**INDEX PAGE**

NAME: Prina Tamang STUDENT ID: 14

PROGRAM: BCA SEMESTER: 4th

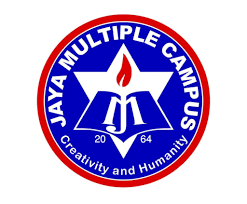
SECTION:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **LAB NO** | **LAB TITLE** | **DATE** | **SIGNATURE** | **REMARKS** |
| 1 | JavaScript |  |  |  |
| 2 | PHP |  |  |  |
| 3 | OOP IN PHP |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

**JAYA MULTIPLE CAMPUS**

**Tribhuvan University**

**Faculties of Humanities and social**



**Lab Report**

**Scripting Language**

**LAB TITLE:**

**OOP IN PHP**

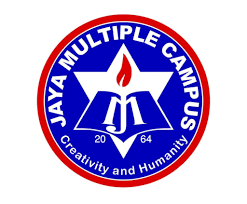
**BACHELORS OF COMPUTER APPLICATION (BCA)**

**TRIBHUVAN UNIVERSITY**

**JAYA MULTIPLE CAMPUS**

**Tribhuvan University**

**Faculties of Humanities and social**



**Lab Report**

**Scripting Language**

**LAB TITLE:**

**PHP**

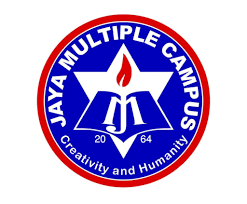
**BACHELORS OF COMPUTER APPLICATION (BCA)**

**TRIBHUVAN UNIVERSITY**

**JAYA MULTIPLE CAMPUS**

**Tribhuvan University**

**Faculties of Humanities and social**



**Lab Report**

**Scripting Language**

**LAB TITLE:**

**JavaScript**

**BACHELORS OF COMPUTER APPLICATION (BCA)**

**TRIBHUVAN UNIVERSITY**

**LAB 1 TITLE: JAVASCRIPT**

**INTRODUCTION:**

JavaScript is a programming language used primarily by web browsers to create a dynamic and interactive experience for the user. It is designed for creating network-centric applications. It is very easy to implement because it is integrated with HTML. It is open and cross-platform.

A JavaScript engine is a computer program that execute JavaScript (JS) code. The JavaScript engine runs in concert with the rendering engine via the Document Object Model.

**OBSERVATION:**

**TASK 1:** Display function in JS.

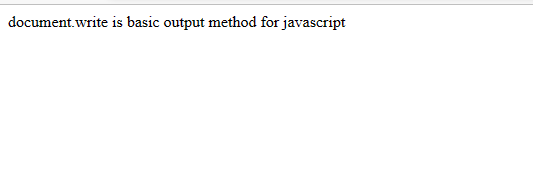
**CODE:**

 <script type="text/javascript">

        document.write("document.write is basic output method for javascript");

    </script>

**Output:**

****

**CONCLUSION:**

In conclusion, above task show the how to display the function in JS.

**JAVASCRIPT FILES**

**INTRODUCTION:**

JavaScript code are stored in separate external file using the .js extension (Ex: Script.js). In the HTML file the <script> tag can also be used to indicate to the location of JavaScript files.

**OBJECTIVE:**

* To show the file location

**OBSERVATION:**

**TASK:** Js file

**Code:**

<html>

<head>

<title>External JavaScript</title>

<script type=”text/javascript” src =”js/3.external\_javascript.js”></script>

</head>

</body>

</html>

**CONCLUSION:**

In the conclusion, the above task show the external location of ja files and link with HTML.

**JAVASCRIPT VARIABLE SCOPE**

**INTRODUCTION:**

The scope of variable is the region of your program in which it is defined JavaScript variables have only two scopes.

1. Global variables
2. Local variables

**OBJECTIVES:**

* To show the work of local variable and global variable

**OBSERVATION:**

**TASK: Global variable and local variable**

**Code:**

**Global variable**

<script>

var usrname= "Ramest";

function modifyUserName() {

userName = "Rajkumar";

};

function showUserName() {

alert(userName);

};

alert(userName);

modifyUserName();

showUserName();

</script>

**Local variable**

<script>

function createUserName() {

var userName = "Ramesh";

}

function showUserName() {

alert(userName);

}

createUserName();

showUserName();

**CONCLUSION:**

In the conclusion, we execute the code and watch the difference in local variable and global variable. Local variable works inside function only and global variable outside of function.

**TASK 2:** Registration form validation using JavaScript

**CODE:**

<!DOCTYPE html>

<html>

<head>

<title>Registration Form</title>

<style>

body {

font-family: Arial, sans-serif;

}

form {

max-width: 400px;

margin: 0 auto;

padding: 20px;

border: 1px solid #ccc;

border-radius: 5px;

}

label {

display: block;

margin-bottom: 5px;

font-weight: bold;

}

input {

width: 100%;

padding: 8px;

margin-bottom: 10px;

border: 1px solid #ccc;

border-radius: 3px;

}

input[type="submit"] {

background-color: #007BFF;

color: #fff;

border: none;

cursor: pointer;

}

input[type="submit"]:hover {

background-color: #0056b3;

}

.error {

color: red;

font-size: 14px;

margin-top: 5px;

}

</style>

<script>

function validateForm() {

var firstName = document.getElementById("firstName").value;

var lastName = document.getElementById("lastName").value;

var email = document.getElementById("email").value;

var password = document.getElementById("password").value;

var confirmPassword = document.getElementById("confirmPassword").value;

var phoneNumber = document.getElementById("phoneNumber").value;

// Simple validation checks

if (firstName === "" || lastName === "" || email === "" || password === "" || confirmPassword === "" || phoneNumber === "") {

alert("All fields are required");

return false;

}

// Check name length

if (firstName.length !== 8) {

alert("First name must be 8 characters long");

return false;

}

// Check password length

if (password.length < 8) {

alert("Password must be at least 8 characters long");

return false;

}

// Check password and confirm password match

if (password !== confirmPassword) {

alert("Passwords do not match");

return false;

}

// Check phone number format

var phonePattern = /^\d{10}$/;

if (!phoneNumber.match(phonePattern)) {

alert("Invalid phone number. Please enter a 10-digit number.");

return false;

}

return true; // Allow form submission

}

</script>

</head>

<body>

<h2>Registration Form</h2>

<form onsubmit="return validateForm()">

<label for="firstName">First Name (8 characters):</label>

<input type="text" id="firstName" name="firstName"><br>

<label for="lastName">Last Name:</label>

<input type="text" id="lastName" name="lastName"><br>

<label for="email">Email:</label>

<input type="text" id="email" name="email"><br>

<label for="password">Password (min 8 characters):</label>

<input type="password" id="password" name="password"><br>

<label for="confirmPassword">Confirm Password:</label>

<input type="password" id="confirmPassword" name="confirmPassword"><br>

<label for="phoneNumber">Phone Number (10 digits):</label>

<input type="text" id="phoneNumber" name="phoneNumber"><br>

<input type="submit" value="Register">

</form>

</body>

</html>

**CONCLUSION:**

In the conclusion, above task we do the form validation using JS. In this task we use the form JS for the validation and show.

**LAB 2 TITLE: PHP**

**INTRODUCTION:**

PHP (Hypertext Preprocessor) is a server-side scripting language designed for web development. It's widely used to create dynamic and interactive web pages by embedding PHP code within HTML code. PHP code is executed on the web server, and the resulting HTML is sent to the client's web browser for rendering. This allows developers to create web applications that can generate content, interact with databases, process forms, and perform a wide range of tasks.

**OBJECTIVES:**

* To Database Connectivity

**OBSERVATIONS:**

TASK 1: Super global

**CODE:**

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Super Global</title>

</head>

<body>

<h4>$GLOBALS</h4>

<?php

$a = 566;

$b = 200;

function sum() {

$GLOBALS['c'] = $GLOBALS['a'] + $GLOBALS['b'];

}

sum();

echo "sum of 566 and 200 is :".$c;

?>

<h4>$\_REQUEST URL:http://localhost/scripting\_language</h4>

<pre>

<?php

print\_r($\_REQUEST);

?>

<h4>$\_SESSION</h4>

<pre>

<?php

session\_start();

$\_SESSION['username']= 'admin';

print\_r($\_SESSION);

?>

</pre>

<h4>$\_COOKIE</h4>

<pre>

<?php

setcookie('name', 'Scripting Language', time() +7\*24\*60\*60);

print\_r($\_COOKIE);

?>

</pre>

<h4>$\_SERVER</h4>

<pre>

<?php

print\_r($\_SERVER)

?>

</pre>

</pre>

</body>

</html>

**Task 3:** CRUD

**Code:**

**Database connection:**

<?php

$servername = "localhost";

$username = "root";

$password = "";

$dbname = "teacher";

// Create a connection

$conn = new mysqli($servername, $username, $password, $dbname);

// Check the connection

if ($conn->connect\_error) {

die("Connection failed: " . $conn->connect\_error);

}

echo "Connected successfully";

// Close the connection when you're done

$conn->close();

?>

**Create:**

<?php

$servername = "localhost";

$username = "root";

$password = " ";

$dbname = "teacher";

$conn = new mysqli($servername, $username, $password, $dbname);

if ($conn->connect\_error) {

die("Connection failed: " . $conn->connect\_error);

}

$username = "john\_doe";

$email = "john@example.com";

$sql = "INSERT INTO Users (username, email) VALUES ('$username', '$email')";

if ($conn->query($sql) === TRUE) {

echo "New record created successfully";

} else {

echo "Error: " . $sql . "<br>" . $conn->error;

}

$conn->close();

?>

**Read:**

<?php

$servername = "localhost";

$username = "root";

$password = " ";

$dbname = "teacher";

$conn = new mysqli($servername, $username, $password, $dbname);

if ($conn->connect\_error) {

die("Connection failed: " . $conn->connect\_error);

}

$sql = "SELECT \* FROM Users";

$result = $conn->query($sql);

if ($result->num\_rows > 0) {

while ($row = $result->fetch\_assoc()) {

echo "Username: " . $row["username"] . " - Email: " . $row["email"] . "<br>";

}

} else {

echo "No records found";

}

$conn->close();

?>

**Update:**

<?php

$servername = "localhost";

$username = "root";

$password = " ";

$dbname = "teacher";

$conn = new mysqli($servername, $username, $password, $dbname);

if ($conn->connect\_error) {

die("Connection failed: " . $conn->connect\_error);

}

$newEmail = "new\_email@example.com";

$userId = 1;

$sql = "UPDATE Users SET email='$newEmail' WHERE id=$userId";

if ($conn->query($sql) === TRUE) {

echo "Record updated successfully";

} else {

echo "Error updating record: " . $conn->error;

}

$conn->close();

?>

**Delete:**

<?php

$servername = "localhost";

$username = "root";

$password = " ";

$dbname = "teacher";

$conn = new mysqli($servername, $username, $password, $dbname);

if ($conn->connect\_error) {

die("Connection failed: " . $conn->connect\_error);

}

$userId = 1;

$sql = "DELETE FROM Users WHERE id=$userId";

if ($conn->query($sql) === TRUE) {

echo "Record deleted successfully";

} else {

echo "Error deleting record: " . $conn->error;

}

$conn->close();

?>

**CONCLUSION:**

In the conclusion, above code show the how to super global variables working process and another task show the how to interact with CRUD operation and show the database connection.

**LAB 3 TITLE: OOP IN PHP**

**INTRODUCTION:**

Object-Oriented Programming (OOP) is a programming paradigm that focuses on organizing code into reusable and self-contained objects. In PHP, you can use OOP principles to create classes and objects, which helps in structuring your code, improving maintainability, and promoting code reusability.

**OBJECTIVES:**

* To building a User Authentication System
* To Modeling a Store Inventory System

**OBSERVATIONS:**

TASK 1: Object and Class

**CODE:**

class Car {

// Properties (attributes)

public $brand;

public $model;

// Constructor

public function \_\_construct($brand, $model) {

$this->brand = $brand;

$this->model = $model;

}

// Method

public function startEngine() {

return "Engine started for $this->brand $this->model";

}

}

// Creating objects

$car1 = new Car("Toyota", "Corolla");

$car2 = new Car("Honda", "Civic");

// Accessing properties and methods

echo $car1->brand; // Output: Toyota

echo $car2->startEngine();

**TASK 2:** Encapsulation

**Code:**

class BankAccount {

private $balance = 0;

public function deposit($amount) {

if ($amount > 0) {

$this->balance += $amount;

}

}

public function getBalance() {

return $this->balance;

}

}

$account = new BankAccount();

$account->deposit(100);

echo $account->getBalance();

**TASK 3: Inheritance**

**CODE:**

class Animal {

public $species;

public function \_\_construct($species) {

$this->species = $species;

}

public function makeSound() {

return "Animal sound";

}

}

class Dog extends Animal {

public function makeSound() {

return "Woof!";

}

}

$dog = new Dog("Canine");

echo $dog->species; // Output: Canine

echo $dog->makeSound();

**TASK 4: Polymorphism**

**CODE:**

class Shape {

public function area() {

return 0;

}

}

class Circle extends Shape {

private $radius;

public function \_\_construct($radius) {

$this->radius = $radius;

}

public function area() {

return 3.14 \* $this->radius \* $this->radius;

}

}

class Rectangle extends Shape {

private $width;

private $height;

public function \_\_construct($width, $height) {

$this->width = $width;

$this->height = $height;

}

public function area() {

return $this->width \* $this->height;

}

}

$shapes = [new Circle(5), new Rectangle(4, 6)];

foreach ($shapes as $shape) {

echo "Area: " . $shape->area() . "<br>";

}

**CONCLUSION:**

In the conclusion, above code show the concept of OOP IN PHP. In this tasks show the OOP in easiest way. Object and class is the first and main part of the OOP.