Introduction to PHP

Prepared by Deepali Sonawane, MIT-WPU, Pune

Evolution of PHP

- PHP began as the Personal Home Page
- PHP developed by Rasmus Lerdorf in 1994
- PHP 2 released 1997 (PHP now stands for Hypertext Preprocessor).
- PHP3 released in 1998
- PHP4 released in 2000
- PHP5.0.0 released in 2004
- PHP5.0.5 released in 2005
- PHP 6 was developed but not released
- PHP 7 released in 2015
- PHP 8 released in 2020. Currently we are using PHP 8.

Features of PHP

- PHP is an acronym for "PHP: Hypertext Preprocessor“
- PHP is a server scripting language (Xampp)
- PHP is a powerful tool for making dynamic and interactive Web pages
- PHP is a widely-used, open source scripting language
- PHP scripts are executed on the server
- Various built-in functions allow for fast development
- Compatible with many popular databases (MySQL)
- Easy to learn : Very similar in syntax to C and C++

Advantages of PHP

- PHP runs on various platforms (Windows, Linux, Unix etc.)
- PHP is compatible with almost all servers used today (Apache, IIS, etc.)
- 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 add, delete, modify data in your database
- PHP can be used to control user-access
- PHP can encrypt data
- PHP can send and receive cookies

PHP File Structure

- PHP files can contain text, HTML, CSS, JavaScript, and PHP code
- PHP code is executed on the server, and the result is returned to the browser as plain HTML
- Structurally similar to C/C++
- Supports procedural and object-oriented paradigm.
- All PHP statements end with a semi-colon(;)
- A PHP script can be placed anywhere in the document.
- PHP files saved with ".php" extension
- A PHP script starts with <?php and ends with ?>

Comments

- A comment in PHP code is a line that is not executed as a part of the program.
- Its only purpose is to be read by someone who is looking at the code.
- It can be used to understand code to others.
- It reminds programmer what he did when created particular code.
- PHP supports several ways of commenting:
  - Single Line Comment : //
  - Single Line Comment : #
  - Multiline Comment: /*……*/

echo and print statement

Echo and print statement are used to **output the data** on the screen

Difference between echo and print statement are:

1. echo has no return value while print has a return value of 1 so it can be used in expressions.
2. echo can take multiple parameters while print can take one parameter.
3. echo is marginally faster than print.

echo statement

- The echo statement can be used with or without parentheses :
- echo or echo()

Examples:

```
<?php
   echo "<h2>PHP is Hypertext Preprocessor!</h2>”;
   echo "I'm about to learn PHP!<br>";
   echo "This ", "string ", "was ", "made ", "with multiple parameters.";
   echo (“Welcome to Internet Progamming Using PHP”);
?>
```

print statement

- The print statement can be used with or without parentheses :
- print or print()

Examples:

```
<?php
   print "<h2>PHP is Hypertext Preprocessor!</h2>“;
   print "I'm about to learn PHP!<br>";
  print(“Welcome to Internet Progamming Using PHP”);
?>
```

Variables

- Variables are used to store the data
- A variable starts with the "$" sign, followed by the name of the variable
- Variables are case sensitive (eg. $name != $NAME !=$Name)
- A Variable can have a short name (eg. $x) or a more descriptive name (eg. $name)
- 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 _)
- Global and Locally-scoped variables used in PHP
  - Global variables can be used anywhere
  - Local variables restricted to a function or class

Data types

- Variables can store data of different types
- PHP supports the following data types:
    1. Integer ($num=10)
    2. Float ($percentage=89.67)
    3. String ($name="MIT");
    4. Character ($a='P')
    5. Boolean ($flag=true)
    6. Array
    7. Object
    8. NULL

Examples of echo statement

```
<?php
   $num=20;
   $name="Hello";
   echo $num; //20
   echo $name; //Hello
   echo $num, $name; //20Hello
   echo "5 X 4 = " , $num; // 5 X 4 = 20
   echo  "5 X 4 =  $num "; // 5 X 4 = 20
   echo  '5 X 4 =  $num'; // 5 X 4 =  $num
?>
```

Constants

- Constants contains fixed value, once they are defined they cannot be changed.
- A constant is an identifier. The value cannot be changed during the script.
- A valid constant name starts with a letter or underscore (no $ sign before the constant name).
- To create a constant, use the define() function.
- Syntax : define(*name*, *value* [,*case-insensitive]*) //Value of case insensitive is false.
- Example : 1) define(“PI", 3.142); echo PI;

  2) define(“PI", “3.142“, true); echo pi;

Operators

- Operator is a symbol of operation.
- Operators are used to perform operations on variables and values.
- Types of Operators:
  1. Arithmetic Operator (+, - , *, /, %, **)
  2. Comparison Operator (<, >, <=, >= ,==, !=, <>)
  3. Logical Operator (&&, | |, !)
  4. Assignment Operator (=, +=, -=, *=, /=, %=)
  5. Increment and Decrement operator(++,--)
  6. Conditional Operator (? :)
  7. String Concatenation Operator (.)

Arithmetic Operators

- Arithmetic Operators are used for arithmetic operations.
- Example:

```
<?php
    $a=10;
    $b=2;
    echo "Addition is ", $a+$b;
   echo “Subtraction is “ , $a-$b;
    echo “Multiplication is “ , $a*$b;
    echo “Division is “ , $a/$b;
   echo “Remainder is “ , $a%$b;
   echo “Exponentiation is “ , $a**$b; //same as $a raised to $b
?>
```

Comparison Operators

- Comparison Operators are used to compare two values and it returns 1(true) or 0(false).
- Example:

```
<?php
   $a=10;
   $b=2;
   echo "Result  is ", $a>$b;
   echo "Result  is  " , $a<>$b;
   ?>
```

Logical Operators

- Logical operators are used to combine conditional statements.
- Example:

```
<?php
   $a=10;
   $b=2;
  $c = 8;
  $d = ($a>$b) && ($a>$c);
   echo “Result  is ", $d;
?>
```

Assignment Operators

- Assignment operator(=) is used to assign value to the variable.
- Shorthand assignment operators are +=, -=, *=, /=, %=, **=
- Example:

```
<?php
   $a=10;
   echo $a;
  $a+=4;  // $a=$a+4;
  echo $a;
?>
```

Increment and Decrement Operators

- Increment operator(++) is used to increase the value of variable by 1.
- Decrement operator(--) is used to decrease the value of variable by 1.
- Example:

```
<?php
  $a=10;
 ++$a; //Prefix Increment
  echo $a;
 $c=($a++)+7; //Post Increment
 echo $a , $c ;
?>
```

Conditional Operators

- ? : is a conditional operator.
- Set the value depending on condition.
- Syntax: (condition) ? True value : False value;
- Example:

```
<?php
   $a = 10;
  $b = 20;
  $max = ($a<$b) ? $b : $a;
  echo $max;
?>
```

String Concatenation Operators

- Dot (.) operator is a string concatenation operator.
- Example:

```
<?php
   $a = “Hello”;
  $b = “World!!!”;
  echo $a . $b;//
  $c = $a . “  “ .$b;
  echo $c;
?>
```

Capturing Form Data

- $_GET and $_POST are used to collect form-data.
- Both GET and POST create an array like array(key1 => value1, key2 => value2, key3 => value3, ...) where keys are the names of the form controls and values are the input data from the user.
- $_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.

Difference between GET and POST

GET	POST
Information sent from a form with the GET method is **visible to everyone**	Information sent from a form with the POST method is **invisible to others**
All variable names and values are displayed in the **URL**	All names/values are embedded within the body of the **HTTP request**
GET has **limits** on the amount of information to send. The limitation is about 2000 characters	POST has **no limits** on the amount of information to send
Results can be **book marked** due to the visibility of the values in the URL	Results **cannot be book marked**
GET may be used for sending **non-sensitive data**	POST used for sending **sensitive data**

Dealing with Multi-value field

HTML Code : lan.html file

```
<form method="get" action="Submitlan.php">
   Select Subject:
 <select name="mySelection[]“ multiple>
 <option value="PHP">PHP Language</option>
 <option value="Java">Java Language</option>
 <option value="CPP">CPP Language</option>
 <option value="C">C Language</option>
 </select>
  <br><input type="Submit">
</form>
```

Dealing with Multi-value field

PHP Code : Submitlan.php file

```
<?php
   foreach ( $_GET["mySelection"] as $v)
   {
                    echo $v ,"<br>";
   }
?>
```

Control Structure

- Decision Making statements / Conditional Statements
  - if statement
  - if-else statement
  - nested if statement
  - else if statement
  - switch statement
- Loop statements/Iterative Statements
  - while loop
  - do while loop
  - for loop
  - foreach loop
- Jump Statement
  - Break statement
  - Continue statement

Decision Making Statement

1) if statement

```
Syntax : if (condition)
              {
                       statements;
              }
```

2) if-else statement

```
Syntax : if(condition)
              {
                       statement1;
              }
              else
              {
                       statement2;
              }
```

Decision Making Statement

3) Nested if statement

```
Syntax: if(condition1)
        {
                if(condition2)
                { statement1; }
                else
                { statement2; }
        }
        else
        {
                statement3;
        }
```

Decision Making Statement

4) else if ladder statement

```
Syntax: if(condition1)
                statement1;
        else if(condition2)
                statement2;
        else if(condition3)
                statement3;
                ………
        else
                statement n;
```

Decision Making Statement

5) switch statement

Syntax: switch(expression)

```
            {
                    case label 1 : statements;
                                   break;
                    case label 2 : statements;
                                   break;
                              ........
                    default : statements;
                                   break;
            }
```

Loop Statements /Iterative statements

1) while loop

Syntax:

```
initialization;
while(condition)
{
        statements;
        update statement;
}
```

Loop Statement /Iterative statements

2) do while loop

Syntax:

```
initialization;
do
{
        statements;
        update statement;
} while(condition);
```

Loop Statement / Iterative statements

3) for loop

Syntax:
```
for(initialization ; condition ; update statement)
      {
            statements;
      }
```

Example : To display numbers from 1 to 10.
```
     for($i=1 ; $i<=10 ; $i++)
     {
            echo $i, "<br>";
     }
```

Loop Statement / Iterative statements

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

Syntax:
```
foreach ($array as $value)
            {
                    statements;
            }
```

Example:
```
<?php
   $courses = array(“OS", “Java", “PHP", “IOT“);

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

Jump Statement

1) break statement : break statement is used to jump out from the loop.

Syntax: break;

Example:

```
<?php
   for($i=1;$i<=10;$i++)
   {
          if($i%2==0)
                    break;
          echo $i,"<br>";
   }
?>
```

Jump Statement

2) continue statement : The continue statement breaks one iteration in the loop, if a specified condition occurs, and continues with the next iteration in the loop.

Syntax: continue;

Example:

```
<?php
   for($i=1;$i<=10;$i++)
   {
          if($i%2==0)
                    continue;
          echo $i,"<br>";
   }
?>
```

Generating File uploaded form

HTML File

```
<html>
<body>

<form action="upload.php" method="post" enctype="multipart/
form-data">
  Select image to upload:
  <input type="file" name="fileToUpload">
  <input type="submit" value="UploadImage" name="submit">
</form>

</body>
</html>
```

Uploading File

PHP File

```php
<?php
$target_dir = "uploads/";
$target_file = $target_dir . basename($_FILES["fileToUpload"]["name"]);
$uploadOk = 1;
$imageFileType = strtolower(pathinfo($target_file,PATHINFO_EXTENSION));

// Check if image file is a actual image or fake image
if(isset($_POST["submit"])) {
  $check = getimagesize($_FILES["fileToUpload"]["tmp_name"]);
  if($check == true) {
    echo "File is an image - " . $check["mime"] . ".";
    $uploadOk = 1;
  } else {
    echo "File is not an image.";
    $uploadOk = 0;
  }
}
```

Uploading File

PHP File

```
// Check if file already exists
if (file_exists($target_file)) {
  echo "Sorry, file already exists.";
  $uploadOk = 0;
}

// Check file size
if ($_FILES["fileToUpload"]["size"] > 500000) {
  echo "Sorry, your file is too large.";
  $uploadOk = 0;
}

// Allow certain file formats
if($imageFileType != "jpg" && $imageFileType != "png" && $imageFileType
!= ”jpeg” && $imageFileType != "gif" )
{
  echo "Sorry, only JPG, JPEG, PNG & GIF files are allowed.";
  $uploadOk = 0;
}
```

Uploading File

PHP File

```
if ($uploadOk == 0)
{
  echo "Sorry, your file was not uploaded.";
}
else {
  if (move_uploaded_file($_FILES["fileToUpload"]["tmp_name"], $target_file))
  {
    echo "The file ". htmlspecialchars(basename( $_FILES["fileToUpload"]["name"])). " 
has been uploaded.";
  }
  else {
    echo "Sorry, there was an error uploading your file.";
  }
}
?>
```

Thank you