Unit 4
Server Side Scripting Languages

CO Mapping: CO1,3

By Prof. K. A. Hule
(M.E., PhD*)

IV-Server Side Scripting Languages

Unit Objective

1. Understand the basics of PHP Scripting Language.
2. Apply various operation on Arrays.
3. Understand how to process form data using PHP.
4. Construct small application by using MySQL & PHP.
5. Understand the working of AJAX.

Unit Contents

1. Introduction to PHP, uses of PHP,
2. General syntactic characteristics,
3. Primitives, operations & expressions, output, control statements,
4. Arrays, functions, pattern matching,
5. Form handling, files,
6. Cookies, session tracking,
7. Using MySQL with PHP
8. AJAX: Introduction, Working of AJAX, AJAX processing steps, coding AJAX script

PHP (Hypertext Preprocessor)

- Introduction to PHP
- Uses of PHP
- General syntactic characteristics, output
- Primitives, operations & expressions
- Control statements & Loops
- functions, Arrays, Array functions
- Form handling
- Pattern matching
- Files
- Cookies & session tracking
- Using MySQL with PHP

Introduction to PHP

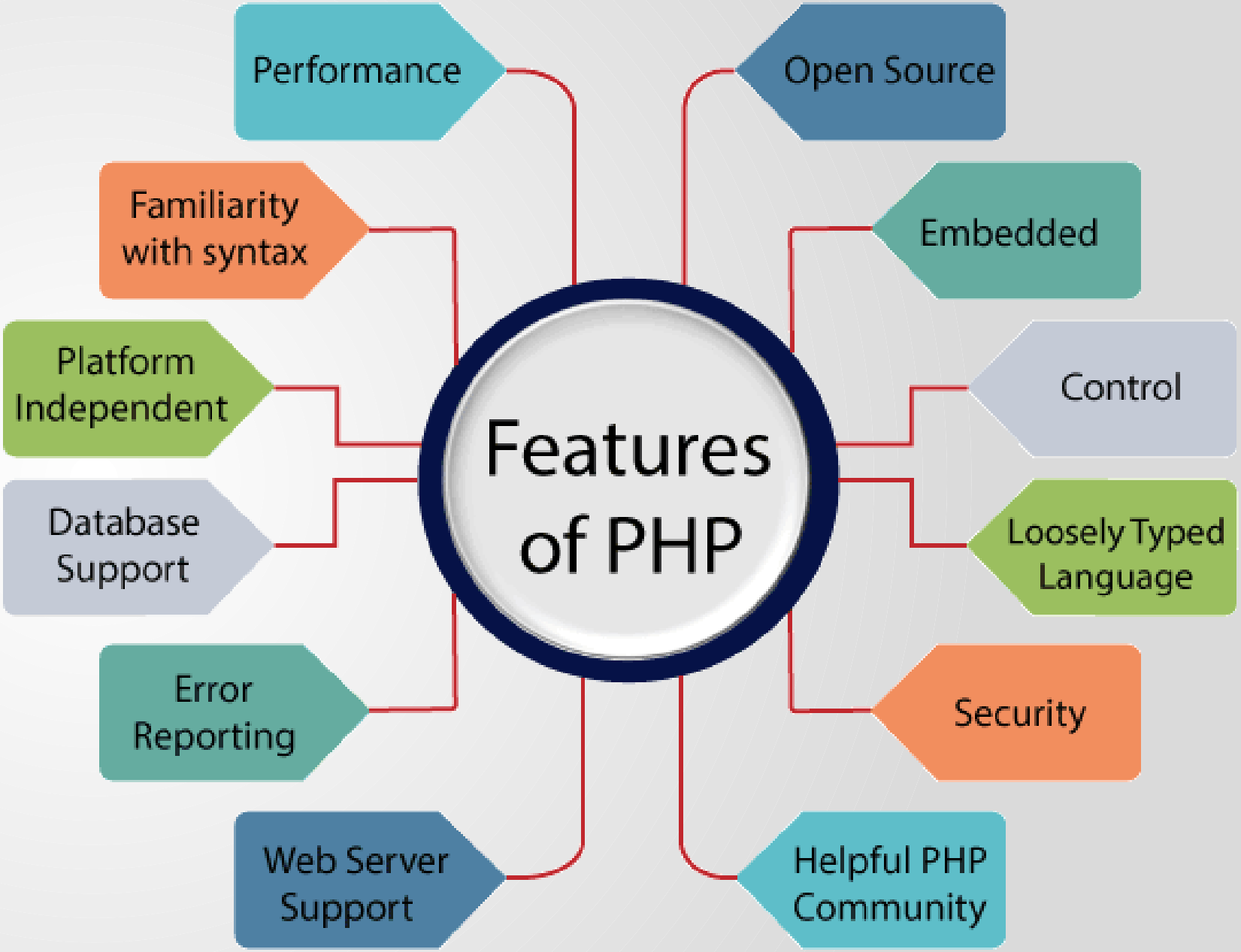
- **Origins & Uses of PHP:**
  - PHP was developed in 1994 by Apache group.
  - PHP originally stood for **Personal Home Page**
  - PHP is an acronym for "**PHP: Hypertext Preprocessor**"
  - PHP is a widely-used, Server-side, open source scripting language
  - PHP scripts are executed on the server
  - PHP is free to download and use
  - Mainly used for form handling & database access.
  - Version History
- **Prerequisite**
  - Before learning PHP, you must have the basic knowledge of **HTML**, **CSS**, & **JavaScript**. So, learn these technologies for better implementation of PHP.

Introduction to PHP

- **What Can PHP Do?**
  - PHP can generate dynamic page content
  - PHP can create, open, read, write, delete, and close files on the server
  - PHP can collect form data
  - PHP can send and receive cookies
  - PHP can add, delete, modify data in your database
  - PHP can be used to control user-access
  - PHP can encrypt data
  - With PHP you are not limited to output HTML. You can output images, PDF files, and even Flash movies. You can also output any text, such as XHTML and XML.

Introduction to PHP

- **Why use php?**
  1. It runs on different platforms such as Windows, Linux, Unix, etc.
  2. This language is very simple to learn and runs efficiently on the server side.
  3. It is compatible with almost all servers used today, such as Apache, IIS, etc.
  4. It supports many databases such as MySQL, Oracle, PostgreSQL etc.
  5. It is perfectly suited for Web development and can be embedded directly into the HTML code.
  6. PHP can also be used to create dynamic web pages.
  7. It is often used together with Apache (web server) on various operating systems. It can be also used with Microsoft's IIS on Windows.
  8. It is open source and it is free downloadable


Features of PHP
Performance
Open Source
Familiarity with syntax
Embedded
Platform Independent
Control
Database Support
Loosely Typed Language
Error Reporting
Security
Web Server Support
Helpful PHP Community

Features of PHP 8

- **What's New in PHP8?**
  1. Union Types
  2. Just In Time Compilation
  3. Named Arguments
  4. Match Expressions
  5. Attributes
  6. Constructor Property Promotion
  7. Nullsafe Operator
  8. Weak Maps
  9. Saner string to number comparisons
  10. Consistent type errors for internal functions

Install PHP

- To install PHP, 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:
  1. **WAMP** for Windows
  2. **LAMP** for Linux
  3. **MAMP** for Mac
  4. **SAMP** for Solaris
  5. **FAMP** for FreeBSD
  6. **XAMPP** (Cross, Apache, MySQL, PHP, Perl) for Cross Platform: It includes some other components too such as FileZilla, OpenSSL, Webalizer, Mercury Mail, etc.

General Syntactic Characteristics

- **What is a PHP File?**
  - PHP files can contain text, HTML, CSS, JavaScript, and PHP code
  - PHP code are executed on the server, and the result is returned to the browser as plain HTML
  - PHP files have extension ".php"

1. All PHP code goes between the php tag. It starts with <?php and ends with ?>. The syntax of PHP tag is given below:

```
<?php
//your code here
?>
```

2. If PHP script is stored in some other file & if it needs to be referred then include construct is used. Like include("myfile.php")
3. The variable names in PHP begin with the $ sign.
4. The $$var (double dollar) is a reference variable that stores the value of the $variable inside it.
5. The comments in PHP can be #, // or /*... */
6. The PHP statements are terminated by semicolon.

XAMPP Server

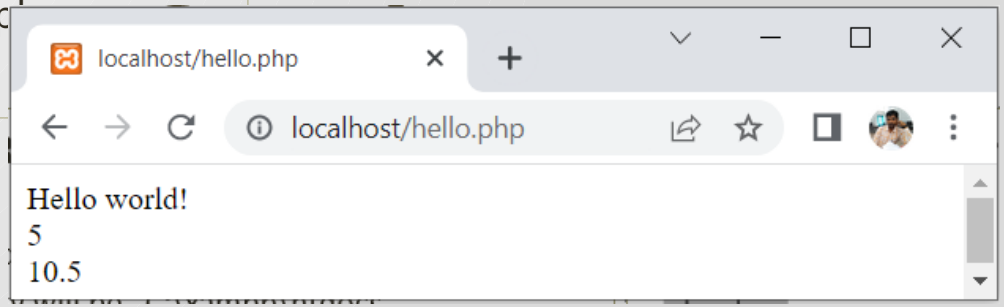
- Xampp is the most popular PHP development environment for Windows, OS X, and Linux platforms.
- Xampp stands for Cross platform(x), Apache(a), Maria db(m), PHP(p), Pearl(p) which is a software distribution server which makes developer's work eaiser for testing and deploying by creating a local web server.
- How to install Xampp?
  - It is completely free and easy to install Apache distribution containing MySQL, PHP, and Perl. First, download XAMP from https://www.apachefriends.org/download.html .
- Bitnami module provides the easiest way to install WordPress, Drupal or Joomla among others on top of

Sample Program

- **How to Run Program**
  - As I'm using window, and my XAMPP server is installed in C drive. So, the path for the htdocs directory will be "C:\xampp\htdocs".
  - **Step1:** Create a simple PHP program like hello world.
  - **Step2:** Save the file with **hello.php** name in the htdocs folder, which resides inside the xampp folder.
  - **Step3:** Run the XAMPP server and start the Apache and MySQL.
  - **Step4:** Now, open the web browser and type localhost http://localhost/hello.php on your browser window.
  - **Step5:** The output for the hello.p will be shown as the screenshot:

- **Example with variable declaration**

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



General Syntactic Characteristics-Output

- **PHP Echo**
- PHP echo is a language construct, not a function. Therefore, you don't need to use parenthesis with it. But if you want to use more than one parameter, it is required to use parenthesis.
- The syntax of PHP echo is given below:

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

- PHP echo statement can be used to print the string, multi-line strings, escaping characters, variable, array, etc. Some important points that you must know about the echo statement are:
  - echo is a statement, which is used to display the output.
  - echo can be used with or without parentheses: echo(), and echo.
  - echo does not return any value.
  - We can pass multiple strings separated by a comma (,) in echo.
  - echo is faster than the print statement.
- Example:

```
<?php
echo ("Hello by PHP echo()");
echo "Hello escape \"sequence\" characters by PHP print";
?>
```

General Syntactic Characteristics-output

- **PHP print**
- Like PHP echo, PHP print is a language construct, so you don't need to use parenthesis with the argument list. Print statement can be used with or without parentheses: print and print(). Unlike echo, it always returns 1.
- The syntax of PHP print is given below:

  int print(string $arg)
- PHP print statement can be used to print the string, multi-line strings, escaping characters, variable, array, etc. Some important points that you must know about the echo statement are:
  - print is 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 the echo statement.
- Example:

```
<?php
print ("Hello by PHP print()");
print "Hello escape \"sequence\" characters by PHP print";
?>
```

PHP Variables

- Variables are the entities that are used for storing the values.
- PHP is a dynamically typed language. That is PHP has no type declaration like Java, C#, etc.
- The value can be assigned to variable in following manner

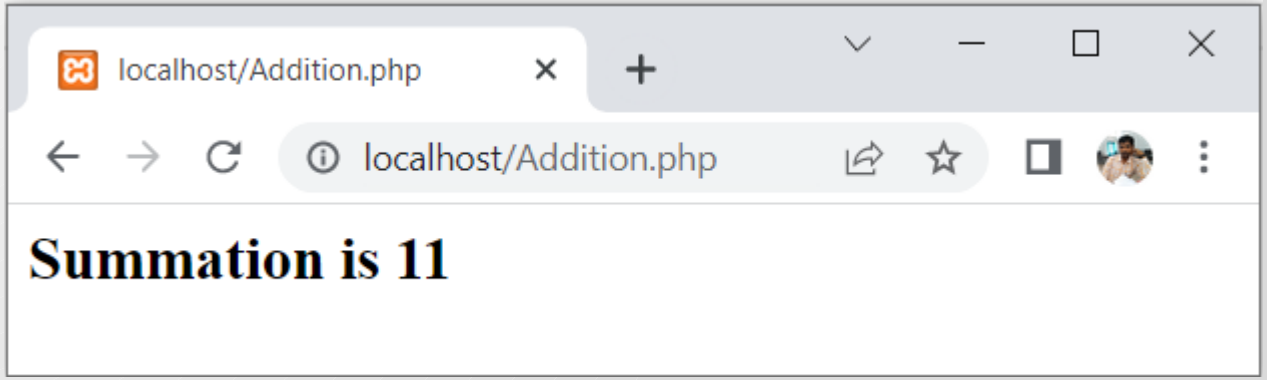
  $variable_name=value;
- If value is not assigned to the variable then by default value is NULL.
- Rules for declaring PHP variable:

1. A variable must start with a dollar ($) sign, followed by the variable name.
2. It can only contain alpha-numeric character and underscore (A-z, 0-9, _).
3. A variable name must start with a letter or underscore (_) character.
4. A PHP variable name cannot contain spaces.
5. One thing to be kept in mind that the variable name cannot start with a number or special symbols.
6. PHP variables are case-sensitive, so $name and $NAME both are treated as different variable.

Sample Program

- **Example**

```
<html>
<body>
    <?php
        $n1=5;
        $n2=6;
        $sum=$n1+$n2;
        Echo "<h2>Summation is ".$sum."</h2>";
     ?>
</body>
</html>
```



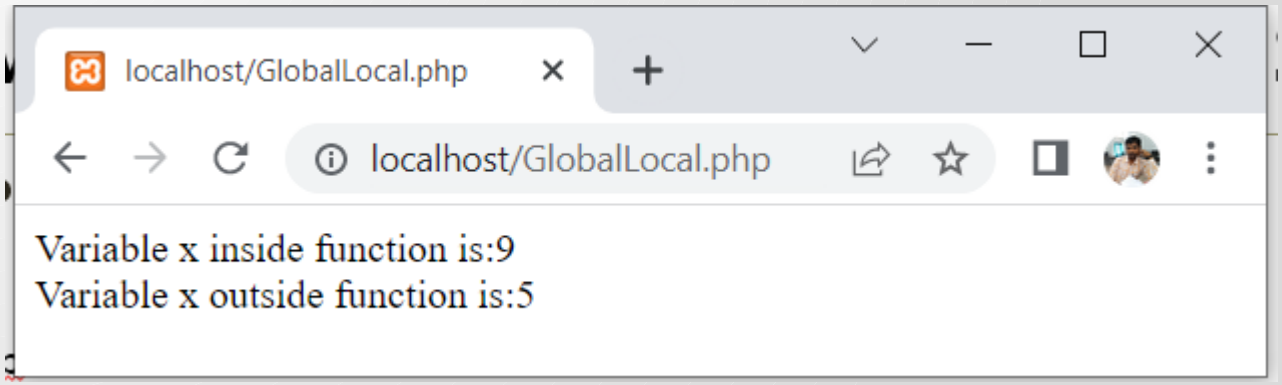
PHP Variable Scope

- In PHP, variables can be declared anywhere in the script.
- The scope of a variable is the part of the script where the variable can be referenced/used.
- PHP has three different variable scopes:
  - local
  - global
  - static

PHP Variables Scope-Global and Local Scope

- **Example**

```
<html>
<body>
    <?php
   $x = 5; // global scope
      function myTest()
      {
      $x=9;          //local scope
      echo "Variable x inside function is:$x";
   }
   myTest();
   echo "<br>Variable x outside function is:$x";
   ?>
</body>
</html>
```

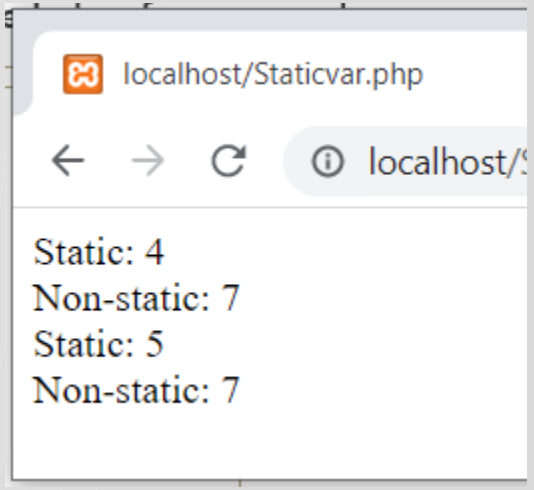


PHP Variables Scope-Global and Local Scope

- We use the static keyword before the variable to define a variable, and this variable is called as **static variable**.
- Static variables exist only in a local function, but it does not free its memory after the program execution leaves the scope. Understand it with the help of an example:

- **Example**

```
<html>
<body>
    <?php
        function static_var()
        {
            static $num1 = 3;       //static variable
            $num2 = 6;          //Non-static variable
$num1++;                //increment in non-static variable
$num2++;                //increment in static variable
            echo "Static: " .$num1 ."</br>";
            echo "Non-static: " .$num2 ."</br>";
        }
        static_var(); //first function call
        static_var(); //second function call
   ?>
</body>
</html>
```



PHP Constant

- PHP constants are name or identifier that can't be changed during the execution of the script except for magic constants , which are not really constants.
- PHP constants can be defined by 2 ways:
  - Using define() function
  - Using const keyword

1. PHP constant: define()

- Use the define() function to create a constant. It defines constant at run time. Let's see the syntax of define() function in PHP.

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

  1. **name:** It specifies the constant name.
  2. **value:** It specifies the constant value.
  3. **case-insensitive:** Specifies whether a constant is case-insensitive. Default value is false. It means it is case sensitive by default.
- Let's see the example to define PHP constant using define().

  constant1.php

```
<?php
define("MESSAGE","Hello JavaTpoint PHP");
echo MESSAGE;
?>
```

2. PHP constant: const keyword

- PHP introduced a keyword const to create a constant. The const keyword defines constants at compile time. It is a language construct, not a function.
- The constant defined using const keyword are case-sensitive.
- *File: constant4.php*

```
<?php
const MESSAGE="Hello const by KP";
echo MESSAGE;
?>
```

PHP Primitives or PHP Data Types

- PHP data types are used to hold different types of data or values.
- PHP supports 8 primitive data types that can be categorized further in 3 types.

PHP Primitives	Scalar (predefined)	boolean integer float string
	Compound Types (user-defined)	Arrays Objects
	Special Types	Resource NULL

PHP Primitives: Scalar Types

PHP Boolean

- Booleans are the simplest data type works like switch. It holds only two values: TRUE (1) or FALSE (0). It is often used with conditional statements. If the condition is correct, it returns TRUE otherwise FALSE.
- **Example:**

```
<?php
   if (TRUE)
      echo "This condition is TRUE.";
   if (FALSE)
      echo "This condition is FALSE.";
?>
```

PHP Integer

- Integer means numeric data with a negative or positive sign. It holds only whole numbers, i.e., numbers without fractional part or decimal points.
- Rules for integer:
  1. An integer can be either positive or negative.
  2. An integer must not contain decimal point.
  3. Integer can be decimal (base 10), octal (base 8), or hexadecimal (base 16).
  4. The range of an integer must be lie between -2,147,483,648 and 2,147,483,647 i.e., -2^31 to 2^31.

```
<?php
   $dec1 = 34;
   $oct1 = 0243;
   $hexa1 = 0x45;
   echo "Decimal number: " .$dec1. "</br>";
   echo "Octal number: " .$oct1. "</br>";
   echo "HexaDecimal number: " .$hexa1. "</
br>";
?>
```

PHP Primitives: Scalar Types

PHP Float

- A floating-point number is a number with a decimal point. Unlike integer, it can hold numbers with a fractional or decimal point, including a negative or positive sign.
- **Example:**

```
<?php
    $n1 = 19.34;
    $n2 = 54.472;
    $sum = $n1 + $n2;
    echo "Addition of floating numbers:
" .$sum;
?>
```

PHP String

- A string is a non-numeric data type. It holds letters or any alphabets, numbers, and even special characters.
- String values must be enclosed either within single quotes or in double quotes. But both are treated differently.
- Example:

```
<?php
    $company = "Army Institute of Technology";
    echo "Hello $company";
    echo "</br>";
    echo 'Hello $company';
?>
```

PHP Primitives: Compound & Special Types

PHP Array

- An array is a compound data type. It can store multiple values of same data type in a single variable.
- **Example:**

```
<?php
  $bikes = array ("Royal Enfield", "Yamaha", "KTM
");
   var_dump($bikes);
//the var_dump() function returns the datatype a
nd values
   echo "</br>";
   echo "Array Element1: $bikes[0] </br>";
   echo "Array Element2: $bikes[1] </br>";
   echo "Array Element3: $bikes[2] </br>";
?>
```

PHP object

- Objects are the instances of user-defined classes that can store both values and functions. They must be explicitly declared.
- Example:

```
<?php
    class bike {
        function model() {
          $model_name = "Royal Enfield";
          echo "Bike Model: " .
$model_name;
          }
    }
    $obj = new bike();
    $obj -> model();
?>
```

PHP Resource

- Resources are not the exact data type in PHP.
- Basically, these are used to store some function calls or references to external PHP resources.
- For example - a database call. It is an external resource.

PHP NULL

- Null is a special data type that has only one value: NULL means no value.
- There is a convention of writing it in capital letters as it is case sensitive.
- **Example:**

```
<?php
   $nl = NULL;
   echo $nl;   //it will not give any output
?>
```

Type Conversion

Implicit Conversion

- The implicit type conversion is called coercion.
- In implicit type conversion the context of expression determines the data type that is expected or required.
- Coercion happen between integer and double, boolean and other scalar type. Also between numeric and string.
- When double value to integer the fractional part is eliminated and value is not rounded.

Explicit Conversion

- There are 3 ways by which the explicit conversion takes place.

Syntax	Example
(datatype)$variablename	(int)$marks
Conversion_function($variable_name)	Intval($marks)
settype($variablename, "datatype")	Settype($marks, "integer")

Operations & Expressions

- PHP Operator is a symbol i.e used to perform operations on operands.
- In simple words, operators are used to perform operations on variables or values. For example:

```
$num=10+20;        //+ is the operator and 10,20 are operands
```

PHP Operators		
	Arithmetic Operators	+, -, *, /, %, **(Exponentiation)
	Assignment Operators	=, +=, -=, *=, /=, %=
	Bitwise Operators	&, |, ^, ~, <<, >>
	Comparison Operators	==, ===(Identical), !=, !==(Not Identical), <>, <, >, <=, >=, <=>(spaceship)
	Incrementing/Decrementing Operators	++, --
	Logical Operators	and, or, xor, !, &&, ||
	String Operators	.(Concatenation), .= (Concatenation and assignment)
	Array Operators	+, ==, !=, ===, !==, <>
	Type Operators	The type operator **instanceof** is used to determine whether an object, its parent and its derived class are the same type or not
	Execution Operators	execution operator **backticks (``)**
	Error Control Operators	one error control operator, i.e., **at (@) symbol**

Operations & Expressions

Type Operators

- The type operator **instanceof** is used to determine whether an object, its parent and its derived class are the same type or not.
- Basically, this operator determines which certain class the object belongs to.
- It is used in object-oriented programming..
- **Example:**

```
<?php
   //class declaration
   class ABC  {}
   class PQR   {}
   //creating an object of type ABC
   $charu = new ABC();
    //testing the type of object
   if( $charu instanceof ABC)  {
      echo "Charu belongs to ABC class.";
   }  else  {
      echo "Charu belongs to PQR class.";
   }
   echo "</br>";
   var_dump($charu instanceof ABC);  //return true.
   var_dump($charu instanceof PQR); //return false.

?>
```

Execution Operators

- PHP has an execution operator backticks (``).
- PHP executes the content of backticks as a shell command. Execution operator and shell_exec() give the same result.
- Example: echo `dir`;
- Execute the shell command and return the result.
- Here, it will show the directories available in current folder.

Error Control Operators

- PHP has one error control operator, i.e., at (@) symbol.
- Whenever it is used with an expression, any error message will be ignored that might be generated by that expression.
- Example: @file ('non_existent_file')
- Intentional file error

PHP Operators Precedence

Operators	Additional Information	Associativity
clone new	clone and new	non-associative
[	array()	left
**	arithmetic	right
++ -- ~ (int) (float) (string) (array) (object) (bool) @	increment/decrement and types	right
instanceof	types	non-associative
!	logical (negation)	right
* / %	arithmetic	left
+ - .	arithmetic and string concatenation	left
<< >>	bitwise (shift)	left
< <= > >=	comparison	non-associative
== != === !== <>	comparison	non-associative
&	bitwise AND	left
^	bitwise XOR	left
|	bitwise OR	left
&&	logical AND	left
||	logical OR	left
?:	ternary	left
= += -= *= **= /= .= %= &= |= ^= <<= >>= =>	assignment	right
and	logical	left
xor	logical	left

PHP Conditions & Loops

Conditional Statements

- if
- if-else
- nested if

Loop Statements

- while
- do while
- for
- foreach

Jump Statements

- break
- continue

PHP If Statement

- If statement Syntax:

```
if(condition){
        //code to be executed
}
```

- Let's see the example

```
<?php
  $num=12;
  if($num<100){
  echo "$num is less than 100";
  }
?>
```

- If else Statement Syntax:

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

- Let's see the example

```
<?php
  $num=12;
  if($num%2==0){
  echo "$num is even number";
  }else{
  echo "$num is odd number";
  }
?>
```

- Nested If Statement Syntax:

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

- Let's see the example

```
<?php
   $age = 23;
   $nationality = "Indian";
   //applying conditions on nationality and age
   if ($nationality == "Indian")
   {
      if ($age >= 18) {
         echo "Eligible to give vote";
      }
      else {
         echo "Not eligible to give vote";
      }
   }
?>
```

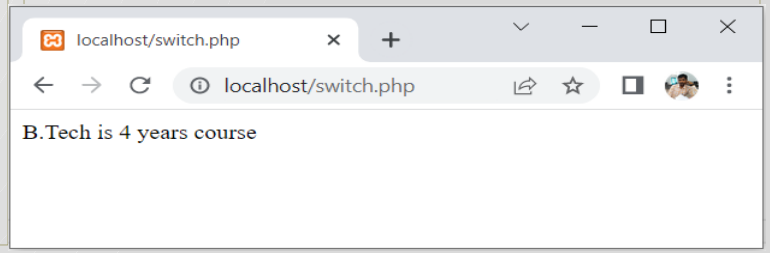
PHP Switch

- PHP switch statement is used to execute one statement from multiple conditions. It works like PHP if-else-if statement.
- Syntax:

```
switch(expression){
   case value1:
                  //code to be executed
                  break;
    case value2:
                 //code to be executed
           break;
    ......
    default:
 code to be executed if all cases are not matched;
}
```

- Important points to be noticed about switch case:
  1. The **default** is an optional statement. Even it is not important, that default must always be the last statement.
  2. There can be only one **default** in a switch statement. More than one default may lead to a **Fatal** error.
  3. Each case can have a **break** statement, which is used to terminate the sequence of statement.
  4. The **break** statement is optional to use in switch. If break is not used, all the statements will execute after finding matched case value.
  5. PHP allows you to use number, character, string, as well as functions in switch expression.
  6. Nesting of switch statements is allowed, but it makes the program more complex and less readable.

```
<?php
   $ch = "B.Tech";
   switch ($ch)
   {
      case "BCA":
         echo "BCA is 3 years course";
         break;
      case "Bsc":
         echo "Bsc is 3 years course";
         break;
      case "B.Tech":
         echo "B.Tech is 4 years course
";
         break;
      case "B.Arch":
         echo "B.Arch is 5 years course
";
         break;
      default:
         echo "Wrong Choice";
         break;
```



PHP Loop Statement

- while loop Statement Syntax:

```
while(condition){     Alternative
while(condition):
        //code to be executed
}                    Alternative  endwhile;
```

- Let's see the example

```
<?php
  $n=1;
  while($n<=10):
  echo "$n<br/>";
  $n++;
  endwhile
?>
```

- do-while loop Statement Syntax:

```
do{
        //code to be executed
}while(condition);
```

- Let's see the example

```
<?php
  $x = 1;
    do {
      echo "1 is not greater than 10.";
      echo "</br>";
      $x++;
    } while ($x > 10);
    echo $x;
?>
```

- for Loop Statement Syntax:

```
for(initialization; condition;
increment/decrement){
        //code to be executed
}
```

- Let's see the example

```
<?php
  for ($i = 1, $j = 0; $i <= 9; $j += $i, print $i,
  $i++)
?>
```

- foreach Loop Statement Syntax:

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

- Let's see the example

```
<?php
  //declare array
    $season = array ("Summer", "Winter",
  "Autumn", "Rainy");
    //access array elements using foreach
  loop
    foreach ($season as $element) {
      echo "$element";
      echo "</br>";
    }
?>
```

PHP Jumping Statements

PHP Break

- PHP break statement breaks the execution of the current for, while, do-while, switch, and for-each loop. If you use break inside inner loop, it breaks the execution of inner loop only.
- The break keyword immediately ends the execution of the loop or switch structure. It breaks the current flow of the program at the specified condition and program control resumes at the next statements outside the loop.
- **Syntax:**

```
jump statement;
break;
```

- **Example:**

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

PHP continue statement

- Used to continue the loop. It continues the current flow of the program and skips the remaining code at the specified condition.
- Used within looping and switch control structure when you immediately jump to next iteration.
- Syntax:

```
jump-statement;
 continue;
```

- Example:

```
<?php
    for ($i =1; $i<=3; $i++) {    //outer loop
      for ($j=1; $j<=3; $j++) {//inner loop
         if (!($i == $j) ) {
            continue;      //skip when i != j
         }
         echo $i.$j;
         echo "</br>";
      }
   }
?>
```

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.

PHP Function

PHP User-defined Functions

- We can declare and call user-defined functions easily. Let's see the syntax to declare user-defined functions.
- **Syntax:**

```
function functionname(){
//code to be executed
}
```

- **Example:**

```
<?php
    function sayHello(){
        echo "Hello PHP Function";
    }
    sayHello();//calling function
?>
<?php    //function returning value
    function cube($n){
        return $n*$n*$n;
    }
    echo "Cube of 3 is: ".cube(3);
?>
```

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.
- Example:

```
<?php
  function sayHello($name,$age){
      echo "Hello $name, you are $age years old<br/>";
  }
  sayHello("Sonoo",27);
  sayHello("Vimal",29);
  sayHello("John",23);
?>
```

PHP Function

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;
?>
```

Default Argument Value

- We can specify a default argument value in function.
- While calling PHP function if you don't specify any argument, it will take the default argument.
- Example:

```
<?php
   function sayHello($age, $name="Hule") {
            echo "Hello $name, you are $age years old<br/>";
   }
   sayHello(27);
   sayHello("Vimal",29);
   sayHello(23);
?>
```

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.
- In PHP, the array() function is used to create an array:
- **Advantage of PHP arrays**
  1. **Less Code**: We don't need to define multiple variables.
  2. **Easy to traverse**: By the help of single loop, we can traverse all the elements of an array.
  3. **Sorting**: We can sort the elements of array.
- **PHP Array Types**
  1. Indexed Array
  2. Associative Array
  3. Multidimensional Array

PHP Arrays Continued....

PHP Indexed Array

- PHP index is represented by number which starts from 0. We can store number, string and object in the PHP array.
- All PHP array elements are assigned to an index number by default.
- There are two ways to define indexed array:
- 1st way:

```
$season=array("summer","winter","spring","autumn");
```

- 2nd way:

```
$season[0]="summer";
$season[1]="winter";
$season[2]="spring";
$season[3]="autumn";
```

- **Example:**

```
<?php

$season=array("summer","winter","spring", "autumn");
```

PHP Associative Array

- PHP allows you to associate name/label with each array elements in PHP using => symbol.
- Such way, you can easily remember the element because each element is represented by label than an incremented number.
- There are 2 ways to define associative array:
- 1st way:

```
$salary=array("Sonoo"=>"550000","Vimal"=>"250000","Ratan"=>"200000");
```

- 2nd way:

```
$salary["Sonoo"]="550000";
$salary["Vimal"]="250000";
$salary["Ratan"]="200000"; .
```

- Example:

```
<?php
  $salary=array("Sonoo"=>"550000", "Vimal"=>"250000","Ratan"=>"200000");
  foreach($salary as $k => $v) {
```

PHP Arrays Continued....

PHP Multidimensional Array

- PHP multidimensional array is also known as array of arrays.
- It allows you to store tabular data in an array. PHP multidimensional array can be represented in the form of matrix which is represented by row * column.
- Definition:

```
$emp = array (
  array(1,"sonoo",400000),
  array(2,"john",500000),  );
```

- **Example:**

```
<?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/>";
 }
 ?>
```

PHP Array Functions

No	Function	Purpose	Syntax	Example
1	array()	creates and returns an array. It allows you to create indexed, associative and multidimensional arrays	array array ([mixed $...])	<?php $season=array("summer","winter","spring","autumn"); echo "Season are: $season[0], $season[1], $season[2] and $season[3]"; ?>
2	array_change_key_case()	PHP array_change_key_case() function changes the case of all key of an array. Note: It changes case of key only.	array array_change_key_case (array $array [, int $case = CASE_LOWER])	<?php $salary=array("Sonoo"=>"550000","Vimal"=>"250000","Ratan"=>"200000"); print_r(array_change_key_case($salary,CASE_LOWER)); ?>
3	array_chunk()	splits array into chunks. By using array_chunk() method, you can divide array into many parts.	array array_chunk (array $array , int $size [, bool $preserve_keys = false])	<?php $salary=array("Sonoo"=>"550000","Vimal"=>"250000","Ratan"=>"200000"); print_r(array_chunk($salary,2)); ?>
4	count()	counts all elements in	int count (mixed	<?php

PHP Array Functions

No	Function	Purpose	Syntax	Example
5	sort()	sorts all the elements in an array.	bool sort (array &$array [, int $sort_flags = SORT_REGULAR])	<?php $season=array("Su","Wi","Sp","Au"); sort($season); foreach($season as $s) echo "$s "; ?>
6	array_reverse()	returns an array containing elements in reversed order.	array array_reverse (array $array [, bool $preserve_keys = false])	<?php $season=array("Su", "Wi", "Sp", "Au"); $reversesason=array_reverse($season); foreach($reverseseason as $s) echo "$s "; ?>
7	array_search()	searches the specified value in an array. It returns key if search is successful.	mixed array_search (mixed $needle , array $haystack [, bool $strict = false])	<?php $season=array("Su","Wi","Sp","Au"); $key=array_search("Sp",$season); echo $key; ?>

PHP Array Functions

No	Function	Purpose	Syntax	Example
9	print_r()	Print all elements of array in standard format.	print_r($array_name)	<?php $a=array('a'=>'apple','b'=>'banana') print_r($a); ?>
10	extract()	Convert array into variable	int extract ($input_array, $extract_rule, $prefix)	<?php $arr=array("a"=>"Pune", "b"=>"Mumbai"); extract($arr); echo "\$a; \$b"; ?>
11	implode()	Convert array into the string.	String implode(separator, $input_array)	<?php $season=array("Su", "Wi", "Sp", "Au"); $text=implode(", ",$season); echo "$text "; ?>
12	explode()	Split the string into array	explode(delimiter, string_name, limit)	<?php $str="Web Technology"; $arr=explode(",",$str); print_r($arr); ?>
13	array_flip()	Used to exchange the keys with their associated values in array. i.e. get keys from array values & values from array become keys.	array_flip(array_name)	<?php $arr=array("Red"=>10, "Blue"=>20, "Black"=>30); $result=array_flip($arr); ?>
14	compact	Converts group of	compact($v1,$v2,	<?php

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 4 ways to specify a string literal in PHP.
  1. single quoted
  2. double quoted
  3. heredoc syntax
  4. newdoc syntax (since PHP 5.3).

Single Quoted String

- 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 (\\).
- All the other instances with backslash such as \r or \n, will be output same as they specified instead of having any special meaning.
- **Example:**

```
<?php
    $str1='Hello text
            multiple line
            text within single quoted
string';
    $str2='Using double "quote"
directly    inside single quoted
string';
    $str3='Using escape sequences \
n in        single quoted string';
    echo "$str1 <br/> $str2 <br/>
```

Double Quoted String

- 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:

```
<?php
  $str1="Hello text
            multiple line
            text within double quoted
string";
  $str2="Using double \"quote\" with
            backslash inside double
quoted string";
  $str3="Using escape sequences \n in
double    quoted string";
  echo "$str1 <br/> $str2 <br/> $str3";
?>
```

Heredoc String

- Heredoc syntax (<<<) is the third way to delimit strings.
- Identifier is provided after this heredoc <<< operator, and immediately a new line is started to write any text.
- To close the quotation, the string follows itself and then again that same identifier is provided. That closing identifier must begin from the new line without any whitespace or tab.
- **Naming Rules:** The identifier should follow the naming rule that it must contain only alphanumeric characters and underscores, and must start with an underscore or a non-digit character.
- **Example:**

```
<?php
    $str = <<<Demo
            It is a valid example
            Demo;    //Valid code as
```

Newdoc String

- Newdoc is similar to the heredoc, but in newdoc parsing is not done.
- It is also identified with three less than symbols <<< followed by an identifier.
- But here identifier is enclosed in single-quote, e.g. <<<'EXP'.
- Newdoc follows the same rule as heredocs.
- The difference between newdoc and heredoc is that - Newdoc is a single-quoted string whereas heredoc is a double-quoted string.
- Example:

```
<?php
  $str = <<<'DEMO'
      Welcome to AIT, Pune.
          Learn with newdoc example.
  DEMO;
  echo $str;
  echo '</br>';
?>
```

PHP String Functions

No	Function	Purpose
1	addcslashes()	It is used to return a string with backslashes.
2	str_replace()	It replaces all occurrences of the search string with the replacement string.
3	chop()	It removes whitespace or other characters from the right end of a string
4	chr()	It is used to return a character from a specified ASCII value.
5	chunk_split()	It is used to split a string into a series of smaller parts.
6	count_chars()	It is used to return information about characters used in a string.
7	crc32()	It is used to calculate a 32-bit CRC for a string.
8	crypt()	It is used to create hashing string One-way.
9	Join()	It is the Alias of implode() function.
10	md5()	It is used to calculate the MD5 hash of a string.
11	printf()	It is used to show output as a formatted string.
12	stripos()	It is used to return the position of the first occurrence of a string

PHP Form Handling

PHP Get Form[$_GET]

- Get request is the default form request.
- The data passed through get request is visible on the URL browser so it is not secured.
- You can send limited amount of data through get request.
- **Example:**

File: form1.html

```
<form action="welcome.php“
method="get">
Name: <input type="text"
name="name"/>
<input type="submit" value="visit"/>
</form>
```

File: welcome.php

```
<?php
    $name=$_GET["name"];//receiving

     name field value in $name
variable
```

PHP Post Form[$_POST]

- Post request is widely used to submit form that have large amount of data such as file upload, image upload, login form, registration form etc.
- The data passed through post request is not visible on the URL browser so it is secured. You can send large amount of data through post request.
- **Example:**

File: form1.html

```
<form action="login.php" method="post">
<table>
<tr><td>Name:</td><td><input type="text" name
="name"/></td></tr>
<tr><td>Password:</td><td> <input
type="password" name="password"/></td></tr>
<tr><td colspan="2"><input type="submit"
value="login"/>  </td></tr>
</table>
</form>
```

File: login.php

```
<?php
$name=$_POST["name"];//receiving name field
value in $name variable
$password=$_POST["password"];//receiving
password field value in $password variable
```

Difference between get and post method

- $_GET is an array of variables passed to the current script via the URL parameters.
- $_POST is an array of variables passed to the current script via the HTTP POST method.
- **When to use GET?**
  - Information sent from a form with the GET method is visible to everyone. GET also has limits on the amount of information to send. The limitation is about 2000 characters.
- **When to use POST?**
  - Information sent from a form with the POST method is invisible to othersand has no limitson the amount of information to send.
  - However, because the variables are not displayed in the URL, it is not possible to bookmark the page.
- **Problem Statements**

1. Create a form containing information Sr.No, bookname, publishers, quantity, price. Read data from the form and display it using PHP script.
2. Write a PHP program to accept a positive integer ‘N’ through a HTML form and display the sum of all the numbers from 1 to N.

Pattern Matching

- The pattern matching technique which is used in PHP is of 2 types:
  - Based on POSIX regular expression & Based on Perl regular expression
- The preg_match() function is a built-in function of PHP that performs a regular expression match. This function searches the string for pattern, and returns true if the pattern exists otherwise returns false.
- Generally, the searching starts from the beginning of $subject string parameter. The optional parameter $offset is used to start the search from the specified position.
- Syntax:

bool preg_match (string $pattern, string $subject, array $matches, int $flags, int $offset)

Parameter	Working
Pattern	It is a string type parameter. This parameter holds the pattern to search as a string.
Subject	This parameter holds the input string in which we search for pattern.
Matches	If matches parameter is provided, it will contain the search results. **matches[0]** - It will hold the text, which matched with the complete pattern. **matches[1]** - It will contain the text, which matched with the first captured parenthesized subpattern, and so on.
flags	The flags can have the following flags given below: **PREG_OFFSET_CAPTURE:** If this flag is passed in preg_match(), for every occurring match the appendant string offset will also return. **PREG_UNMATCHED_AS_NULL:** If this flag is passed in preg_match(), unmatched subpattern will be reported as NULL, otherwise they will be reported as empty string.
offset	By default, the search starts from the beginning of the $subject parameter. The

Pattern Matching Examples

```php
<?php
    $str = "hulekuldeep";
    $pattern = "/kuldeep/i";
    if (preg_match($pattern, $str))
        echo"string matched.";
    else
        echo"string doesnot matched.";
?>
```

```php
<?php

  // get host name from URL
  preg_match('@^(?:https://)?([^/]+)@i',
     "https://www.hulekuldeep.weebly.com/academics", $matches);
  $host = $matches[1];

  // get last two segments of host name
  preg_match('/[^.]+\.[^.]+$/', $host, $matches);
  echo "Domain name is: {$matches[0]}\n";
?>
```

Regex (Regular Expression) syntax

RE	Matches
[abc]	Matches a single character - a, b, or c
[^abc]	Matches any single character but a, b, or c
[a-z]	Matches any single character within the range a-z
[a-zA-Z]	Any single character within the range a-z or A-Z
^	Start of line
$	End of line
\A	Start of string
\z	End of string
.	Any single character
\s	Any whitespace character
\S	Any non-whitespace character
\d	Any digit
\D	Any non-digit
\w	Any word character (letter, number, underscore)
\W	Any non-word character
\b	Word boundary character

RE	Matches
(a|b)	a or b
a?	Zero or one of a
a*	Zero or more of a
a+	One or more of a
a{3}	Exactly 3 of a
a{3,}	3 or more of a
a{3,6}	Between 3 and 6 of a
i	Case insensitive check
m	Make dot match newlines
x	Ignore whitespace in regex

Explaining the pattern
"[^[a-zA-Z0-9._-] +@[a-zA-Z0-9-]+\.[a-zA-Z.]{2,5}$/]"

- **""/?/""** It shows start and end of regular expression.
- **"[^[a-zA-Z0-9._-]"** It matches any uppercase or lowercase letters, numbers between 0 to 9, dot, underscore, or dashes.
- **"+@[a-zA-Z0-9-]"** It matches the @ symbol followed by the upper or lowercase letters, numbers between 0 and 9 or dashes.
- **"+\.[a-zA-Z.]{2,5}$/"** The dot is escaped by using backslash and then matches any lower or

More detail about Pattern Matching

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:**
  - PHP fopen() function is used to open file or URL and returns resource. The fopen() function accepts two arguments: $filename and $mode. The $filename represents the file to be opended and $mode represents the file mode for example read-only, read-write, write-only etc.
  - **Syntax:**

resource fopen (string $filename , string $mode)

Mode	Description
r	Opens file in **read-only** mode. It places the file pointer at the beginning of the file.
r+	Opens file in **read-write** mode. It places the file pointer at the beginning of the file.
w	Opens file in **write-only** mode. It places the file pointer to the beginning of the file and truncates the file to zero length. If file is not found, it creates a new file.
w+	Opens file in **read-write** mode. It places the file pointer to the beginning of the file and truncates the file to zero length. If file is not found, it creates a new file.
a	Opens file in **write-only** mode. It places the file pointer to the end of the file. If file is not found, it creates a new file.
a+	Opens file in **read-write** mode. It places the file pointer to the end of the file. If file is not found, it creates a new file.
x	Creates and opens file in **write-only** mode. It places the file pointer at the beginning of the file. If file is found, fopen() function returns FALSE.
x+	It is same as x but it creates and opens file in **read-write** mode.

PHP File Handling- Read File

- There are different functions that allow you to read all file data, read data line by line and read data character by character.
- Those are: fread(), fgets(), fgetc().
- The PHP **fread()** function is used to read data of the file.
- **Syntax:**

string fread (resource $handle , int $length)

$handle represents file pointer that is created by fopen() function.

$length represents length of byte to be read.

- **Example:**

```
<?php
  $filename = "c:\\file1.txt";
  $fp = fopen($filename, "r");
  $contents = fread($fp,
  filesize($filename));//read file
  echo "<pre>$contents</pre>";
  fclose($fp);//close file
?>
```

- The PHP **fgets()** function is used to read single line from the file.
- **Syntax:**

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

- **Example:**

```
<?php
  $fp = fopen("c:\\file1.txt", "r");
  echo fgets($fp);
  fclose($fp);
?>
```

- The PHP **fgetc()** function is used to read single character from the file.
- To get all data using fgetc() function, use **!feof() function** inside the while loop.
- **Syntax:**

string fgetc (resource $handle)

- **Example:**

```
<?php
  $fp = fopen("c:\\file1.txt", "r");
  while(!feof($fp)) {
    echo fgetc($fp);
  }
  fclose($fp);
```

PHP File Handling- Write File

- PHP fwrite() and fputs() functions are used to write data into file.
- To write data into file, you need to use w, r+, w+, x, x+, c or c+ mode.
- The PHP **fwrite()** function is used to write content of the string into file.
- **Syntax**

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

  handle represents file pointer that is created by fopen() function.

  $length represents length of byte to be read.
- **Example:**

```
<?php
  $fp = fopen('data.txt', 'w');
  fwrite($fp, 'welcome ');
  fwrite($fp, 'to php file write');
  fclose($fp);
  echo "File written successfully";
?>
```

- You can append data into file by using a or **a+ or a** mode in fopen() function.
- Let's see a simple example that appends data into data.txt file.
- **Example:**

```
<?php
  $fp = fopen('data.txt', 'a');
  fwrite($fp, ' this is additional text ');
  fwrite($fp, 'appending data');
  fclose($fp);
  echo "File appended successfully";
?>
```

- In PHP, we can delete any file using unlink() function.
- The unlink() function accepts one argument only: file name.
- PHP unlink() generates E_WARNING level error if file is not deleted. It returns TRUE if file is deleted successfully otherwise FALSE.
- **Syntax:**

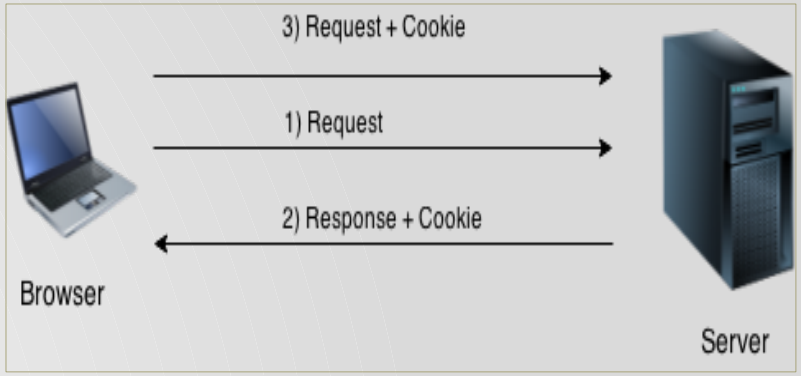
  bool unlink (string $filename [, resource $context])
- **Example:**

```
<?php
  if(unlink('data.txt'))
  echo "File deleted successfully";
  else
  echo "Sorry!";
?>
```

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).
- **How to create a Cookie?:**
- PHP setcookie() function is used to set cookie with HTTP response to create Cookie.
- Once cookie is set/created, you can access it by $_COOKIE superglobal variable.
- **Syntax:**

  setcookie(name, value, expire, path, domain, secure, httponly);
- Only the name parameter is required. All other parameters are optional.
- PHP Delete Cookie:
  - If you set the expiration date in past, cookie will be deleted.
  - Example: setcookie ("CookieName", "", time() - 3600);

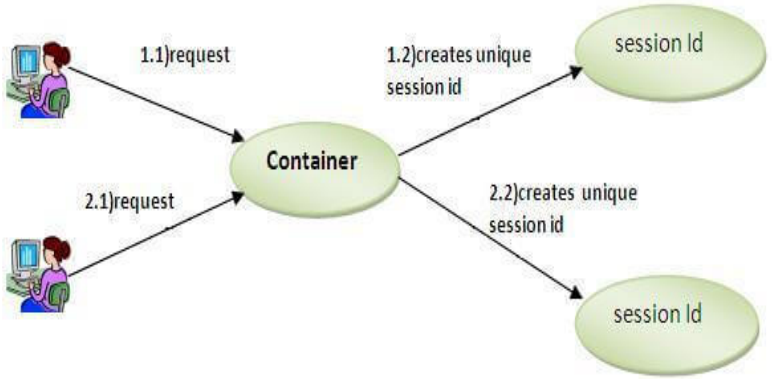


Example:

```
<?php
   setcookie("user","Kuldeep",time()
+(86400*30),"/"); ?>
<html lang="en">
<head>
   <title>Cookie Management</title>
</head>
<body>
<?php
   if(isset($_COOKIE["user"])) {
      $name=$_COOKIE["user"];
      echo "Hello ",$name;
   } else {
      echo "Cookie is not set.";
   }
?>
</body>
</html>
```

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_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();



- **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"];
- **PHP Destroying Session**
  PHP session_destroy() function is used to destroy all session variables completely.

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

Program for Session Management

- **Example:**
- **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>
```

AJAX

- **What is AJAX?**
  - AJAX = **A**synchronous **J**avaScript **A**nd **X**ML.
  - AJAX is not a programming language.
  - AJAX just uses a combination of:
  - A browser built-in XMLHttpRequest object (to request data from a web server)
  - JavaScript and HTML DOM (to display or use the data)

AJAX - Technologies

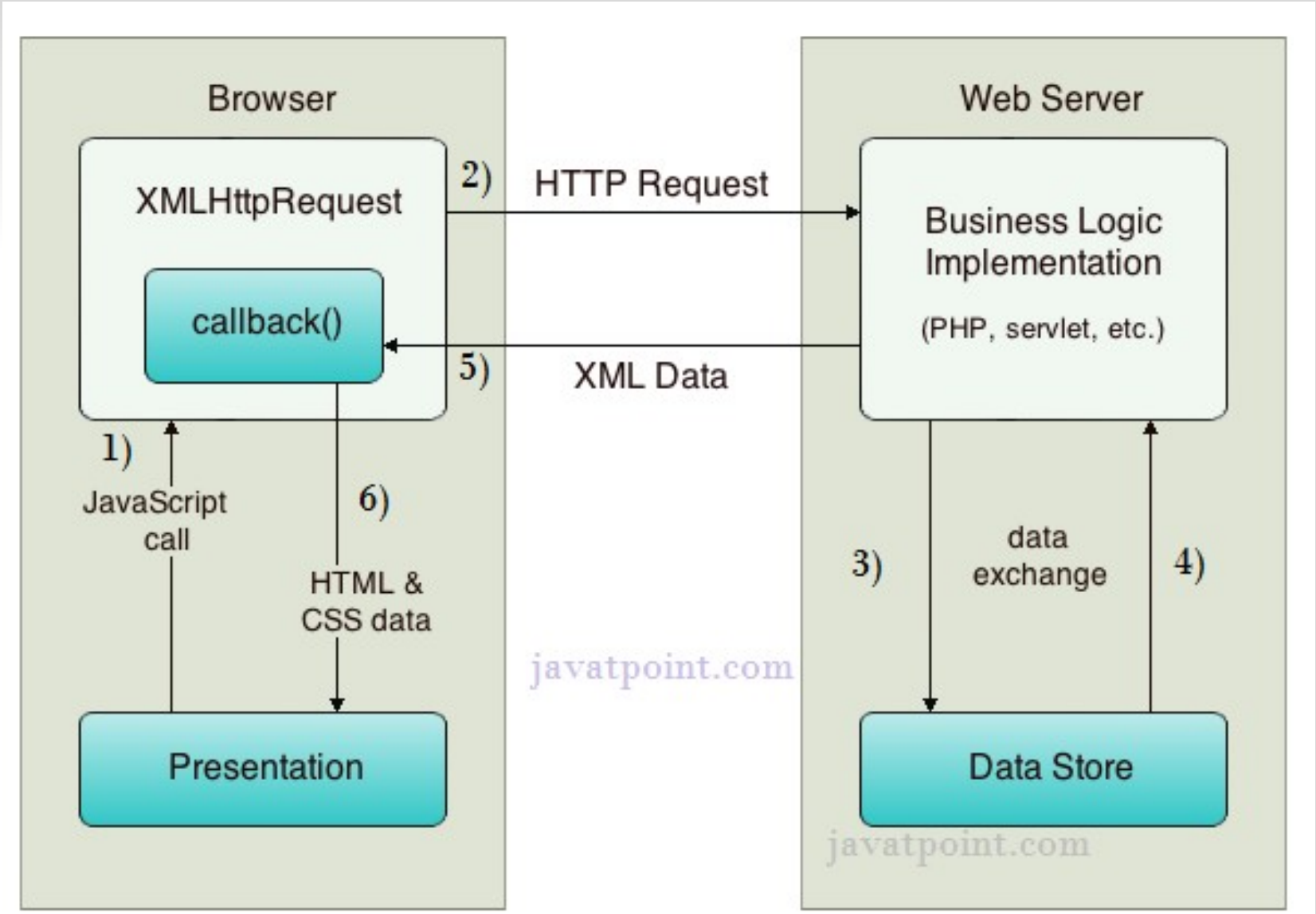
- AJAX cannot work independently. It is used in combination
  with other technologies to create interactive webpages.
- **JavaScript**
  - Loosely typed scripting language.
  - JavaScript function is called when an event occurs in a page.
  - Glue for the whole AJAX operation.
- **DOM**
  - API for accessing and manipulating structured documents.
  - Represents the structure of XML and HTML documents.
- **CSS**
  - Allows for a clear separation of the presentation style

AJAX – Real Time Examples

- Here is a list of some famous web applications that make use of AJAX.

1. **Google Maps**
   - A user can drag an entire map by using the mouse, rather than clicking on a button.
2. **Google Suggest**
   - As you type, Google offers suggestions. Use the arrow keys to navigate the results.
3. **Gmail**
   - Gmail is a webmail built on the idea that emails can be more intuitive, efficient, and useful.
4. **Yahoo Maps (new)**
   - Now it's even easier and more fun to get where

How AJAX Works



AJAX Processing Steps

- **Steps of AJAX Operation**
  - A client event occurs.
  - An XMLHttpRequest object is created.
  - The XMLHttpRequest object is configured.
  - The XMLHttpRequest object makes an asynchronous request to
    the Webserver.
  - The Webserver returns the result containing XML document.
  - The XMLHttpRequest object calls the callback() function and processes the result.
  - The HTML DOM is updated.

AJAX Example – table.html

```
<html>
<head>
<script>
var request;

function sendInfo() {
var v=document.f1.t1.value;
 var url="index.jsp?val="+v;

if(window.XMLHttpRequest)
{  request=new
XMLHttpRequest();
}

request.onreadystatechange=getInf
o;  request.open("GET",url,true);
request.send();

}
```

```
 function getInfo(){
 if (request.readyState==4) {  var
 val=request.responseText;
document.getElementById('amit').innerHTM
L=val;
    }
  }
  </script>
 </head>
 <body>
 <h1>This is an example of ajax</h1>
<form name="f1">
<input type="text" name="t1">
<input type="button" value="ShowTable"
onClick="sendInfo()">
</form>
<span id="amit"> </span>
  </body>
</html>
```

AJAX Example-index.jsp

```
<%
        int
n=Integer.parseInt(request.getParameter("val"));
        for(int i=1;i<=10;i++)
                out.print(i*n+"<br>");
%>
```

AXAX Example output