**INTRODUCTION**

1. **PHP Overview**
2. **PHP Comments**
3. **Variables in PHP**
4. **Variable Scope**

**Data Types & Operators**

1. **PHP DataTypes**
2. **PHP Operators**

**Conditional & Iterative Statements**

1. **PHP Conditional Statements**
2. **PHP Iterative Statements**

**PHP Functions**

1. **Function Basics**
2. **Function Arguments**

**PHP Arrays**

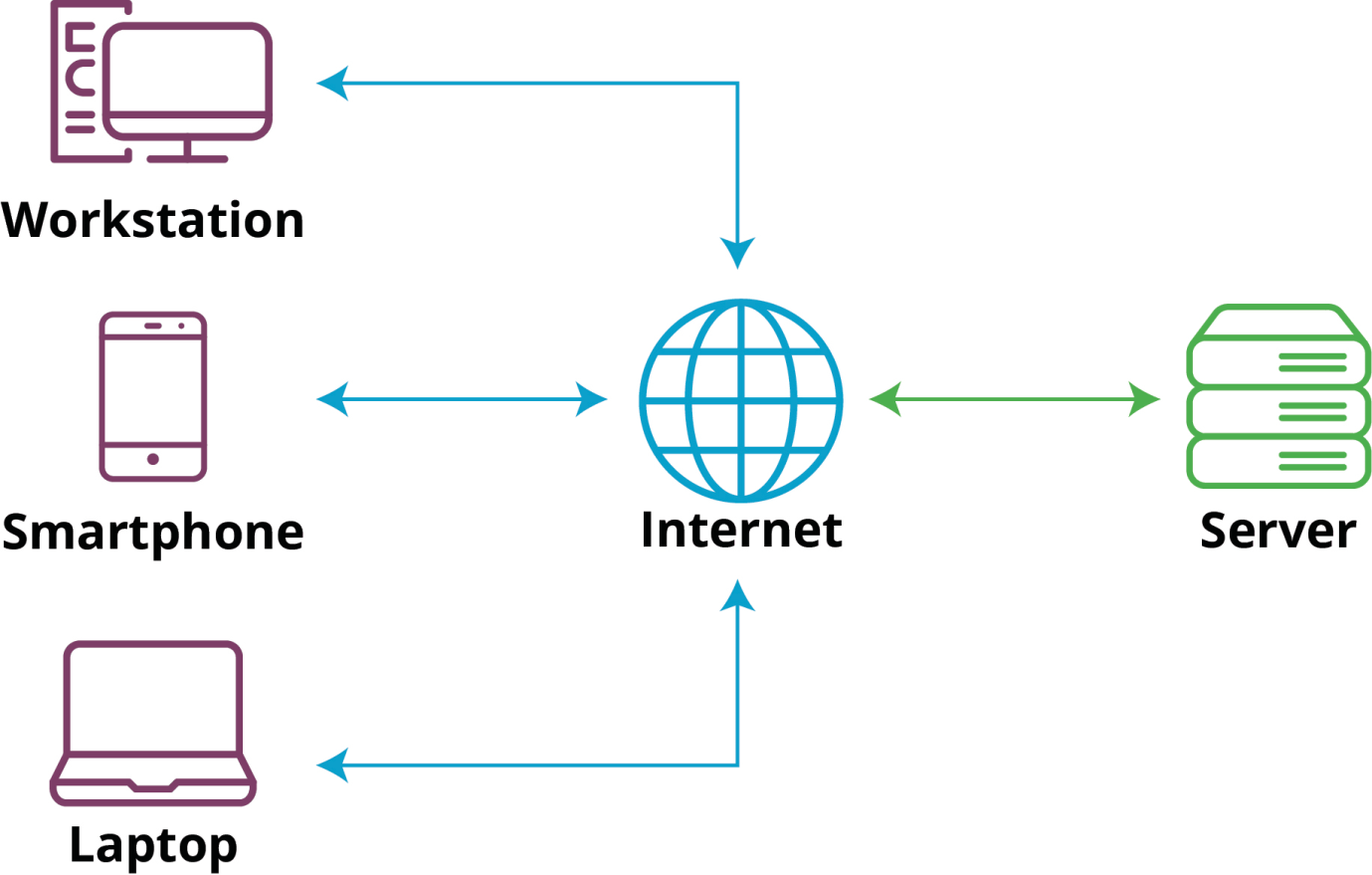
1. **Arrays**

**PHP Overview**

PHP (Hypertext Preprocessor) is a widely-used server-side scripting language designed for web development. Here's a brief overview of PHP:

PHP (Hypertext Preprocessor) was created by Rasmus Lerdorf.

The development began in 1994 Over time, Lerdorf continued to develop and improve PHP, adding more functionality and features. The first version that resembled what we now know as PHP was released in 1995



Client Server Architecture

1. **Purpose:**
   * PHP is mainly used for server-side scripting, enabling the creation of dynamic web pages and web applications.
   * It can be embedded in HTML code and executed on the server, producing HTML output sent to the client's browser.
2. **Key Features:**
   * **Open Source:** PHP is an open-source language, making it widely accessible and continuously evolving with community contributions.
   * **Cross-Platform:** PHP runs on various operating systems, including Windows, Linux, macOS, and can be easily integrated with web servers like Apache and Nginx.
   * **Database Integration:** PHP can interact with a variety of databases, including MySQL, PostgreSQL, SQLite, and others, facilitating dynamic content generation.

**Uses of PHP**

PHP can perform several system functions like opening files, CRUD operations on data stores, general-purpose scripting, etc. Besides system operations, there are also other uses like

1. Handling Forms: PHP can handle form operations. It can gather data, save data to a file and send data through emails.
2. Database Operations: PHP can also create, read, update and delete elements in your database.
3. Encryption: It can perform advanced encryption and encrypt data for you.
4. Dynamic Page Content: It can generate dynamic page content.

**Basic Syntax PHP**

A PHP script can be written anywhere inside the HTML document. A PHP script starts with <?php tag and ends with ?>. We can write our logic inside this tag and it will be executed accordingly.

**<?php**

// PHP code goes here

**?>**

**Displaying output in php**

In php,Output is displayed on the browser using echo as follows:

**<?php**

echo "hello";

**?>**

**Hello World**

A basic PHP Hello World program looks something like this. We will use a built-in PHP contruct or keyword “echo” to output the text “Hello World!” on our webpage.

<!DOCTYPE html>

<html>

<body>

<h1>My first PHP page</h1>

**<?php**

echo "Hello World!";

**?>**

</body>

</html>

In PHP, keywords are reserved words that have special meanings and cannot be used as identifiers (such as variable names, function names, class names, etc.). Here is a list of some PHP keywords:

1. **Basic Language Constructs:**
   * **echo**
   * **print**
   * **unset**
   * **return**
   * **include**
   * **include\_once**
   * **require**
   * **require\_once**
   * **global**
   * **if**
   * **else**
   * **elseif** (alternative to **else if**)
   * **switch**
   * **case**
   * **default**
   * **while**
   * **do**
   * **for**
   * **foreach**
   * **break**
   * **continue**
   * **declare**
   * **function**
   * **class**
   * **interface**
   * **trait**
   * **new**
   * **instanceof**
   * **extends**
   * **implements**
   * **namespace**
   * **use**
   * **const**
   * **final**
   * **abstract**
   * **static**

**PHP Comments**

A comment is a part of the coding file that the programmer does not want to execute, rather the programmer uses it to either explain a block of code or to avoid the execution of a specific part of code while testing.

**PHP supports several ways of commenting:**

**Single Line Comments**

**<?php**

// This is a single-line comment

# This is also a single-line comment

**?>**

**Multiple-Line Comments**

$x = 5 /\* + 15 \*/ + 5;

echo $x;

The  + 15 part will be ignored in the calculation:

**<?php**

/\*

This is a

multiple line

Comment.

\*/

**?>**

**Variables in PHP**

**Variables are containers that can store information which can be manipulated or referenced later by the programmer within the code.**

In PHP, we don’t need to declare the variable type explicitly. The type of variable is determined by the value it stores. There are some important things to know about variables in PHP.

* All variables should be denoted with a Dollar Sign ($)
* Variables are assigned with the = operator, with the variable on the left-hand side and the expression to be evaluated on the right.
* Variable names can only contain alpha-numeric characters and underscores (A-z, 0-9, and \_ ).
* Variables must start with a letter or the underscore “\_” character.
* Variables are case sensitive
* Variable names cannot start with a number.

**For Example:**

**<?php**

$txt = "Hello world!"; # Type String

$x = 5; # Type int

$y = 10.5; # Type Float

**?>**

**Note:** When you assign a text value to a variable, put quotes around the value.

**Note:** Unlike other programming languages, PHP has no command for declaring a variable. It is created the moment you first assign a value to it.

$txt = "programming";

echo "I love $txt!";

The following example will output the sum of two variables:

### Example

$x = 5;

$y = 4;

echo $x + $y;

## PHP is a Loosely Typed Language

In the example above, notice that we did not have to tell PHP which data type the variable is.

PHP automatically associates a data type to the variable, depending on its value. Since the data types are not set in a strict sense, you can do things like adding a string to an integer without causing an error.

In PHP 7, type declarations were added. This gives an option to specify the data type expected when declaring a function, and by enabling the strict requirement, it will throw a "Fatal Error" on a type mismatch.

## Variable Types

PHP has no command for declaring a variable, and the data type depends on the value of the variable.

### Example

$x = 5; // $x is an integer

$y = "John"; // $y is a string

echo $x;

echo $y;

PHP supports the following data types:

* String "some text"
* Integer 8
* Float (floating point numbers - also called double) 8.5
* Boolean 0 or 1
* Array [12,23,34]
* Object
* NULL
* Resource

## Get the Type

To get the data type of a variable, use the var\_dump() function.

### Example

The var\_dump() function returns the data type and the value:

$x = 5;

var\_dump($x);

See what var\_dump() returns for other data types:

var\_dump(5);

var\_dump("John");

var\_dump(3.14);

var\_dump(true);

var\_dump([2, 3, 56]);

var\_dump(NULL);

## Assign String to a Variable

Assigning a string to a variable is done with the variable name followed by an equal sign and the string:

### Example

$x = "John";

echo $x;

## Assign Multiple Values

You can assign the same value to multiple variables in one line:

### Example

All three variables get the value "Fruit":

$x = $y = $z = "Fruit";

**Variable Scope**

The scope of the variable is the area within which the variable has been created. Based on this a variable can either have a local scope or a global scope or a static scope in PHP.

**Global Variable:**

A variable which was created in the main body of the code and that can be accessed anywhere in the program is called Global Variable. Global variables can be directly accessed or used in or outside of a function with GLOBAL keyword before variable. However, we can also call them without the global keyword.

**For Example:**

**<?php**

$name = "hello Bhai"; //Global Variable

function global\_var()

{

global $name;

echo "Variable inside the function: ". $name;

echo "</br>";

}

global\_var();

echo "Variable outside the function: ". $name;

**?>**

Output:

Variable inside the function: hello Bhai

Variable outside the function: hello Bhai

**Local Variable:**

A local variable is created within a function and can be only used inside the function. This means that these variables cannot be accessed outside the function, as they have local scope.

**For Example:**

**<?php**

function mytest()

{

$capital = "Delhi";

echo "Capital of India is: " .$capital;

}

mytest(); //Calling the function

//using $capital outside the function will generate an error

echo $capital;

**?>**

Output:

Capital of India is: Delhi Notice: Undefined variable: capital in D:\xampp\htdocs\program\var.php on line 28

**Static Variable:**

PHP has a feature that deletes the variable once it has finished execution and frees the memory. When we need a local variable which can store its value even after the execution, we use the static keyword before it and the variable is called static variable.

These variables only exist in a local function and do not get deleted after the execution has been completed.

**For Example:**

**<?php**

function static\_var()

{

static $num1 = 3; //static variable

$num2 = 6; //Non-static variable

//increment in non-static variable which will increment its value to 7

$num1++;

//increment in static variable which will increment its value to 4 after first execution and 5 after second execution

$num2++;

echo "Static: " .$num1 ."</br>";

echo "Non-static: " .$num2 ."</br>";

}

//first function call

static\_var();

//second function call

static\_var();

**?>**

Output:

Static: 4

Non-static: 7

Static: 5

Non-static: 7