**Introduction to PHP**

**Scripting Language:** This is a programming language that don’t require an explicit compilation step.  
Two types of scripting languages:  
 1. **Client side scripting language:**   
 The client is the system on which the *Web browser* is running. *JavaScript* is the main client  
 side scripting language for the Web. Client-side scripts are *interpreted* by the browser. The  
 process with client-side scripting is:   
 a) the user requests a Web page from the server   
 b) the server finds the page and sends it to the user   
 c) the page is displayed on the browser with any *scripts running during or after display*.  
 2. **Server side scripting language:** The server-side environment that runs a scripting language is a *Web server*. A user's request is  
 fulfilled by *running a script* directly on the web server to *generate dynamic HTML pages*. This  
 HTML is then *sent to the client browser*. It is usually used to provide interactive web sites that  
 interface to databases or other data stores on the server.  
 **Ex.** PHP, JSP, Python, ASP, Javascript(Node.js), Perl, Ruby etc.

**PHP: Hypertext Preprocessor**

PHP file:

* PHP files have extension “.php”
* It can contain text, HTML, CSS, JavaScript and PHP code.
* PHP codes are executed on the server and the result is returned to the browser as plain HTML.

PHP functionalities:

* 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

PHP file syntax:

**Keywords** are not case-sensitive.

<!DOCTYPE html><html><head><meta charset=**"UTF-8"**><title>**Sample PHP Page**</title></head>

<body><?php // put your code here #oh, I am also single line comment **echo** "hello php"; /\*this is a  multi-line  comment \*/ ?></body></html>

PHP variables

* PHP variables start with the $ sign, followed by the name of the variable
* No explicit datatype declaration is required, PHP automatically converts the variable to the correct data type depending on its value.
* 3 types of variable scope: local, global, static

<!DOCTYPE html>

<html>

<head>

<meta charset=**"UTF-8"**>

<title>**Sample PHP Page**</title>

</head>

<body>

<?php

$global=3.1416;

**function** myfunc(){

$local="sample text";

**echo** "local variable: **$local**<br/>";

//global variable

**echo** "global variable: ".$GLOBALS["global"]." <br/>";

//static variable

**static** $static=10;

$static++;

**echo** "static variable: **$static**<br/>";

}

myfunc();

myfunc();

myfunc();

?>

</body>

</html>

Different Data Types:

|  |  |
| --- | --- |
| **Data Type Name** | **Sample Example** |
| String | $var=”Hello World”, $var1=’Hello World’ |
| Integer | $x=100 |
| Float | $x=3.1416 |
| Boolean | $t=true, $f=false |
| Array | $arr=array(‘item1’,’item2’,’item3’) |
| Null | $nullvar=null |

PHP Strings

|  |  |
| --- | --- |
| **Functions** | **Sample code** |
| echo() | echo(“hello world”) |
| htmlspecialchars() htmlspecialchars\_decode() | htmlspecialchars(“this is some <i>italic</i> text”) |
| parse\_str() | parse\_str(“name=xyz&id=123”); echo $name.”<br/>”.$id; |
| str\_replace() | str\_replace(“abc”,”def”,”abcghiabcjkl”) |
| str\_split() | str\_split(“abcd”) |
| strcmp() | strcmp(‘abcd’,’abcd’) |
| strlen() | strlen(“length of the string”); |
| strpos() | strops(“abcdabcd”,”cd”) |
| strrev() | strrev(“gnirts”) |
| strtolower()/strtoupper() | strtolower(“Hello World”) |
| substr() | substr(“abcdef”,5) substr(“abcdef”,-1) substr(“abcde”,0,3) |
| trim() | trim(“ abcde ”) trim(“abcde”,”ae”) |
| implode()/explode() | $arr=array(“item1”,”item2”,”item3”); echo implode(“ ”,$arr);  $str=”hello P H P”; print\_r(explode(“ ”,$str)); |

PHP Operators

* Arithmetic: +, -, \*, /, %
* Assignment: =, +=, -=, \*=, /=, %=
* Comparison: ==, **===**, !=, <>, **!==**, >, <, >=, <=
* Increment/Decrement op: ++$var, $var++, --$var, $var—
* Logical: and, &&, or, ||, xor, !
* String op: . (concatenation), .= (concatenation assignment)
* Array op: + (join two arrays), == , === , !=, !==

if … else if … else statement

<!DOCTYPE html>

<html>

<body>

<?php

$t = **date**("H");

**echo** "<p>The hour (of the server) is " . $t;

**echo** ", and will give the following message:</p>";

**if** ($t < "10") {

**echo** "Have a good morning!";

} **elseif** ($t < "20") {

**echo** "Have a good day!";

} **else** {

**echo** "Have a good night!";

}

?>

</body>

</html>

while statement

<!DOCTYPE html>

<html>

<body>

<?php

$x = 1;

**while**($x <= 5) {

**echo** "The number is: **$x** <br>";

$x++;

}

?>

</body>

</html>

for loop

<!DOCTYPE html>

<html>

<body>

<?php

**for** ($x = 0; $x <= 10; $x++) {

**echo** "The number is: **$x** <br>";

}

?>

</body>

</html>

foreach loop (works only on arrays)

<!DOCTYPE html>

<html>

<body>

<?php

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

**foreach** ($colors **as** $value) {

**echo** "**$value** <br>";

}

?>

</body>

</html>

function

<!DOCTYPE html>

<html>

<body>

<?php

**function** sum($x, $y=0) {

$z = $x + $y;

**return** $z;

}

**echo** "5 + 10 = " . sum(5,10) . "<br>";

**echo** "7 + 13 = " . sum(7,13) . "<br>";

**echo** "2 + 0 = " . sum(2);

?>

</body>

</html>

Arrays in PHP

1. Indexed array:  
    <!DOCTYPE html>

<html>

<body>

<?php

$cars = **array**("Volvo", "BMW", "Toyota");

**echo** "I like " . $cars[0] . ", " . $cars[1] . " and " . $cars[2] . ".";

?>

</body>

</html>

1. Associative array (named indexing)

<!DOCTYPE html>

<html>

<body>

<?php

$age = **array**("Peter"=>"35", "Ben"=>"37", "Joe"=>"43");

// $age[‘peter’]=”35”;

**echo** "Peter is " . $age['Peter'] . " years old.";

**foreach**($age **as** $x => $x\_value) {

**echo** "Key=" . $x . ", Value=" . $x\_value;

**echo** "<br>";

}

?>

</body>

</html>

|  |  |
| --- | --- |
| Functions | Sample code |
| count($arr) | Returns the number of elements in an array |
| sort($arr) rsort($arr) | $cars=**array**("Volvo","BMW","Toyota");  **sort**($cars); |
| ksort($assarr) | sorts an associative array in ascending order, according to the key |
| asort($assarr) | sorts an associative array in ascending order, according to the value |
| array\_unshift($arr,value)  array\_shift($arr) | <?php  $a=**array**("a"=>"red","b"=>"green");  **array\_unshift**($a,"blue");  **print\_r**($a);  ?> |
| array\_search(val,$arr)  ///it returns the index value | <?php  $a=**array**("a"=>"red","b"=>"green","c"=>"blue");  **echo** **array\_search**("red",$a);  ?> |
| array\_reverse($arr) | <?php  $a=**array**("a"=>"Volvo","b"=>"BMW","c"=>"Toyota");  **print\_r**(**array\_reverse**($a));  ?> |
| array\_push($arr,val1,val2,…)  array\_pop($arr) | <?php  $a=**array**("red","green");  **array\_push**($a,"blue","yellow");  **print\_r**($a);  ?> |

$\_REQUEST / $\_POST/ $\_GET superglobal variables

<!DOCTYPE html>

<html>

<body>

<form method=**"post"** action=**"**<?php **echo** $\_SERVER['PHP\_SELF'];?>**"**>

**Name:** <input type=**"text"** name=**"fname"**>

<input type=**"submit"**>

</form>

<?php

**if** ($\_SERVER["REQUEST\_METHOD"] == "POST") {

// collect value of input field

$name = $\_POST['fname'];

**if** (**empty**($name)) {

**echo** "Name is empty";

} **else** {

**echo** $name;

}

}

?>

</body>

</html>

$\_SERVER superglobal in PHP (ref: <http://php.net/manual/en/reserved.variables.server.php> )

<!DOCTYPE html>

<html>

<body>

<?php

$indicesServer = **array**('PHP\_SELF',

'SERVER\_ADDR',

'SERVER\_NAME',

'SERVER\_PROTOCOL',

'REQUEST\_METHOD',

'REQUEST\_TIME',

'QUERY\_STRING',

'HTTPS',

'REMOTE\_ADDR',

'REMOTE\_HOST',

'REMOTE\_PORT',

'REMOTE\_USER',

'SERVER\_ADMIN',

'SERVER\_PORT',

'SERVER\_SIGNATURE',

'AUTH\_TYPE',

'PATH\_INFO') ;

**echo** '<table cellpadding="10">' ;

**foreach** ($indicesServer **as** $arg) {

**if** (**isset**($\_SERVER[$arg])) {

**echo** '<tr><td>'.$arg.'</td><td>' . $\_SERVER[$arg] . '</td></tr>' ;

}

**else** {

**echo** '<tr><td>'.$arg.'</td><td>-</td></tr>' ;

}

}

**echo** '</table>' ;

?>

</body>

</html>

PHP MySQL Connection (ref: <http://php.net/manual/en/class.pdo.php> )

|  |  |
| --- | --- |
| **Operation** | **Sample PHP Code** |
| Connecting to mysql database | <?php  $servername = "localhost";  $username = "username";  $password = "password";  **try** {  $conn = **new** PDO("mysql:host=**$servername**;dbname=myDB", $username, $password);  // set the PDO error mode to exception  $conn->setAttribute(PDO::ATTR\_ERRMODE, PDO::ERRMODE\_EXCEPTION);  **echo** "Connected successfully";  }  **catch**(PDOException $e)  {  **echo** "Connection failed: " . $e->getMessage();  }  ?> |
| To disconnect from database | $conn = null |
| To insert data into database table  $conn->exec(“query”) | <?php  $servername = "localhost";  $username = "username";  $password = "password";  $dbname = "myDBPDO";  **try** {  $conn = **new** PDO("mysql:host=**$servername**;dbname=**$dbname**", $username, $password);  // set the PDO error mode to exception  $conn->setAttribute(PDO::ATTR\_ERRMODE, PDO::ERRMODE\_EXCEPTION);  $sql = "INSERT INTO MyGuests (firstname, lastname, email)  VALUES ('John', 'Doe', 'john@example.com')";  // use exec() because no results are returned  $conn->**exec**($sql);  **echo** "New record created successfully";  }  **catch**(PDOException $e)  {  **echo** $sql . "<br>" . $e->getMessage();  }  $conn = **null**;  ?> |
| To delete data from database  $conn->exec(“query”) | <?php  $servername = "localhost";  $username = "username";  $password = "password";  $dbname = "myDBPDO";  **try** {  $conn = **new** PDO("mysql:host=**$servername**;dbname=**$dbname**", $username, $password);  // set the PDO error mode to exception  $conn->setAttribute(PDO::ATTR\_ERRMODE, PDO::ERRMODE\_EXCEPTION);  // sql to delete a record  $sql = "DELETE FROM MyGuests WHERE id=3";  // use exec() because no results are returned  $conn->**exec**($sql);  **echo** "Record deleted successfully";  }  **catch**(PDOException $e)  {  **echo** $sql . "<br>" . $e->getMessage();  }  $conn = **null**;  ?> |
| To update data in the database table  $conn->exec(“query”) Or, $stmt=$conn->prepare(“query”) $stmt->execute(); $stmt->rowCount(); | <?php  $servername = “localhost”;  $username = “username”;  $password = “password”;  $dbname = “myDBPDO”;  **try** {  $conn = **new** PDO(“mysql:host=**$servername**;dbname=**$dbname**”, $username, $password);  // set the PDO error mode to exception  $conn->setAttribute(PDO::ATTR\_ERRMODE, PDO::ERRMODE\_EXCEPTION);  $sql = “UPDATE MyGuests SET lastname=’Doe’ WHERE id=2”;  // Prepare statement  $stmt = $conn->prepare($sql);  // execute the query  $stmt->execute();  // echo a message to say the UPDATE succeeded  **echo** $stmt->rowCount() . “ records UPDATED successfully”;  }  **catch**(PDOException $e)  {  **echo** $sql . “<br>” . $e->getMessage();  }  $conn = **null**;  ?> |
| Basic Select query  $stmt=$conn->prepare(); $stmt->execute(); $res=$stmt->fetchAll(); | <?php  $servername = "localhost";  $username = "root";  $password = "";  **try** {  $conn = **new** PDO("mysql:host=**$servername**;dbname=hrschema", $username, $password);  // set the PDO error mode to exception  $conn->setAttribute(PDO::ATTR\_ERRMODE, PDO::ERRMODE\_EXCEPTION);  **echo** "Connected successfully";      $stmt=$conn->prepare("select \* from employees");  $stmt->execute();    $result=$stmt->fetchAll(PDO::FETCH\_ASSOC);  **foreach**($result **as** $inarr){  **foreach**($inarr **as** $a=>$b){  **echo** $a." => ".$b." ";  }  **echo** "<br/>";  }    }  **catch**(PDOException $e)  {  **echo** "Connection failed: " . $e->getMessage();  }  ?> |
| Sometimes you need to execute the same query multiple times for different values, for that case you don’t know some values within the sql queries.   Here, in the example I have written the sql query once using ? sign for the unknowns, then I am using the query multiple times for different values. | $stmt=$conn->prepare("select employee\_id, first\_name from employees where employee\_id<? and department\_id<?;");  $emp\_id=110;$dept\_id=80;  $stmt->execute(**array**($emp\_id,$dept\_id));  $result=$stmt->fetchAll(PDO::FETCH\_ASSOC);  **foreach**($result **as** $inarr){  **foreach**($inarr **as** $a=>$b){  **echo** $a." => ".$b." ";  }  **echo** "<br/>";  }    **echo** "new query------------------------------------<br/>";    $emp\_id=120;$dept\_id=60;  $stmt->execute(**array**($emp\_id,$dept\_id));  $result=$stmt->fetchAll(PDO::FETCH\_ASSOC);  **foreach**($result **as** $inarr){  **foreach**($inarr **as** $a=>$b){  **echo** $a." => ".$b." ";  }  **echo** "<br/>";  } |

References: <https://www.w3schools.com/php/default.asp>