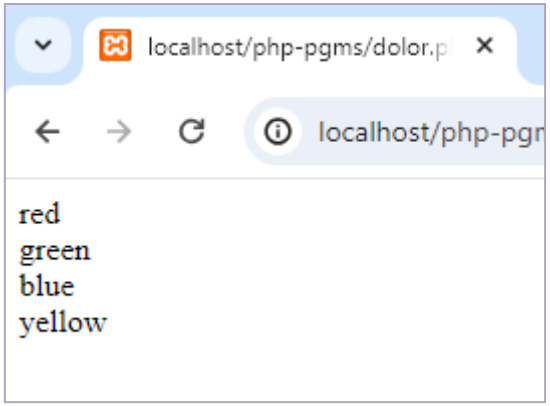
```
$age = array(
    "Bill"=>10,
    "Joe"=> 20,
    "Peter"=> 30
);
```

foreach loop

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

```
foreach($array as $value){
}
```

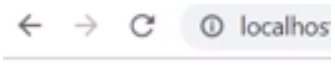
```
<!DOCTYPE html>
<html>
<body>

<?php
$colors = array("red", "green", "blue", "yellow");

foreach ($colors as $value) {
  echo "$value <br>";
}
?>

</body>
</html>
```



foreach loop for Associative array:

```php
<?php

$age = [
        "bill" => 25,
        "steve" => 28,
        "elon" => 22,
      ];

foreach($age as $key => $value){
  echo $key . " = " .$value . "<b
}
```

```php
<?php

$age = [
        "bill" => 25,
        "steve" => 28,
        "elon" => 22,
      ];

echo "<ul>";
foreach($age as $key => $value){
  echo "<li>$key = $value </li>";
}
echo "</ul>";
```



bill = 25
steve = 28
elon = 22

- bill = 25
- steve = 28
- elon = 22

PHP Functions

- PHP function is a piece of code that can be reused many times. It can take input as argument list and return value. There are **thousands of built-in functions in PHP**.
- In PHP, we can define **Conditional function**, **Function within Function** and **Recursive function** also.
- Advantage of PHP Functions:
- **Code Reusability**:
  - PHP functions are defined only once and can be invoked many times, like in other programming languages.
- **Less Code**:
  - It saves a lot of code because you don't need to write the logic many times. By the use of function, you can write the logic only once and reuse it.
- **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.

User Defined Functions

- A function is a block of statements that can be used repeatedly in a program.
- A function will not execute immediately when a page loads.
- A function will be executed by a call to the function.

Call by value:

```
<?php
function trialFunction($Name, $Marks)
{
   echo "$Name got $Marks .<br>";
}
trialFunction("Amit","97");
trialFunction("Ajay","78");
trialFunction("Zoya","83");
 ?>
```

PHP Call By Reference

- Value passed to the function doesn't modify the actual value by default (call by value).

  But we can do so by passing value as a reference.
- By default, value passed to the function is call by value.
- To pass value as a reference, you need to use ampersand (&) symbol before the argument name.
- **Example:**

```
<?php
function adder(&$str2)
{
   $str2 .= 'Call By Reference';
}
$str = 'Hello ';
adder($str);
echo $str;
?>
```

Output:

Hello Call By Reference

PHP Function: Default Argument Value

```
<?php
function sayHello($name="Sonoo"){
echo "Hello $name<br/>";
}
sayHello("Rajesh");
sayHello();//passing no value
sayHello("John");
?>
```

Output:

Hello Rajesh
Hello Sonoo
Hello John

PHP Function: Returning Value

```
<?php
function cube($n)
{
return $n*$n*$n;
}
echo "Cube of 3 is: ".cube(3);
?>
```

Output:

Cube of 3 is: 27

PHP Recursive Function

```
<?php
function display($number) {
    if($number<=5){
     echo "$number <br/>";
     display($number+1);
    }
}

display(1);
?>
```

Output:

1
2
3
4
5

PHP String Functions

- **strlen()**

  returns the length of a string
- **str_word_count()**

  counts the number of words in a string
- **strrev()**

  reverses a string
- **strpos()**

  searches for a specific text within a string(returns position of first match)
- **str_replace()**

  replaces some characters with some other characters in a string.

PHP Arrays

- PHP array is an ordered map (contains value on the basis of key). It is used to hold multiple values of similar type in a single variable.

- **Advantage of PHP Array**
  - **Less Code**:

    We don't need to define multiple variables.
  - **Easy to traverse**:

    By the help of single loop, we can traverse all the elements of an array.

- **PHP Array Types**

1. Indexed Array
2. Associative Array
3. Multidimensional Array

Example: Arrays

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

Output:

Season are: summer, winter, spring and autumn

```
<?php
$salary=array("Sonoo"=>"350000","John"=>"450000","Kartik"=>"200000");
echo "Sonoo salary: ".$salary["Sonoo"]."<br/>";
echo "John salary: ".$salary["John"]."<br/>";
echo "Kartik salary: ".$salary["Kartik"]."<br/>";
?>
```

Output:

Sonoo salary: 350000
John salary: 450000
Kartik salary: 200000

Example: Arrays

```
<?php
$emp = array
  (
  array(1,"sonoo",400000),
  array(2,"john",500000),
  array(3,"rahul",300000)
  );

for ($row = 0; $row < 3; $row++) {
  for ($col = 0; $col < 3; $col++) {
  echo $emp[$row][$col]."   ";
  }
  echo "<br/>";
}
?>
```

Output:

1 sonoo 400000
2 john 500000
3 rahul 300000

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

Array Operators: Example

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

<?php
$x = array("a" => "red", "b" => "green");
$y = array("c" => "blue", "d" => "yellow");

var_dump($x != $y);
?>

</body>
</html>
```

Array Functions(1)

PHP array_change_key_case() function

- **Syntax**

array array_change_key_case (**array** $array [, int $case = CASE_LOWER])

- Note: It changes case of key only.

- **Example**

```
<?php
$salary=array("Sonoo"=>"550000","Vimal"=>"250000","Ratan"=>"200000");
print_r(array_change_key_case($salary,CASE_UPPER));
?>
```

Output:

Array ([SONOO] => 550000 [VIMAL] => 250000 [RATAN] => 200000)

Array Functions(2)

PHP array_chunk() function

- PHP array_chunk() function splits array into chunks. By using array_chunk() method, you can divide array into many parts.
- **Syntax**

array array_chunk (**array** $array , int $size [, bool $preserve_keys = false])

- **Example**

```
<?php
$salary=array("Sonoo"=>"550000","Vimal"=>"250000","Ratan"=>"200000")
  print_r(array_chunk($salary,2));
?>
```

Output:

Array ([0] => Array ([0] => 550000 [1] => 250000)
[1] => Array ([0] => 200000))

Array Functions(3)

PHP count() function

- PHP count() function counts all elements in an array.
- **Syntax**

int count (mixed $array_or_countable [, int $mode = COUNT_NORMAL])

- **Example**

```
<?php
$season=array("summer","winter","spring","autumn");
echo count($season);
?>
```

Output:

4

Array Functions(4)

PHP sort() function

- PHP sort() function sorts all the elements in an array.
- **Syntax**

bool sort (**array** &$array [, int $sort_flags = SORT_REGULAR])

- **Example**

```
<?php
$season=array("summer","winter","spring","autumn");
sort($season);
foreach( $season as $s )
{
  echo "$s<br />";
}
?>
```

Output:

autumn
spring
summer
winter

Array Functions(5)

PHP array_reverse() function

- PHP array_reverse() function returns an array containing elements in reversed order.
- **Syntax**

array array_reverse (**array** $array [, bool $preserve_keys = false])

- **Example**

```
<?php
$season=array("summer","winter","spring","autumn");
$reverseseason=array_reverse($season);
foreach( $reverseseason as $s )
{
  echo "$s<br />";
}
?>
```

Output:

autumn
spring
winter
summer

Array Functions(6)

PHP array_search() function

- PHP array_search() function searches the specified value in an array. It returns key if search is successful.
- **Syntax**

mixed array_search (mixed $needle , **array** $haystack [, bool $strict = false])

- **Example**

```
<?php
$season=array("summer","winter","spring","autumn");
$key=array_search("spring",$season);
echo $key;
?>
```

Output:

2

Array Functions(7)

PHP array_intersect() function

- PHP array_intersect() function returns the intersection of two array. In other words, it returns the matching elements of two array.
- **Syntax**

array array_intersect (**array** $array1 , **array** $array2 [, **array** $...])

- **Example**

```
<?php
$name1=array("sonoo","john","vivek","smith");
$name2=array("umesh","sonoo","kartik","smith");
$name3=array_intersect($name1,$name2);
foreach( $name3 as $n )
{
  echo "$n<br />";
}
?>
```

Output:

sonoo
smith

PHP String

- PHP string is a sequence of characters i.e., used to store and manipulate text.
- PHP supports only 256-character set and so that it does not offer native Unicode support.
- There are various ways to specify a string literal in PHP.
  - **single quoted**
  - **double quoted**
  - **heredoc syntax**
  - **newdoc syntax (since PHP 5.3)**

PHP String

Single Quoted

- We can create a string in PHP by enclosing the text in a single-quote.
- It is the easiest way to specify string in PHP.
- For specifying a literal single quote, escape it with a backslash (\) and to specify a literal backslash (\) use double backslash (\\).
- **Example:**

```
<?php
    $str='Hello text within single quote';
    echo $str;
?>
```

Output:

Hello text within single quote

PHP String

Double Quoted

- In PHP, we can specify string through enclosing text within double quote also. But escape sequences and variables will be interpreted using double quote PHP strings.
- **Example 1**

```
<?php
$str="Hello text within double quote";
echo $str;
?>
```

Output:

Hello text within double quote

String Function(1)

PHP strtoupper() function

- The strtoupper() function returns string in uppercase letter.
- **Syntax**

string strtoupper (string $string)

- **Example**

```
<?php
$str="My name is abc";
$str=strtoupper($str);
echo $str;
?>
```

Output:

MY NAME IS ABC

String Function(2)

PHP ucfirst() function

- The ucfirst() function returns string converting first character into uppercase. It doesn't change the case of other characters.
- **Syntax**

string ucfirst (string $str)

- **Example**

```
<?php
$str="my name is abc";
$str=ucfirst($str);
echo $str;
?>
```

Output:

My name is abc

String Function(3)

PHP ucwords() function

- The ucwords() function returns string converting first character of each word into uppercase.
- **Syntax**

string ucwords (string $str)

- **Example**

```
<?php
$str="my name is Sonoo jaiswal";
$str=ucwords($str);
echo $str;
?>
```

Output:

My Name Is Sonoo Jaiswal

String Function(4)

PHP strrev() function

- The strrev() function returns reversed string.
- **Syntax**

string strrev (string $string)

- **Example**

```
<?php
$str="my name is Sonoo jaiswal";
$str=strrev($str);
echo $str;
?>
```

Output:

lawsiaj oonoS si eman ym

String Function(5)

PHP strlen() function

- The strlen() function returns length of the string.
- **Syntax**

int strlen (string $string)

- **Example**

```
<?php
$str="my name is Sonoo jaiswal";
$str=strlen($str);
echo $str;
?>
```

Output:

24

String: Example

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

<?php
echo strlen("Hello world!");
echo str_word_count("Hello world!");
echo strrev("Hello world!");
echo strpos("Hello world!", "world");
echo str_replace("world", "Dolly", "Hello world!");
?>

</body>
</html>
```

String Operator

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

String Operator: Example

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

<?php
$txt1 = "Hello";
$txt2 = " world!";
echo $txt1 . $txt2;
?>

</body>
</html>
```

HTML Forms

- HTML Forms are required, when you **want to collect some data from the site visitor.**
- For example, during user registration you would like to collect information such as name, email address, credit card, etc.
- A form will **take input from the site visitor and then will post it to a back-end application** such as CGI, ASP Script or PHP script etc.
- There are various form elements available like text fields, textarea fields, drop-down menus, radio buttons, checkboxes, etc.

Forms Syntax

<form **action** = "Script URL“ **method** = "GET|POST">

form elements like input, textarea etc.

</form>

action

- Backend script ready to process your passed data.

method

- Method to be used to upload data. The most frequently used are GET and POST methods.

Forms Syntax

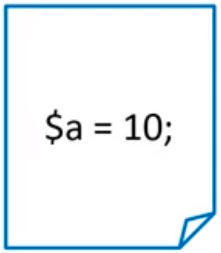
Form elements

Text Input Controls , Checkboxes Controls, Radio Box Controls, Select Box Controls, File Select boxes , Hidden Controls, Clickable Buttons , Submit and Reset Button

target

- Specify the target window or frame where the result of the script will be displayed. It takes values like _blank, _self, _parent etc.

PHP : Super Global Variables



File1.php

File2.php

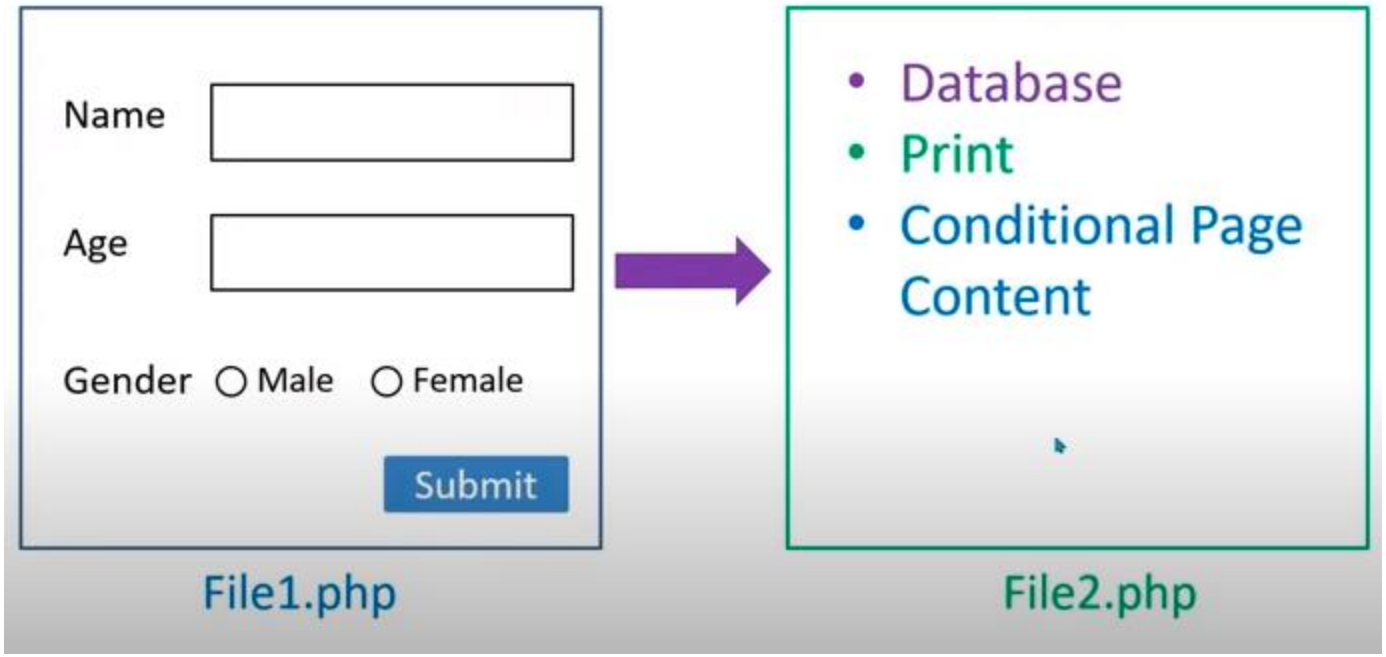
- $_GET
- $_POST
- $_REQUEST
- $_SERVER
- $_SESSION
- $_COOKIE
- $_FILES

GET Method

- The default method when submitting form data is GET.
- However, when GET is used, the submitted form data will be **visible in the page address field**:

When to use GET??

- Appends form-data into the URL in name/value pairs
- The length of a URL is limited (about 3000 characters)
- Never use GET to send sensitive data! (will be visible in the URL)
- Useful for form submissions where a user wants to bookmark the result
- GET is better for non-secure data, like query strings in Google


Name
Age
Gender Male Female
Submit
File1.php
Database
Print
Conditional Page Content
File2.php

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

<p>This form will be submitted using the GET method:</p>

<form method="GET“   target="_blank" >
  First name:<br>
  <input type="text" name="firstname" >
  <br>
  Last name:<br>
  <input type="text" name="lastname" >
  <br><br>
  <input type="submit" value="Submit">
</form>

</body>
</html>
```

```html
<html>
<head><title></title></head>
<body>
<p>This form will be submitted using the GET method:</p>
<form method="GET" target="_blank" action="testform.php" >
 First name:<br>
 <input type="text" name="fname" >
 <br>
 Last name:<br>
 <input type="text" name="lname">
 <br><br>
 <input type="submit" value="Submit">
</form>
</body>
</html>
```

This form will be submitted using the GET method:

First name:

Last name:

Submit

This form will be submitted using the GET method:

First name:
veena
Last name:
Badgujar

Submit

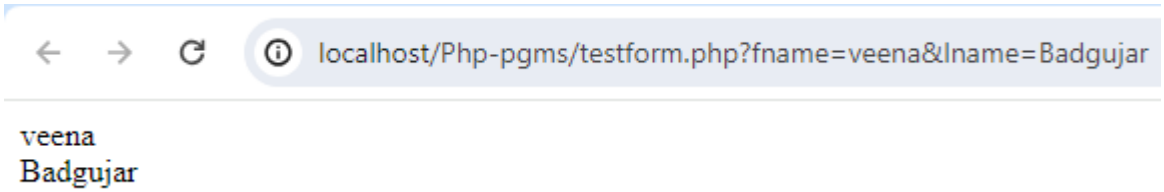
Testform.php

```php
<?php

echo "<pre>";
print_r ($_GET);
echo "</pre>";

?>
```

localhost/Php-pgms/testform.php?fname=veena&lname=Badgujar

```
Array
(
    [fname] => veena
    [lname] => Badgujar
)
```

```php
<?php

echo $_GET['fname'];
echo "<br>";
echo $_GET['lname'];

?>
```



POST Method

- Always use POST if the form data contains sensitive or personal information.
- The POST method does not display the submitted form data in the page address field.
- **When to use POST??**

➢ POST has no size limitations, and can be used to send large amounts of data.

➢ Form submissions with POST cannot be bookmarked

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

<p>This form will be submitted using the POST method:</p>

<form method="POST“ target="_blank" >
  First name:<br>
  <input type="text" name="firstname" value="Mickey">
  <br>
  Last name:<br>
  <input type="text" name="lastname" value="Mouse">
  <br><br>
  <input type="submit" value="Submit">
</form>

</body>
</html>
```

PHP REQUEST method

- $_REQUEST is a PHP super global variable which contains submitted form data, and all cookie data.

- In other words, $_REQUEST is an array containing data from $_GET, $_POST, and $_COOKIE.

- **Using $_REQUEST on $_POST Requests**
- POST request are usually data submitted from an HTML form.
- Example:
- PHP file-

```
<?php

echo $_REQUEST['fname'];
echo "<br>";
echo $_REQUEST['lname'];
echo "<pre>";

?>
```

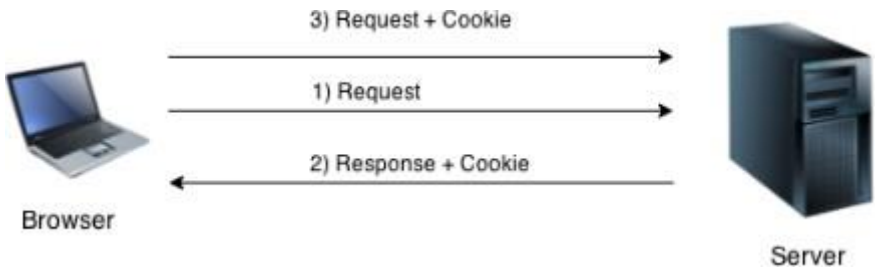
$_SERVER

- $_SERVER is a PHP super global variable which holds information about headers, paths, and script locations.

```php
<?php
echo "<pre>";
print_r($_GET);
echo "</pre>";


echo "<pre>";
print_r($_SERVER);
echo "</pre>";
?>
```

PHP Cookie

- PHP cookie is a small piece of information which is stored at client browser. It is used to recognize the user.
- Cookie is created at server side and saved to client browser.
- Each time when client sends request to the server, cookie is embedded with request. Such way, cookie can be received at the server side.
- In short, cookie can be created, sent and received at server end.



PHP Cookie (contd..)

PHP setcookie() function –create cookie

- PHP setcookie() function is used to set cookie with HTTP response. Once cookie is set, you can access it by $_COOKIE superglobal variable.
- **Syntax**

bool setcookie (string $name [, string $value [, int $expire = 0 [, string $path [, string $domain [, bool $secure = false [, bool $httponly = false]]]]]])

setcookie(name, value, expire, path, domain, secure, httponly)

- **Example**
- **expire-**
  - 1hr=60*60=3600 sec
  - 24hr=3600*24=86,400sec
  - Time()+(86400) //for 24 hrs
  - Time()+(86400 *30) //for 1 month
- **Path- "/"** // you can access it from any page.
- Domain= //access cookies from specific domain
- Secure=true/false
- Httponly=true/false

```
setcookie("CookieName", "CookieValue");
/* defining name and value only*/

setcookie("CookieName", "CookieValue", time()+1*60*60);
//using expiry in 1 hour(1*60*60 seconds or 3600 seconds)

setcookie("CookieName", "CookieValue", time()+1*60*60, "/mypath/", "
mydomain.com", 1);
```

PHP Cookie (contd..)

PHP $_COOKIE

- PHP $_COOKIE superglobal variable is used to get cookie.
- **Example**

```
$value=$_COOKIE["CookieName"];//returns cookie value
```

- **Example**
- **1) create cookie:**
- **$cookie_name=“user”;**
- **$cookie_value=“John”;**

setcookie(cookie_name, cookie_value,time()+(86400),”/”);
// defining 4 parameters.

2) View cookie:
$_COOKIE[name];

PHP $_COOKIE

- PHP $_COOKIE superglobal variable is used to get cookie.
- **Example**

$value=$_COOKIE["CookieName"];//returns cookie value

PHP Cookie (contd..)

```
<?php
$cookie_name="user";
$cookie_value="John";

setcookie(cookie_name, cookie_value,time()+(86400),”/”
?>
<html>
<body>
<?php
echo $_COOKIE[$cooke_name];
?>
</body>
</html>
```

Example: Cookie

```
<?php
setcookie("user", "Sonoo");
?>
<html>
<body>
<?php
if(!isset($_COOKIE["user"])) {
   echo "Sorry, cookie is not found!";
}
else
{
   echo "<br/>Cookie Value: " . $_COOKIE["user"];
}
?>
</body>
</html>
```

Output:

Sorry, cookie is not found!

Firstly cookie is not set. But, if you *refresh* the page, you will see cookie is set now.

Cookie Value: Sonoo

PHP Cookie (contd..)

PHP Delete Cookie

- If you set the expiration date in past, cookie will be deleted.
- Example:

```
<?php
setcookie ("CookieName", "", time() -3600);
// set the expiration date to one hour ago
?>
```

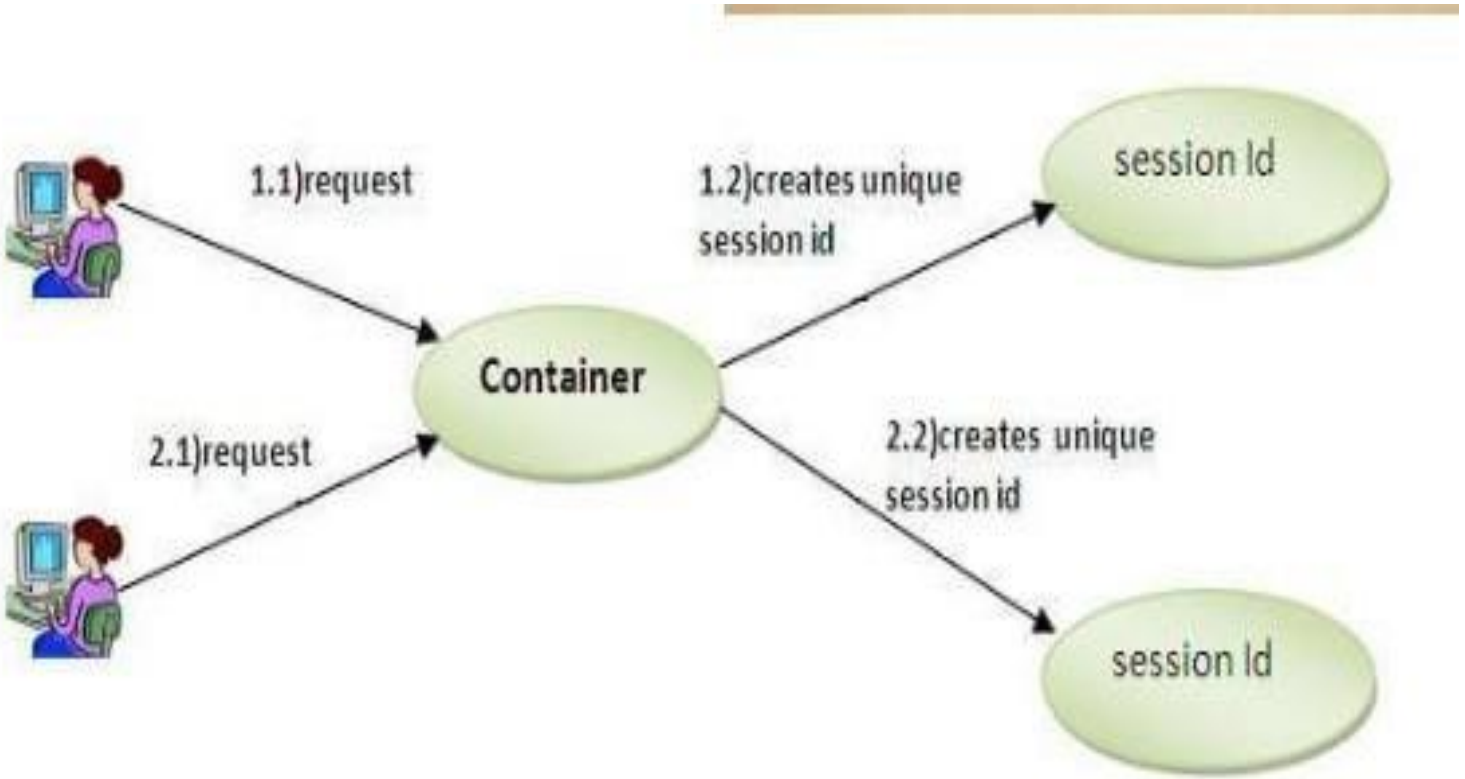
PHP Session

- PHP session is used to store and pass information from one page to another temporarily (until user close the website).

- PHP session technique is widely used in shopping websites where we need to store and pass cart information e.g. username, product code, product name, product price etc from one page to another.

- PHP session creates unique user id for each browser to recognize the user and avoid conflict between multiple browsers.

PHP Session


1.1)request
1.2)creates unique session id
session Id
Container
2.1)request
2.2)creates unique session id
session Id

PHP Session

PHP session_start() function

- PHP session_start() function is used to start the session. It starts a new or resumes existing session. It returns existing session if session is created already. If session is not available, it creates and returns new session.
- **Syntax**

bool session_start (void)

- **Example**

```
session_start();
```

Step 1 : session_start();

Step 2 : $_SESSION[name] = value; Set Session name & value

Step 3 : echo $_SESSION[name]; Get Session value

Delete Session

Step 1 : session_unset(); Remove all session variables

Step 2 : session_destroy(); Destroy the session

PHP Session

PHP $_SESSION

- PHP $_SESSION is an associative array that contains all session variables. It is used to set and get session variable values.
- **Example: Store information**

$_SESSION["user"] = "Sachin";

- **Example: Get information**

echo $_SESSION["user"];

Example: PHP Session

Session1.php

```
<?php
session_start();
?>
<html>
<body>
<?php
$_SESSION["user"] = "Sachin";
echo "Session information are set successfully.<br/>";
?>
<a href="session2.php">Visit next page</a>

</body>
</html>
```

Session2.php

```
<?php
session_start();
?>
<html>
<body>
<?php
echo "User is: ".$_SESSION["user"];
?>
</body>
</html>
```

PHP Session Counter Example

```
<?php
  session_start();

  if (!isset($_SESSION['counter'])) {
    $_SESSION['counter'] = 1;
  } else {
    $_SESSION['counter']++;
  }
  echo ("Page Views: ".$_SESSION['counter']);
?>
```

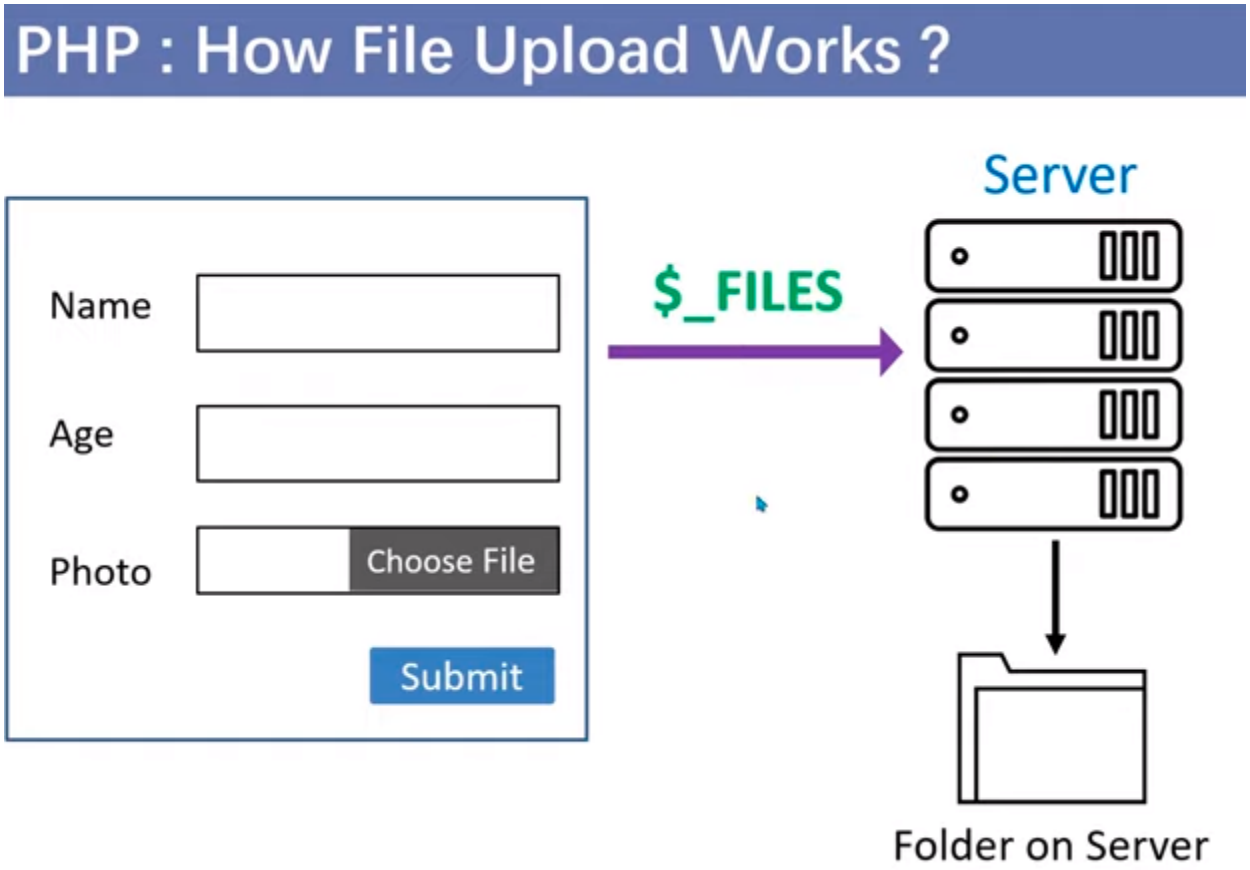
PHP Session

PHP Destroying Session

- PHP session_destroy() function is used to destroy all session variables completely.

```
<?php
session_start();
session_destroy();
?>
```

$_File Variable




```
<input type="file" name="image" />
```

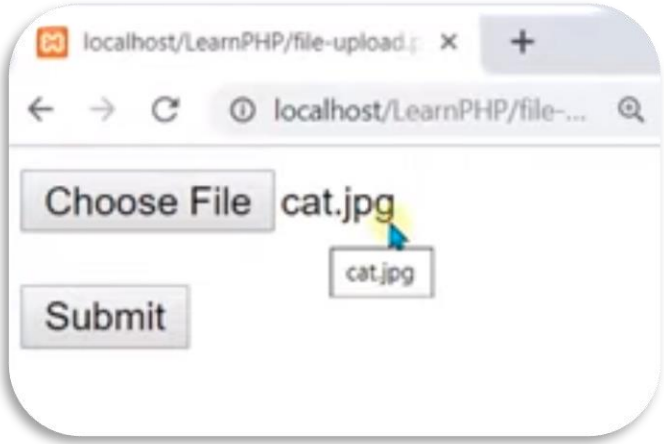
```
$_FILES['image']
```

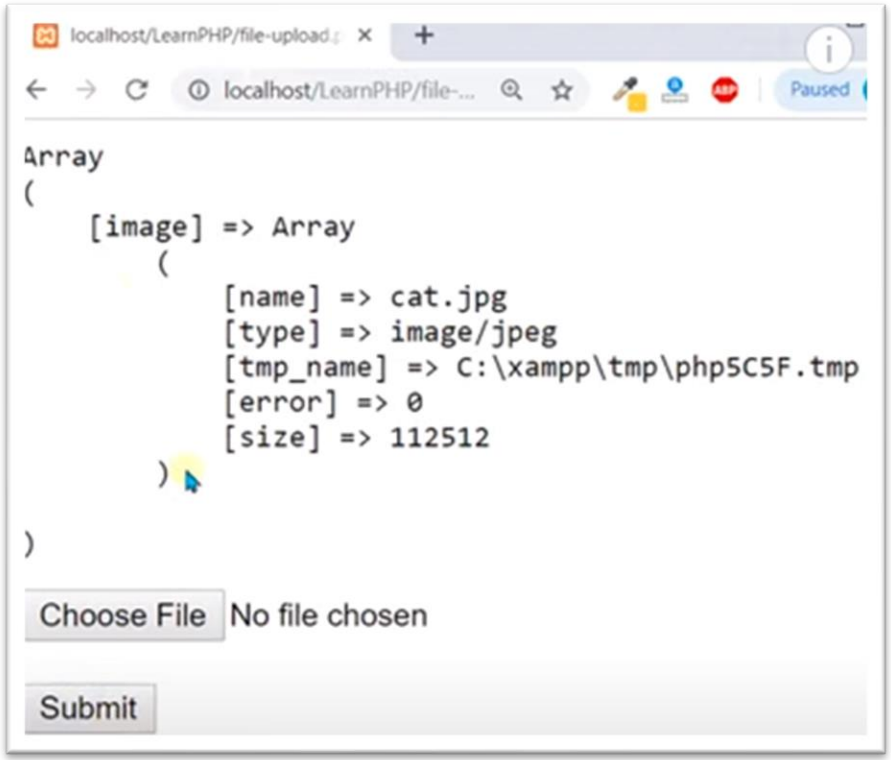
- name
- size
- tmp_name
- type JPG / PNG / GIF

```
move_uploaded_file(file, dest)
```

example

```php
<?php
 if(isset($_FILES['image'])){
    echo "<pre>";
    print_r($_FILES);
    echo "</pre>";
 }

?>
<html>
    <body>
        <form action="" method="POST" enctype="multipart/form-data">
            <input type="file" name="image" /><br><br>
            <input type="submit"/>
        </form>
    </body>
</html>
```


localhost/LearnPHP/file-upload.
localhost/LearnPHP/file-...
Choose File
cat.jpg
cat.jpg
Submit

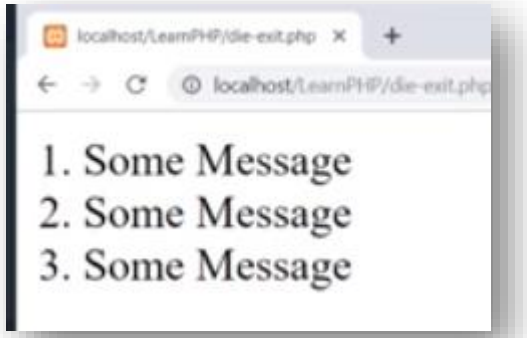

localhost/LearnPHP/file-upload.
localhost/LearnPHP/file-...
Paused
Array
(
[image] => Array
(
[name] => cat.jpg
[type] => image/jpeg
[tmp_name] => C:\xampp\tmp\php5C5F.tmp
[error] => 0
[size] => 112512
)
)
Choose File
No file chosen
Submit

```php
if(isset($_FILES['image'])){

  $file_name = $_FILES['image']['name'];
  $file_size = $_FILES['image']['size'];
  $file_tmp = $_FILES['image']['tmp_name'];
  $file_type = $_FILES['image']['type'];

  if(move_uploaded_file($file_tmp,"upload-images/". $file_name)){
    echo "Successfully uploaded.";
  }else{
    echo "Could not upload the file.";
  }
}
```

Die and Exit functionsin PHP

```php
<?php

echo "1. Some Message <br>";
echo "2. Some Message <br>";
echo "3. Some Message <br>";
echo "4. Some Message <br>";
echo "5. Some Message <br>";

?>
```


```php
<?php

echo "1. Some Message <br>";
echo "2. Some Message <br>";
echo "3. Some Message <br>";
die();
echo "4. Some Message <br>";
echo "5. Some Message <br>";

?>
```



File Handling

PHP readfile()-
The readfile() function is useful if all you want to do is open up a file and read its contents.

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

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

</body>
</html>
```

PHP File Handling

- PHP File System allows us to create file, read file line by line, read file character by character, write file, append file, delete file and close file.
- **PHP Open File - fopen()**
- The PHP fopen() function is used to open a file
- **Syntax**

fopen (string $filename , string $mode)

- **Example**

```
<?php
$handle = fopen("c:\\folder\\file.txt", "r") or die("Unable to open file!");;
?>
```

Modes	Description
r	**Open a file for read only**. File pointer starts at the beginning of the file
w	**Open a file for write only**. Erases the contents of the file or creates a new file if it doesn't exist. File pointer starts at the beginning of the file
a	**Open a file for write only**. The existing data in file is preserved. File pointer starts at the end of the file. Creates a new file if the file doesn't exist
x	**Creates a new file for write only**. Returns FALSE and an error if file already exists
r+	**Open a file for read/write**. File pointer starts at the beginning of the file
w+	**Open a file for read/write**. Erases the contents of the file or creates a new file if it doesn't exist. File pointer starts at the beginning of the file
a+	**Open a file for read/write**. The existing data in file is preserved. File pointer starts at the end of the file. Creates a new file if the file doesn't exist
x+	**Creates a new file for read/write**. Returns FALSE and an error if file already exists

PHP File Handling

PHP Close File - fclose()

- The PHP fclose() function is used to close an open file pointer.
- **Syntax**

fclose (file)

- **Example**

```
<?php
fclose($handle);
?>
```

PHP File Handling

PHP Read File - fread()

- The PHP fread() function is used to read the content of the file. It accepts two arguments: resource and file size.
- **Syntax**

string fread (resource $handle , int $length)

- **Example**

```
<?php
$filename = "c:\\myfile.txt";
$handle = fopen($filename, "r");//open file in read mode

$contents = fread($handle, filesize($filename));//read file

echo $contents;//printing data of file

fclose($handle);//close file
?>
```

Output:

hello php file

PHP File Handling

PHP Write File - fwrite()

- The PHP fwrite() function is used to write content of the string into file.
- **Syntax**

int fwrite (resource $handle , string $string [, int $length])

- **Example**

```
<?php
$fp = fopen('data.txt', 'w');//open file in write mode
fwrite($fp, 'hello ');
fwrite($fp, 'php file');
fclose($fp);

echo "File written successfully";
?>
```

Output:

File written successfully

PHP File Handling

PHP Delete File - unlink()

- The PHP unlink() function is used to delete file.
- **Syntax**

bool unlink (string $filename [, resource $context])

- **Example**

```
<?php
unlink('data.txt');

echo "File deleted successfully";
?>
```

PHP File Handling

PHP Close File - fclose()

- The PHP fclose() function is used to close an open file pointer.
- **Syntax**

ool fclose (resource $handle)

- **Example**

```
<?php
fclose($handle);
?>
```

PHP File Handling

PHP Read File - fread()

- The PHP fread() function is used to read the content of the file. It accepts two arguments: resource and file size.
- **Syntax**

string fread (resource $handle , int $length)

- **Example**

```
<?php
$filename = "c:\\myfile.txt";
$handle = fopen($filename, "r");//open file in read mode

$contents = fread($handle, filesize($filename));//read file

echo $contents;//printing data of file

fclose($handle);//close file
?>
```

Output:

hello php file

PHP File Handling

PHP Write File - fwrite()

- The PHP fwrite() function is used to write content of the string into file.
- **Syntax**

int fwrite (resource $handle , string $string [, int $length])

- **Example**

```
<?php
$fp = fopen('data.txt', 'w');//open file in write mode
fwrite($fp, 'hello ');
fwrite($fp, 'php file');
fclose($fp);

echo "File written successfully";
?>
```

Output:

File written successfully

PHP File Handling

PHP Delete File - unlink()

- The PHP unlink() function is used to delete file.
- **Syntax**

bool unlink (string $filename [, resource $context])

- **Example**

```
<?php
unlink('data.txt');

echo "File deleted successfully";
?>
```