Functions and Classes

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.

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.

User-defined Functions

- PHP has a large number of built-in functions such as mathematical, string, date, array functions etc. It is also possible to define a function as per specific requirement. Such function is called user defined function.
- A function is a block of statements that can be used repeatedly in a program.
- A function will not execute automatically when a page loads.
- A function will be executed by a call to the function.

A user-defined function declaration starts with the word function

Syntax

```
//define a function
function myfunction($arg1, $arg2, ... $argn)
{
  statement1;
  statement2;
  ..
  ..
  return $val;
}
//call function
$ret=myfunction($arg1, $arg2, ... $argn);
```

Function may be defined with optional but any number of arguments. However, same number of arguments must be provided while calling. Function's body can contain any valid PHP code i.e. conditionals, loops etc. (even other functions or classes may be defined inside a function). After executing statements in the block, program control goes back to the location from which it was invoked irrespective of presence of last statement of function block as return. An expression in front of return statement returns its value to calling environment.

Example

```
<?php
//function definition
function sayHello(){
  echo "Hello World!";
}
//function call
sayHello();
?>
```

This script will produce following result when run from command line −

Output

Hello World!

Creating Classes in php

- Like C++ and Java, PHP also supports object oriented programming
- Classes are the blueprints of objects. One of the big differences between functions and classes is that a class contains both data (variables) and functions that form a package called an: ‘object’.
- Class is a programmer-defined data type, which includes local methods and local variables.
- Class is a collection of objects. Object has properties and behaviour.
- Syntax: We define our own class by starting with the keyword ‘class’ followed by the name you want to give your new class.

```
<?php
  class person {
```

```
    }
?>
```

Example 1:

```
<?php
class Demo
{
    // Constructor
    public function __construct(){
        echo 'The class "' . __CLASS__ . '" was initiated!<br>';
    }

}

// Create a new object
$obj = new Demo;
?>
```

Output:

The class “Demo" was initiated.

Example 2:

```
<?php
class Demo
{
    // Destructor
    public function __destruct(){
        echo 'The class "' . __CLASS__ . '" was destroyed!';
    }

}
```

```
// Create a new object
$obj = new Demo;
?>
```

Output:

The class “Demo" was destroyed.

Advanced Object-Oriented Programming in PHP

- Extending Classes
- Protected Scope
- Method Overloading
- Accessors And Mutators
- The __get Method
- The __set Method
- The __construct Method

Other Magical Methods

- Static Stuff
- Interfaces

1. **Extending Classes :**

If you think about it, many things in this world have a main type and then a sub-type. For example, there are humans (main type) and then there are men (sub-type) and women (sub-type). As human beings, men and women share a lot of attributes. But the main difference is that all women can have babies and all men cannot have babies.

```
<?
class Human {
        public $numHeads = 1;
        public $numArms = 2;
        public $numLegs = 2;
        public $canHaveBabies = false;
  public function brushTeeth()
  {
```

```
    // lots of code here
  }
  public function goToWork()
  {
    // lots of code here
  }
  public function eatFood()
  {
    // lots of code here
  }
}
class Man extends Human {
        // Our "Human" default already has the correct default for
        // $canHaveBabies, but just in case the default ever changes later...
        public $canHaveBabies = false;
}
class Woman extends Human{
        public $canHaveBabies = true;
}
```

2. Protected Scope :

Protected properties are private, but they CAN be accessed by child classes. So if the Human class had a protected property, then any class method inside of the Human class or inside of the Man or Woman classes could access that property. So to recap;

- Public - Can be seen and changed by any code, anywhere
- Private - Can only be seen and changed by the same class

- Protected - Can only be seen and changed by the same class or any of its child classes

3. Method Overloading :

In many languages, overloading means that you have two different class methods with the same name, but different arguments. For example:

```
public function eatFood($whatFood)
{
  echo "Yum! Eating with bare hands is messy!";
}


public function eatFood($whatFood,$useAFork)
{
  if($useAFork)
  {
     echo "Yum! Eating with a fork is much cleaner!";
  }
  else
  {
     echo "Yum! Eating with bare hands is messy!";
  }
}
```

Overloading in PHP means something different, and is a little too complicated for this article. The above code will give you an error about redeclaring the same function. However, the traditional concept of overloading is still extremely useful, and can still be accomplished in PHP, but in a slightly different fashion.

There's a PHP function called func_get_args() that comes in handy for this. Let's say you have a function or class method that looks like this:

```
function myFunction()
{
}

// All of the three below calls are valid:
myFunction();
myFunction(123);
myFunction(123,"abc");
```

So if all three of those calls are valid, then how would myFunction access the values of 123 and/or "abc" ? This is where func_get_args() comes in. The func_get_args() is used INSIDE of a function or class method and gives you an array that has all the arguments that were passed to the function.

So if myFunction() simply looked like this:

```
function myFunction()
print_r(func_get_args());
}
```

Then myFunction() would echo out an empty array (no arguments passed), and myFunction(123) would show an array with 1 element in it containing 123. Likewise, myFunction(123,"abc") would display an array with 2 elements in it containing 123 and "abc".

So by using func_get_args(), you can have one class method handle ALL the different ways that it could be called.

4. Accessors And Mutators

Accessors ("getters") are functions that simply get/return the value of a class property (usually a private property), while Mutators ("setters") are functions that set/change the value of a class property. There's nothing really technically different about these functions - the functions are just called "Mutators" and "Accessors" because of what they do. Here's an example:

```
class Human
{
	private $name;

	public function getName()
	{
		return $this->name;
	}

	public function setName($name)
	{
		$this->name = $name;
	}
}
```

5. The __get Method :

There are some "magic" methods that you can define in your classes to make your class "smarter." These magic methods already exist in every class, but PHP handles them by default. However, you can tell PHP that you want to take over that job instead. All you have to do is create a new class method using any of those special names.

The first magic method is __get, which is a very simple accessor function. Whenever you try to get a property that doesn't exist, the __get function is called

```
class DrSeuss {
  public $doesNotLike = "Green Eggs and Ham";
	public $doesLike = array("Blue Waffles","Red Jelly");

  public function __get($var)
```

```
  {
    if($var == "likes")
    {
        return implode(" and ",$this->doesLike);
    }
    else
    {
        return "... uhhh.... I don't know?";
    }
  }
}

$x = new DrSeuss();
echo "He doesn't like " . $x->doesNotLike; // He doesn't like Green Eggs and Ham

echo "He DOES like " . $x->likes; // He DOES like Blue Waffles and Red Jelly

echo "He isn't SURE about " . $x->unsure; // He isn't SURE about ... uhhh.... I don't know?
```

So really, __get is just a way of either dealing with specially-named variables, or handling property-not-found errors in a graceful way.

6. The __set Method

The __set method is just the opposite of __get. Instead of handling the cases where your code is trying to access a property that doesn't exist, the __set method handles the cases where your code tries to SET a property that doesn't exist. I will leave the example up to your imagination.

7. The __construct Method

The __construct method is called whenever you create a new instance of a class. This is a very frequently-defined magic method, because you can create some really elegant code. Most people use this __construct method to automatically run some extra code as soon as the instance is created, like this:

```
class Baby {


        public $babyName = "";
  public $currentStatus = "";
        private $secondsLeftOfCrying = 0; // If only this were public...


  public function __construct($babyName)
  {
        $this->babyName = $babyName;
        $this->startCrying();
  }


  private function startCrying()
  {
        $this->currentStatus = "crying";
        $this->secondsLeftOfCrying = 85800;
  }


}


$UnnamedBaby = new Baby(); // Our __construct requires a $babyName, so this would throw an error.
```

$FatBaby = new Baby("Junior"); // Junior is born and he immediately starts crying except for 10 minutes each day

Other Magical Methods

1. Static Stuff

Static properties and methods are simple. So far, whenever we've created a class method, we've had to create an instance of that class FIRST, and THEN call the class method using that instance.

However, when you have a static class method, it simply means that you don't have to create an instance of a class before being able to use it. It looks like this:

```
class Timer {

  public $countDown = 60;

  public function resetTimer()
  {
        // blah blah
  }

  public static function getCurrentDateAndTime()
  {
        return date("m/d/Y H:i:s");
  }
}

echo Timer::getCurrentDateAndTime();
```

2. Interfaces

If you hired someone to create a class, but it HAD to have X, Y, and Z methods, then how would you enforce it? You could manually check the code that they give you back, but that could be cumbersome if you did this regularly, and/or had a large number of methods.

Interfaces are poorly named, in my opinion. Interfaces SHOULD be called "Class Skeletons" or "Class Rules" because that's closer to what they are. An interface LOOKS like a class, but without all the code INSIDE your methods. Instead, it looks like a class with a bunch of empty methods:

```
interface HumanInterface
{
        public function brushTeeth();
        public function eatFood();
}
```

When you create a class, you can tell it to "implement" an interface, which basically means that the class should obey the rules set by that interface, and it should have all the functions listed in the interface. So if we had this class:

```
class Human implements HumanInterface
{
  public function brushTeeth()
  {
        echo "Brush brush brush.";
  }
}
```

```
$X = new Human();
```

Working with Database MySQL and PHP

What is MySQL?

MySQL is an open-source relational database management system (RDBMS). It is the most popular database system used with PHP. MySQL is developed, distributed, and supported by Oracle Corporation.

The data in a MySQL database are stored in tables which consists of columns and rows.

MySQL is a database system that runs on a server.

MySQL is ideal for both small and large applications.

MySQL is very fast, reliable, and easy to use database system.It uses standard SQL

MySQL compiles on a number of platforms.

Downloading MySQL Database

MySQL can be downloaded for free from this link.

How to connect PHP with MySQL Database?

PHP 5 and later can work with a MySQL database using:

1. MySQLi extension.
2. PDO (PHP Data Objects).

Difference Between MySQLi and PDO

- PDO works on 12 different database systems, whereas MySQLi works only with MySQL databases.
- Both PDO and MySQLi are object-oriented, but MySQLi also offers a procedural API.

If at some point of development phase, the user or the development team wants to change the database then it is easy to that in PDO than MySQLi as PDO supports 12 different database systems.He would have to only change the connection string and a few queries. With MySQLi,he will need to rewrite the entire code including the queries.

There are three ways of working with MySQl and PHP

- MySQLi (object-oriented)
- MySQLi (procedural)
- PDO

Connecting to MySQL database using PHP

There are 3 ways in which we can connect to MySQl from PHP as listed above and described below:

Using MySQLi object-oriented procedure: We can use the MySQLi object-oriented procedure to establish a connection to MySQL database from a PHP script.

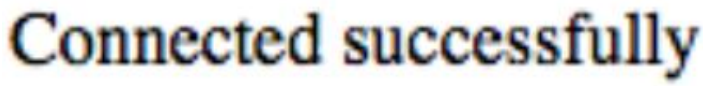
Syntax:

```
<?php
$servername = "localhost";
$username = "username";
$password = "password";

// Creating connection
$conn = new mysqli($servername, $username, $password);
```

```
// Checking connection
if ($conn->connect_error) {
    die("Connection failed: " . $conn->connect_error);
}
echo "Connected successfully";
?>
```

Output:



Explanation: We can create an instance of the mysqli class providing all the necessary details required to establish the connection such as host, username, password etc. If the instance is created successfully then the connection is successful otherwise there is some error in establishing connection.

Using MySQLi procedural procedure : There is also a procedural approach of MySQLi to establish a connection to MySQL database from a PHP script as described below.

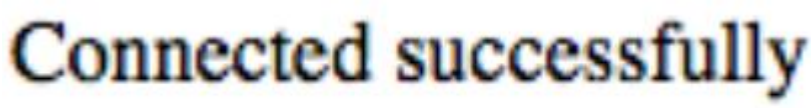
Syntax:

```
<?php
$servername = "localhost";
$username = "username";
$password = "password";
```

```
// Creating connection
$conn = mysqli_connect($servername, $username, $password);

// Checking connection
if (!$conn) {
    die("Connection failed: " . mysqli_connect_error());
}
echo "Connected successfully";
?>
```

Output:



Explanation: In MySQLi procedural approach instead of creating an instance we can use the mysqli_connect() function available in PHP to establish a connection. This function takes the information as arguments such as host, username , password , database name etc. This function returns MySQL link identifier on successful connection or FALSE when failed to establish a connection.

Using PDO procedure: PDO stands for PHP Data Objects. That is, in this method we connect to the database using data objects in PHP as described below:

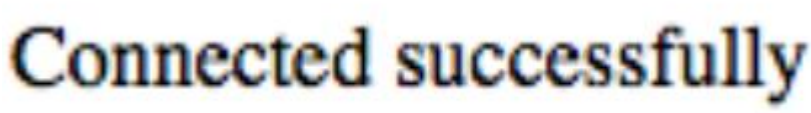
Syntax:

```
<?php
$servername = "localhost";
```

```
$username = "username";
$password = "password";


try {
   $conn = new PDO("mysql:host=$servername;dbname=myDB", $username, $password);
   // setting 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();
   }
?>
```

Output:



Explanation:The exception class in PDO is used to handle any problems that may occur in our database queries. If an exception is thrown within the try{ } block, the script stops executing and flows directly to the first catch(){ } block.

Closing A Connection

When we establish a connection to MySQL database from a PHP script , we should also disconnect or close the connection when our work is finished. Here we have described the syntax of closing the connection to a MySQL database in all 3 methods described above. We have assumed that the reference to the connection is stored in $conn variable.

Using MySQLi object oriented procedure

Syntax

$conn->close();

Using MySQLi procedural procedure

Syntax

mysqli_close($conn);

Using PDO procedure

Syntax

$conn = null;

Creating Database using MySQL and PHP

The CREATE DATABASE statement is used to create a database in MySQL.

```
<?php
$servername = "localhost";
$username = "username";
$password = "password";

// Create connection
$conn = new mysqli($servername, $username, $password);
// Check connection
if ($conn->connect_error) {
  die("Connection failed: " . $conn->connect_error);
```

```
}

// Create database
$sql = "CREATE DATABASE myDB";
if ($conn->query($sql) === TRUE) {
  echo "Database created successfully";
} else {
  echo "Error creating database: " . $conn->error;
}

$conn->close();
?>
```

Read data from Database using PHP

As we know Database is a collection of tables that stores data in it.

To retrieve or fetch data from MySQL database it is simple to do it using MySQL ” Select ” query in PHP .

Here in this blog post we will be going to see how to fetch data and to display it in front end.

MySQL Select Query:

```
SELECT column_name(s)
FROM table_name
```

Copy

PHP:

$query = mysql_query("select * from tablename", $connection);

Copy

For this you must have a database in MySQL . Here, we have a database named “company” which consists of a table named “employee” with 5 fields in it.

Next we have created a PHP page named “updatephp.php” where following steps will be going to perform:

We first establish connection with server .

$connection = mysql_connect("localhost", "root", "");

Copy

Selects database.

$db = mysql_select_db("company", $connection);

Copy

Executes MySQL select query.

$query = mysql_query("select * from employee", $connection);

Copy

Display fetched data

Name: <?php echo $row1['employee_name']; ?>

E-mail: <?php echo $row1['employee_email']; ?>

Contact No: <?php echo $row1['employee_contact']; ?>

Address: <?php echo $row1['employee_address']; ?>

Copy

Closing connection with server.

mysql_close($connection);

Update data using php

Data can be updated into MySQL tables by executing SQL UPDATE statement through PHP function mysql_query.

Below is a simple example to update records into employee table. To update a record in any table it is required to locate that record by using a conditional clause. Below example uses primary key to match a record in employee table.

Example

Try out following example to understand update operation. You need to provide an employee ID to update an employee salary.

```
<html>

  <head>
    <title>Update a Record in MySQL Database</title>
  </head>

  <body>
    <?php
      if(isset($_POST['update'])) {
        $dbhost = 'localhost:3036';
        $dbuser = 'root';
        $dbpass = 'rootpassword';

        $conn = mysql_connect($dbhost, $dbuser, $dbpass);

        if(! $conn ) {
          die('Could not connect: ' . mysql_error());
```

```
   }

   $emp_id = $_POST['emp_id'];
   $emp_salary = $_POST['emp_salary'];

   $sql = "UPDATE employee ". "SET emp_salary = $emp_salary ".
      "WHERE emp_id = $emp_id" ;
   mysql_select_db('test_db');
   $retval = mysql_query( $sql, $conn );

   if(! $retval ) {
      die('Could not update data: ' . mysql_error());
   }
   echo "Updated data successfully\n";

   mysql_close($conn);
}else {
   ?>
      <form method = "post" action = "<?php $_PHP_SELF ?>">
         <table width = "400" border =" 0" cellspacing = "1"
            cellpadding = "2">

            <tr>
               <td width = "100">Employee ID</td>
               <td><input name = "emp_id" type = "text"
                  id = "emp_id"></td>
            </tr>
```

```
            <tr>
              <td width = "100">Employee Salary</td>
              <td><input name = "emp_salary" type = "text"
                id = "emp_salary"></td>
            </tr>

            <tr>
              <td width = "100"> </td>
              <td> </td>
            </tr>

            <tr>
              <td width = "100"> </td>
              <td>
                <input name = "update" type = "submit"
                  id = "update" value = "Update">
              </td>
            </tr>

          </table>
        </form>
      <?php
    }
  ?>

</body>
```

```
</html>
```

Deleting Data from MySQL Database

Data can be deleted from MySQL tables by executing SQL DELETE statement through PHP function mysql_query.

Below is a simple example to delete records into employee table. To delete a record in any table it is required to locate that record by using a conditional clause. Below example uses primary key to match a record in employee table.

Example

Try out following example to understand delete operation. You need to provide an employee ID to delete an employee record from employee table.

```
<html>

  <head>
    <title>Delete a Record from MySQL Database</title>
  </head>

  <body>
    <?php
      if(isset($_POST['delete'])) {
        $dbhost = 'localhost:3036';
        $dbuser = 'root';
        $dbpass = 'rootpassword';
        $conn = mysql_connect($dbhost, $dbuser, $dbpass);

        if(! $conn ) {
```

```
   die('Could not connect: ' . mysql_error());
}

$emp_id = $_POST['emp_id'];

$sql = "DELETE FROM employee WHERE emp_id = $emp_id" ;
mysql_select_db('test_db');
$retval = mysql_query( $sql, $conn );

if(! $retval ) {
   die('Could not delete data: ' . mysql_error());
}
echo "Deleted data successfully\n";
mysql_close($conn);
}else {
?>
   <form method = "post" action = "<?php $_PHP_SELF ?>">
      <table width = "400" border = "0" cellspacing = "1"
         cellpadding = "2">

         <tr>
            <td width = "100">Employee ID</td>
            <td><input name = "emp_id" type = "text"
               id = "emp_id"></td>
         </tr>

         <tr>
```

```
                <td width = "100"> </td>
                <td> </td>
              </tr>

              <tr>
                <td width = "100"> </td>
                <td>
                  <input name = "delete" type = "submit"
                    id = "delete" value = "Delete">
                </td>
              </tr>

            </table>
          </form>
        <?php
      }
    ?>

  </body>
</html>
```