Unit –1 Introduction to PHP

1.1 Introduction to Static and Dynamic Websites

❖ **What is a static website?**

- A static website is made up of WebPages created using HTML, CSS and JavaScript (all examples of web development languages).
- Each page on a static website is stored as a single HTML file, which is delivered directly from the server to the webpage exactly as is.
- This content essentially becomes a part of the design on your page, and won't change unless the original HTML file is edited at a code level.
- Changes to a static website can be done manually, and will only be made page by page, HTML file by HTML file. For example, edits made to the HTML file of a homepage will only be reflected on the homepage.
- This is true even for elements that are identical across the whole site, such as the footer.
- If you're using a website builder, changes to static pages will be made automatically every time you use the website editor.

Advantages of a static website

- Faster page loading speed
- Quick creation
- Potential for enhanced security

Disadvantages of a static website

- Limited scalability
- Less efficient management

❖ **What is a dynamic website?**

- Built using server side language and technology, dynamic websites allow for the content of each page to be delivered and displayed dynamically, or on-the-fly, according to user behavior or from user-generated content.
- With a dynamic website all of your data and content are organized in a database or backend Content Management System (CMS), which connects to your website pages.
- The way this information is arranged and connected to your site's design controls how and when its content is revealed on a page.
- What does all of this mean? Well, dynamic content gives you the ability to customize and personalize the website experience, and what is displayed, for a specific user.
- It also allows you to make changes to many pages at the same time, since modifications made to one dynamic page can be automatically made across thousands.

Advantages of a dynamic website

- Easily updated
- A better user experience

- Greater functionality

Disadvantages of a dynamic website

- It takes more resources to create
- Performance issues

1.2 Introduction to PHP and it's History

PHP started out as a small open source project that evolved as more and more people found out how useful it was. Rasmus Lerdorf unleashed the first version of PHP way back in 1994.

- PHP is a recursive acronym for "PHP: Hypertext Preprocessor".
- PHP is a server side scripting language that is embedded in HTML. It is used to manage dynamic content, databases, session tracking, even build entire e-commerce sites.
- It is integrated with a number of popular databases, including MySQL, PostgreSQL, Oracle, Sybase, Informix, and Microsoft SQL Server.
- PHP is pleasingly zippy in its execution, especially when compiled as an Apache module on the Unix side. The MySQL server, once started, executes even very complex queries with huge result sets in record-setting time.
- PHP supports a large number of major protocols such as POP3, IMAP, and LDAP. PHP4 added support for Java and distributed object architectures (COM and CORBA), making n-tier development a possibility for the first time.
- PHP is forgiving: PHP language tries to be as forgiving as possible.
- PHP Syntax is C-Like.

❖ Common uses of PHP

- PHP performs system functions, i.e. from files on a system it can create, open, read, write, and close them.
- PHP can handle forms, i.e. gather data from files, save data to a file, through email you can send data, return data to the user.
- You add, delete, modify elements within your database through PHP.
- Access cookies variables and set cookies.
- Using PHP, you can restrict users to access some pages of your website.
- It can encrypt data.

❖ Characteristics of PHP

Five important characteristics make PHP's practical nature possible −

- Simplicity
- Efficiency
- Security
- Flexibility
- Familiarity

❖ **"Hello World" Script in PHP**

To get a feel for PHP, first start with simple PHP scripts. Since "Hello, World!" is an essential example, first we will create a friendly little "Hello, World!" script.
As mentioned earlier, PHP is embedded in HTML. That means that in amongst your normal HTML (or XHTML if you're cutting-edge) you'll have PHP statements like this −

```
<html>

  <head>
    <title>Hello World</title>
  </head>

  <body>
    <?php echo "Hello, World!";?>
  </body>

</html>
```

1.3 Basic PHP syntax and file structure

❖ **Basic PHP Syntax**

- A PHP script can be placed anywhere in the document.
- A PHP script starts with <?php and ends with ?>:

```
<?php
    // PHP code goes here
?>
```

- The default file extension for PHP files is ".php".
- A PHP file normally contains HTML tags, and some PHP scripting code.

❖ **HTML tags**

- There are HTML tags for PHP code to indicate the start and end of PHP code in an HTML file.
- Any one of the following 4 tags can be used:
  1. <?php php-code-here ?>
  2. <SCRIPT LANGUAGE="php"> php-code-here </SCRIPT>
  3. <? php-code-here ?>
  4. <% php-code-here %>
- The first and second tags are the ones most recommended and most widely used.
- Using a tag which is rarely used ***may*** result in a web-server being unable to detect the start and end of the PHP code.

❖ **Commenting**

- # and // are used to comment out a single line of code, while /* and */ indicate the start and end of a commented block of code.

1.4 Output statements: echo and print

❖ PHP echo and print Statements

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

❖ The PHP echo Statement

- The echo statement can be used with or without parentheses: echo or echo().
- The following example shows how to output text with the echo command
- ```
  <?php
  echo "<h2>PHP is Fun!</h2>";
  echo "Hello world!<br>";
  echo "I'm about to learn PHP!<br>";
  echo "This ", "string ", "was ", "made ", "with multiple parameters.";
  ?>
  ```

❖ The PHP print Statement

- The print statement can be used with or without parentheses: print or print().
- The following example shows how to output text with the print command <?php
  ```
  print "<h2>PHP is Fun!</h2>";
  print "Hello world!<br>";
  print "I'm about to learn PHP!";
  ?>
  ```

1.5 PHP variables and value types

❖ Creating (Declaring) PHP Variables

- In PHP, a variable starts with the $ sign, followed by the name of the variable:

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

❖ PHP Variables

- A variable can have a short name (like x and y) or a more descriptive name (age, carname, total_volume).

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

❖ PHP Data Types

- Variables can store data of different types, and different data types can do different things.
- PHP supports the following data types:
  - String
  - Integer
  - Float (floating point numbers - also called double)
  - Boolean
  - Array
  - Object
  - NULL
  - Resource

1.6 PHP Constants and magic constants

❖ 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).

Syntax

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

Parameters:

- *name*: Specifies the name of the constant
- *value*: Specifies the value of the constant
- *case-insensitive*: Specifies whether the constant name should be case-insensitive. Default is false

Example

```
<?php
   define("GREETING", "Welcome!");
   echo GREETING;
?>
```

❖ **Magic Constants**

- Magic constants are the predefined constants in PHP which get changed on the basis of their use.
- They start with double underscore (__) and ends with double underscore.
- They are similar to other predefined constants but as they change their values with the context, they are called **magic** constants.
- There are **nine** magic constants in PHP. In which eight magic constants start and end with double underscores (__).

1. __LINE__
2. __FILE__
3. __DIR__
4. __FUNCTION__
5. __CLASS__
6. __TRAIT__
7. __METHOD__
8. __NAMESPACE__
9. ClassName::class

1.7 PHP Operators and their precedence:

- PHP Operators can be categorized in following forms:
  - Arithmetic Operators
  - Assignment Operators
  - Bitwise Operators
  - Comparison Operators
  - Incrementing/Decrementing Operators
  - Logical Operators
  - String Operators
  - Array Operators
  - Type Operators
  - Execution Operators
  - Error Control Operators
- We can also categorize operators on behalf of operands. They can be categorized in 3 forms:
  - **Unary Operators:** works on single operands such as ++, -- etc.
  - **Binary Operators:** works on two operands such as binary +, -, *, /

- **Ternary Operators:** works on three operands such as "?:".

❖ Arithmetic operators

- The PHP arithmetic operators are used to perform common arithmetic operations such as addition, subtraction, etc. with numeric values.

Operator	Name	Example	Explanation
+	Addition	$a + $b	Sum of operands
-	Subtraction	$a - $b	Difference of operands
*	Multiplication	$a * $b	Product of operands
/	Division	$a / $b	Quotient of operands
%	Modulus	$a % $b	Remainder of operands
**	Exponentiation	$a ** $b	$a raised to the power $b

❖ Increment-decrement operators

- The increment and decrement operators are used to increase and decrease the value of a variable.

Operator	Name	Example	Explanation
++	Increment	++$a	Increment the value of $a by one, then return $a
		$a++	Return $a, then increment the value of $a by one
--	decrement	--$a	Decrement the value of $a by one, then return $a
		$a--	Return $a, then decrement the value of $a by one

❖ Assignment operators

- The assignment operators are used to assign value to different variables. The basic assignment operator is "=".

Operator	Name	Example	Explanation
=	Assign	$a = $b	The value of right operand is assigned to the left operand.
+=	Add then Assign	$a += $b	Addition same as $a = $a + $b
-=	Subtract then Assign	$a -= $b	Subtraction same as $a = $a - $b
*=	Multiply then Assign	$a *= $b	Multiplication same as $a = $a * $b

/=	Divide then Assign (quotient)	$a /= $b	Find quotient same as $a = $a / $b
%=	Divide then Assign (remainder)	$a %= $b	Find remainder same as $a = $a % $b

❖ **Logical operators**

- The logical operators are used to perform bit-level operations on operands.
- These operators allow the evaluation and manipulation of specific bits within the integer.

Operator	**Name**	**Example**	**Explanation**
and	And	$a and $b	Return TRUE if both $a and $b are true
Or	Or	$a or $b	Return TRUE if either $a or $b is true
xor	Xor	$a xor $b	Return TRUE if either $ or $b is true but not both
!	Not	! $a	Return TRUE if $a is not true
&&	And	$a && $b	Return TRUE if either $a and $b are true
||	Or	$a || $b	Return TRUE if either $a or $b is true

❖ **Bitwise operators**

- The bitwise operators are used to perform bit-level operations on operands.
- These operators allow the evaluation and manipulation of specific bits within the integer.

Operator	**Name**	**Example**	**Explanation**
&	And	$a & $b	Bits that are 1 in both $a and $b are set to 1, otherwise 0.
|	Or(Inclusive or)	$a | $b	Bits that are 1 in either $a or $b are set to 1
^	Xor(Exclusive or)	$a ^ $b	Bits that are 1 in either $a or $b are set to 0.
~	Not	~$a	Bits that are 1 set to 0 and bits that are 0 are set to 1
<<	Shift left	$a << $b	Left shift the bits of operand $a $b steps
>>	Shift right	$a >> $b	Right shift the bits of $a operand by $b number of places

❖ **Comparison operators**

- Comparison operators allow comparing two values, such as number or string.
- Below the list of comparison operators are given**:**

Operator	Name	Example	Explanation
==	Equal	$a == $b	Return TRUE if $a is equal to $b
===	Identical	$a === $b	Return TRUE if $a is equal to $b, and they are of same data type
!==	Not identical	$a !== $b	Return TRUE if $a is not equal to $b, and they are not of same data type
!=	Not equal	$a != $b	Return TRUE if $a is not equal to $b
<>	Not equal	$a <> $b	Return TRUE if $a is not equal to $b
<	Less than	$a < $b	Return TRUE if $a is less than $b
>	Greater than	$a > $b	Return TRUE if $a is greater than $b
<=	Less than or equal to	$a <= $b	Return TRUE if $a is less than or equal $b
>=	Greater than or equal to	$a >= $b	Return TRUE if $a is greater than or equal $b
<=>	Spaceship	$a <=>$b	Return -1 if $a is less than $b Return 0 if $a is equal $b Return 1 if $a is greater than $b

1.8 Decision-making statements: if statement, if-else statement, else-if clause, switch-case statement, the ? operator

In PHP we have the following conditional statements:

- 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.

Syntax:

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

❖ If...Else Statement

- if you want to execute some code if a condition is true and another code if a condition is false, use the if....else statement.

Syntax

```
if (condition)
  code to be executed if condition is true;
else
  code to be executed if condition is false;
```

Example

```
<?php
    $d = date("D");

    if ($d == "Fri")
      echo "Have a nice weekend!";

    else
      echo "Have a nice day!";
  ?>
```

❖ **ElseIf Statement**

- If you want to execute some code if one of the several conditions are true use the elseif statement

Syntax

```
if (condition)
  code to be executed if condition is true;
elseif (condition)
  code to be executed if condition is true;
else
  code to be executed if condition is false;
```

Example

```
<?php
    $d = date("D");

    if ($d == "Fri")
      echo "Have a nice weekend!";

    elseif ($d == "Sun")
      echo "Have a nice Sunday!";

    else
      echo "Have a nice day!";
  ?>
```

❖ Switch Statement

- If you want to select one of many blocks of code to be executed, use the Switch statement.
- The switch statement is used to avoid long blocks of if..elseif..else code.

Syntax

```
switch (expression)
{
  case label1:
    code to be executed if expression = label1;
    break;

  case label2:
    code to be executed if expression = label2;
    break;

default:
    code to be executed   if expression is different  from both label1 and label2;
}
```

Example

```
<?php
      $d = date("D");

      switch ($d)
        {
        case "Mon":
          echo "Today is Monday";
          break;

        case "Tue":
          echo "Today is Tuesday";
          break;

        case "Wed":
          echo "Today is Wednesday";
          break;

        case "Thu":
          echo "Today is Thursday";
          break;

        case "Fri":
```

```
    echo "Today is Friday";
    break;

  case "Sat":
    echo "Today is Saturday";
    break;

  case "Sun":
    echo "Today is Sunday";
    break;

  default:
    echo "Wonder which day is this ?";
 }
?>
```

1.9 Loops: while loop, for loop, foreach loop, nesting loops

❖ For loop statement

- The for statement is used when you know how many times you want to execute a statement or a block of statements.

Syntax

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

Example

```
<?php
    $a = 0;
    $b = 0;

    for( $i = 0; $i<5; $i++ )
  {
    $a += 10;
    $b += 5;
    }

    echo ("At the end of the loop a = $a and b = $b" );
  ?>
```

❖ while loop statement

- The while statement will execute a block of code if and as long as a test expression is true.
- If the test expression is true then the code block will be executed. After the code has executed the test expression will again be evaluated and the loop will continue until the test expression is found to be false.

Syntax

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

Example

```
<?php
    $i = 0;
    $num = 50;

    while( $i < 10) {
      $num--;
      $i++;
    }

    echo ("Loop stopped at i = $i and num = $num" );
  ?>
```

❖ **do...while loop statement**

- The do...while statement will execute a block of code at least once - it then will repeat the loop as long as a condition is true.

Syntax

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

Example

```
<?php
    $i = 0;
    $num = 0;

    do
  {
      $i++;
    }
```

```
    while( $i < 10 );
    echo ("Loop stopped at i = $i" );
  ?>
```

❖ **foreach loop statement**

- The foreach statement is used to loop through arrays. For each pass the value of the current array element is assigned to $value and the array pointer is moved by one and in the next pass next element will be processed.

Syntax

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

Example

```
<?php
    $array = array( 1, 2, 3, 4, 5);

    foreach( $array as $value )
   {
      echo "Value is $value <br />";
    }
 ?>
```

1.10 Break and continue statements

❖ **Break statement**

- The PHP break keyword is used to terminate the execution of a loop prematurely.
- The break statement is situated inside the statement block. It gives you full control and whenever you want to exit from the loop you can come out. After coming out of a loop immediate statement to the loop will be executed.

Example

```
<?php
    $i = 0;

    while( $i < 10)
{
      $i++;
      if( $i == 3 )
            break;
 }
```

```
        echo ("Loop stopped at i = $i" );
?>
```

❖ **Continue statement**

- The PHP continue keyword is used to halt the current iteration of a loop but it does not terminate the loop.
- Just like the break statement the continue statement is situated inside the statement block containing the code that the loop executes, preceded by a conditional test. For the pass encountering continue statement, rest of the loop code is skipped and next pass starts.

Example

```
<?php
    $array = array( 1, 2, 3, 4, 5);

    foreach( $array as $value )
   {
      if( $value == 3 )
          continue;
      echo "Value is $value <br />";
    }
   ?>
```