Web-Programming-Module-4-Important-Topics-PYQs

- iii) Sort the Array and Print the Sorted Key-Value Pairs
- iv) The Entry Identified by the Key "Raj"

- 6. Design an HTML form for entering a number by the user. Write a PHP code to display a message indicating, whether the number is positive or negative, when clicking on the submit button.
- 7. Write a PHP form handling program to perform the user registration of any website with a minimum of 5 different fields and insert the data into a My SQL table after establishing necessary connections with the Database.
- 8. With necessary example give two methods to identify a user across multiple pages in PHP?
    - 1. Using Sessions
        - How it Works
        - Example
    - 2. Using Cookies
        - How it Works
        - Example
- 9. Assume an HTML page exists with fields name, age named as t1 and t2. Develop a PHP code to retrieve the values in textfields in your page. Also print the corresponding name from t1 is minor if age is less than 18 else print corresponding name from t1 is major(eg:-“Anu is minor”)?
    - HTML Page
    - PHP Script to Process Form Data:
    - How it Works:
    - Example Output:
- 10. How dynamic content is handled in PHP?
    - Key Steps in Handling Dynamic Content in PHP
        - 1. Using Variables to Inject Dynamic Data
        - 2. Handling User Input
        - 3. Database Integration
        - 4. Conditional Statements for Content
- 11. Design the HTML page which enters two numbers and embed the PHP code to display the sum, difference, product and quotient of these two numbers, when clicking the ‘CALCULATE’ button.

- 12. What are the uses of cookies in web pages? Describe syntax for setting cookies in PHP. How can you access and delete the cookie using setcookie() function?
- 13. What is a PHP session, explain how to start, modify and destroy session variables with example.
  - How to Start, Modify, and Destroy Session Variables
- 14. Explain CREATE, INSERT and SELECT operations on MySQL table with example.
  - CREATE
  - INSERT
  - SELECT
- 15. Explain form processing in PHP. Design an HTML form for entering a number by the user. Write a PHP code to display a message indicating, whether the number is prime or not, when clicking on the submit button.
  - Form Processing in PHP
  - Steps for Form Processing in PHP
  - Prime Number Checker
    - HTML Form
    - PHP Script to Process the Form

Important Topics

1. Various steps for establishing PHP-MySQL connection with a MySQL

1. **Install MySQL**: Make sure you have a MySQL server running.
2. **Install PHP MySQL Extension**: Ensure the `mysqli` or `PDO` extension is enabled in your PHP installation.
3. **Create a Database**: Create a database in MySQL to store your data.
4. **Write PHP Code to Connect**:

```php
<?php
$servername = "localhost";
$username = "your_username";
$password = "your_password";
$dbname = "your_database";
```

```php
// Create connection
$conn = new mysqli($servername, $username, $password, $dbname);

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

- Replace `your_username`, `your_password`, and `your_database` with actual credentials. The connection is established using `mysqli`.

2. Cookies and Session in PHP with examples.

1. Cookies

Cookies are small pieces of data stored on the user's browser by the server. They are used to remember information like login credentials or preferences across multiple requests.

How to Use Cookies

Setting a Cookie:

```php
<?php
// Create a cookie named "username" that expires in 7 days
setcookie("username", "JohnDoe", time() + (86400 * 7), "/");
?>
```

- **Explanation**:
  - `setcookie(name, value, expiration, path)`
  - `"username"`: Name of the cookie.
  - `"JohnDoe"`: Value of the cookie.
  - `time() + (86400 * 7)`: Expiration time set to 7 days from now.
  - `"/"`: Cookie is available throughout the website.

Retrieving a Cookie:

```php
<?php
if (isset($_COOKIE["username"])) {
    echo "Hello, " . $_COOKIE["username"]; // Outputs: Hello, JohnDoe
} else {
    echo "Cookie not found!";
}
?>
```

- **Explanation**:
  - `$_COOKIE["username"]` : Accesses the cookie value.

Deleting a Cookie:

```php
<?php
setcookie("username", "", time() - 3600, "/"); // Set expiration time to the
past
?>
```

- **Explanation**:
  - Setting the cookie's expiration time to a past value deletes it.

2. Sessions

A **session** stores data on the server, providing a secure way to track user activity across multiple pages. Unlike cookies, session data is not accessible to the user directly.

How to Use Sessions

Starting a Session:

```php
<?php
session_start(); // Starts the session
```

```
$_SESSION['username'] = "JohnDoe"; // Store a session variable
?>
```

- **Explanation**:
  - `session_start()`: Initializes a session or resumes the current one.
  - `$_SESSION`: Associative array used to store session data.

Accessing a Session Variable:

```
<?php
session_start();
echo "Hello, " . $_SESSION['username']; // Outputs: Hello, JohnDoe
?>
```

- **Explanation**:
  - Access session variables using `$_SESSION`.

Modifying a Session Variable:

```
<?php
session_start();
$_SESSION['username'] = "JaneDoe"; // Update the session variable
?>
```

- **Explanation**:
  - Updates the value of an existing session variable.

Destroying a Session:

```
<?php
session_start();
session_unset(); // Removes all session variables
```

```
session_destroy(); // Destroys the session
?>
```

- **Explanation**:
  - `session_unset()` : Clears all session variables.
  - `session_destroy()` : Ends the session completely.

Key Differences Between Cookies and Sessions

Aspect	Cookies	Sessions
Storage	Stored in the user's browser.	Stored on the server.
Security	Less secure as users can modify cookie data.	More secure since data is on the server.
Lifetime	Can persist for days or years.	Ends when the browser is closed (by default).
Size	Limited to 4KB.	Can store larger data.

Previous Year Questions

1. Illustrate with a sample PHP script, how CREATE operations on My SQL table are implemented in PHP.

```php
<?php
$servername = "localhost";
$username = "your_username";
$password = "your_password";
$dbname = "your_database";

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

// Check connection
if ($conn->connect_error) {
```

```
    die("Connection failed: " . $conn->connect_error);
}

// Create table
$sql = "CREATE TABLE users (
    id INT(6) UNSIGNED AUTO_INCREMENT PRIMARY KEY,
    username VARCHAR(30) NOT NULL,
    email VARCHAR(50) NOT NULL
)";

if ($conn->query($sql) === TRUE) {
    echo "Table users created successfully";
} else {
    echo "Error creating table: " . $conn->error;
}

// Close connection
$conn->close();
?>
```

- This script connects to the MySQL database and creates a table named `users` with specified columns.

2. What is cookie? How does PHP support the concept of cookies.

Cookies are small pieces of data that websites store on a user's computer to remember information about the user. They are often used to keep track of user sessions, preferences, or other information.

PHP Support for Cookies:
PHP has built-in functions to create, retrieve, and delete cookies.

Setting a Cookie:

```
<?php
// Set a cookie
setcookie("username", "JohnDoe", time() + (86400 * 30), "/"); // 86400
```

```
seconds = 1 day
?>
```

Explanation:

- The `setcookie()` function creates a cookie named "username" with the value "JohnDoe". It expires in 30 days and is available across the entire site ("/").

Retrieving a Cookie:

```php
<?php
if(isset($_COOKIE["username"])) {
    echo "Welcome back, " . $_COOKIE["username"];
} else {
    echo "Hello, guest!";
}
?>
```

Explanation:

- The code checks if the cookie "username" is set. If it is, it greets the user; otherwise, it welcomes a guest.

Deleting a Cookie:

```php
<?php
// Delete a cookie
setcookie("username", "", time() - 3600, "/"); // Set expiration time in the
past
?>
```

Explanation:

- To delete a cookie, you can set its expiration time to a time in the past.

3. Differentiate between require and include in PHP with proper syntax

Difference Between require and include in PHP

In PHP, both `require` and `include` are used to include and evaluate external PHP files into your script. However, they differ in how they handle errors.

Key Differences

Feature	`require`	`include`
Error Handling	Throws a **fatal error** if the file is missing or not found.	Throws a **warning** if the file is missing, and the script continues.
Usage	Used when the file is essential for the script to run.	Used when the file is optional, and the script can continue without it.
Performance Impact	Slightly slower if errors occur due to termination.	Slightly faster as the script does not stop on error.

Syntax and Examples

1. Using `require`:

```php
<?php
// Required file
require 'config.php';

echo "This line will not be executed if config.php is missing.";
?>
```

Explanation:

- If `config.php` does not exist, the script stops execution and throws a fatal error:

  ```
  Fatal error: require(): Failed opening required 'config.php'
  ```

2. Using `include`:

```php
<?php
// Included file
include 'config.php';
```

```
echo "This line will still execute even if config.php is missing.";
?>
```

Explanation:

- If `config.php` is missing, the script shows a warning but continues execution:

  ```
  Warning: include(config.php): Failed to open stream
  ```

When to Use

1. **Use `require`**:
   - When the file is critical for your application to work (e.g., configuration files or essential functions).
2. **Use `include`**:
   - When the file is not essential, and you want the script to continue even if the file is unavailable (e.g., optional templates or extra features).

4. Write PHP statements to insert data into MySQL table.

```php
<?php
// Database connection details
$servername = "localhost";
$username = "root";
$password = ""; // Set your password
$dbname = "test_db"; // Your database name

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

// Check connection
if ($conn->connect_error) {
    die("Connection failed: " . $conn->connect_error);
}

// SQL query to insert data
```

```php
$name = "John Doe";
$email = "johndoe@example.com";

$sql = "INSERT INTO users (name, email) VALUES ('$name', '$email')";

if ($conn->query($sql) === TRUE) {
    echo "New record created successfully";
} else {
    echo "Error: " . $sql . "<br>" . $conn->error;
}

// Close the connection
$conn->close();
?>
```

5. Write equivalent PHP statements corresponding to the following

- i)Declare an associative array named “marks” to store the key-value pairs (“Ram”, 40), (“Alice”,20),(“Raj”,45),(“Mary”,35).
- ii)Modify the value associated with the key “Ram” to 50.
- iii)Sort the array and print the sorted key value pairs.
- iv)The entry identified by the key “Raj”

i) Declare an Associative Array Named "marks"

```php
<?php
$marks = array(
    "Ram" => 40,
    "Alice" => 20,
    "Raj" => 45,
    "Mary" => 35
);
?>
```

ii) Modify the Value Associated with the Key "Ram" to 50

```php
<?php
$marks["Ram"] = 50; // Update Ram's marks
?>
```

iii) Sort the Array and Print the Sorted Key-Value Pairs

```php
<?php
asort($marks); // Sort the array by values, maintaining key association

foreach ($marks as $name => $score) {
    echo "$name: $score<br>";
}
?>
```

iv) The Entry Identified by the Key "Raj"

```php
<?php
$raj_marks = $marks["Raj"]; // Retrieve marks for Raj
echo "Raj's marks: $raj_marks"; // Output: Raj's marks: 45
?>
```

6. Design an HTML form for entering a number by the user. Write a PHP code to display a message indicating, whether the number is positive or negative, when clicking on the submit button.

HTML Form:

```html
<!DOCTYPE html>
<html>
<head>
    <title>Check Number</title>
</head>
<body>
    <form action="check_number.php" method="POST">
        <label for="number">Enter a number:</label>
        <input type="number" name="number" id="number" required>
```

```
        <input type="submit" value="Submit">
    </form>
</body>
</html>
```

PHP Code (`check_number.php`):

```
<?php
if ($_SERVER["REQUEST_METHOD"] == "POST") {
    $number = $_POST['number']; // Get the number from the form

    if ($number > 0) {
        echo "The number $number is positive.";
    } elseif ($number < 0) {
        echo "The number $number is negative.";
    } else {
        echo "The number is zero.";
    }
}
?>
```

- The HTML form takes a number as input and submits it using the POST method to `check_number.php`.
- The PHP script checks whether the number is positive, negative, or zero and displays the appropriate message.

7. Write a PHP form handling program to perform the user registration of any website with a minimum of 5 different fields and insert the data into a My SQL table after establishing necessary connections with the Database.

HTML Registration Form:

```
<!DOCTYPE html>
<html>
        <head>
            <title>User Registration</title>
```

```
        </head>
<body>
    <form action="register.php" method="POST">
        <label for="username">Username:</label>
        <input type="text" name="username" id="username" required><br><br>

        <label for="email">Email:</label>
        <input type="email" name="email" id="email" required><br><br>

        <label for="password">Password:</label>
        <input type="password" name="password" id="password" required><br>
<br>

        <label for="fullname">Full Name:</label>
        <input type="text" name="fullname" id="fullname" required><br><br>

        <label for="age">Age:</label>
        <input type="number" name="age" id="age" required><br><br>

        <input type="submit" value="Register">
    </form>
</body>
</html>
```

PHP Code to Handle Registration (`register.php`):

```
<?php
// Database connection settings
$servername = "localhost";
$username = "your_db_username";
$password = "your_db_password";
$dbname = "your_database_name";

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

// Check connection
if ($conn->connect_error) {
    die("Connection failed: " . $conn->connect_error);
}
```

```php
// Check if the form is submitted
if ($_SERVER["REQUEST_METHOD"] == "POST") {
    $user = $_POST['username'];
    $email = $_POST['email'];
    $pass = $_POST['password']
    $fullname = $_POST['fullname'];
    $age = $_POST['age'];

    // Prepare and bind
    $stmt = $conn->prepare("INSERT INTO users (username, email, password,
fullname, age) VALUES (?, ?, ?, ?, ?)");
    $stmt->bind_param("ssssi", $user, $email, $pass, $fullname, $age);

    // Execute the statement
    if ($stmt->execute()) {
        echo "Registration successful!";
    } else {
        echo "Error: " . $stmt->error;
    }

    // Close the statement and connection
    $stmt->close();
}

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

- The HTML form collects user registration data: username, email, password, full name, and age. It submits the data to `register.php`.
- The PHP script connects to the MySQL database and checks if the form is submitted.
- It hashes the password for security, prepares an SQL statement to insert the user data into a `users` table, and executes the statement.

8. With necessary example give two methods to identify a user across multiple pages in PHP?

To maintain user identity across multiple pages, PHP offers two common methods: **Sessions** and **Cookies**.

1. Using Sessions

A session allows you to store user-specific information on the server, which can be accessed across multiple pages during the session's lifetime.

How it Works

- Session data is stored on the server, and a unique session ID is sent to the user's browser as a cookie.
- The session ID links the user to their session data on the server.

Example

Page 1: Set Session

```php
<?php
// Start session
session_start();

// Store user information in session variables
$_SESSION['username'] = 'JohnDoe';
$_SESSION['email'] = 'johndoe@example.com';

echo "Session data set. Go to page 2.";
?>
```

Page 2: Retrieve Session

```php
<?php
// Start session
session_start();

// Access session variables
if (isset($_SESSION['username'])) {
    echo "Welcome, " . $_SESSION['username'] . "<br>";
    echo "Your email is " . $_SESSION['email'];
} else {
    echo "No session data found.";
}
?>
```

2. Using Cookies

A cookie is a small file stored on the user's browser. It can hold user-specific data that can be accessed across multiple pages.

How it Works

- Cookies are stored in the user's browser.
- Each page request sends the cookie data back to the server.

Example

Page 1: Set Cookie

```php
<?php
// Set cookies for username and email
setcookie("username", "JohnDoe", time() + (86400 * 7), "/"); // Valid for 7 days
setcookie("email", "johndoe@example.com", time() + (86400 * 7), "/");

echo "Cookies set. Go to page 2.";
?>
```

Page 2: Retrieve Cookie

```php
<?php
// Check if cookies are set
if (isset($_COOKIE['username'])) {
    echo "Welcome, " . $_COOKIE['username'] . "<br>";
    echo "Your email is " . $_COOKIE['email'];
} else {
    echo "No cookies found.";
}
?>
```

9. Assume an HTML page exists with fields name, age named as t1 and t2. Develop a PHP code to retrieve the values in

textfields in your page. Also print the corresponding name from t1 is minor if age is less than 18 else print corresponding name from t1 is major(eg:-“Anu is minor”)?

HTML Page

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Age Checker</title>
</head>
<body>
    <h1>Check if Minor or Major</h1>
    <form action="process.php" method="post">
        <label for="t1">Name:</label>
        <input type="text" id="t1" name="t1" required><br><br>

        <label for="t2">Age:</label>
        <input type="number" id="t2" name="t2" required><br><br>

        <button type="submit">Submit</button>
    </form>
</body>
</html>
```

PHP Script to Process Form Data:

```
<?php
// Check if form data is submitted
if ($_SERVER["REQUEST_METHOD"] == "POST") {
    // Retrieve form values
    $name = $_POST['t1']; // Sanitize input to prevent XSS
    $age = intval($_POST['t2']);            // Convert age to integer

    // Determine if the user is a minor or major
    if ($age < 18) {
        echo "$name is a minor.";
    } else {
        echo "$name is a major.";
```

```
    }
} else {
    echo "No data submitted.";
}
?>
```

How it Works:

1. **HTML Form**:
    - The form contains two input fields:
        - `t1` for the name.
        - `t2` for the age.
    - The `action="process.php"` sends the form data to the `process.php` file when submitted.
2. **PHP Script**:
    - Checks if the request method is `POST` (form submission).
    - Retrieves the name (`t1`) and age (`t2`) using `$_POST`.
    - Converts the `age` to an integer using `intval()` to ensure numeric comparison.
    - Displays whether the user is a minor or major based on their age.

Example Output:

If the user enters:

- Name: **Anu**
- Age: **16**

The output will be:

```
Anu is a minor.
```

If the user enters:

- Name: **Ravi**
- Age: **20**

The output will be:

```
Ravi is a major.
```

10. How dynamic content is handled in PHP?

Dynamic content in PHP is content that is generated on the server based on user input, database queries, or other runtime conditions. PHP enables dynamic content generation by embedding logic within HTML to handle data and display customized output.

Key Steps in Handling Dynamic Content in PHP

1. Using Variables to Inject Dynamic Data

PHP can use variables to display dynamic content within an HTML page.
Example:

```php
<?php
$name = "John";
echo "<h1>Welcome, $name!</h1>";
?>
```

Output:

```
Welcome, John!
```

2. Handling User Input

Dynamic content can be generated based on user input from forms.
Example: HTML Form:

```html
<form method="post" action="process.php">
    <label for="name">Name:</label>
    <input type="text" id="name" name="name">
    <button type="submit">Submit</button>
</form>
```

PHP Script (`process.php`):

```php
<?php
$name = $_POST['name'];
echo "<h1>Hello, $name!</h1>";
?>
```

Output:

If the user inputs "Alice," the page will display:

```
Hello, Alice!
```

3. Database Integration

Dynamic content is often fetched from a database and displayed on a web page.

Example:

```php
<?php
$conn = new mysqli("localhost", "root", "", "mydatabase");
$sql = "SELECT title FROM posts";
$result = $conn->query($sql);

if ($result->num_rows > 0) {
    while ($row = $result->fetch_assoc()) {
        echo "<h2>" . $row['title'] . "</h2>";
    }
} else {
    echo "No posts found.";
}
$conn->close();
?>
```

Output:

Displays a list of post titles stored in the database.

4. Conditional Statements for Content

PHP allows the display of content based on conditions.

Example:

```php
<?php
$hour = date("H");

if ($hour < 12) {
    echo "Good morning!";
} else {
    echo "Good afternoon!";
}
?>
```

Output:

Displays `Good morning!` if the time is before 12 PM, and `Good afternoon!` otherwise.

11. Design the HTML page which enters two numbers and embed the PHP code to display the sum, difference, product and quotient of these two numbers, when clicking the 'CALCULATE' button.

```php
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Calculator</title>
</head>
<body>
    <h1>Simple Calculator</h1>
    <form method="post">
        <label for="num1">Enter first number:</label>
        <input type="number" name="num1" id="num1" required><br>
        <label for="num2">Enter second number:</label>
        <input type="number" name="num2" id="num2" required><br>
        <input type="submit" name="calculate" value="CALCULATE">
    </form>

    <?php
    if ($_SERVER["REQUEST_METHOD"] == "POST") {
```

```
        $num1 = $_POST['num1'];
        $num2 = $_POST['num2'];

        $sum = $num1 + $num2;
        $difference = $num1 - $num2;
        $product = $num1 * $num2;
        $quotient = $num2 != 0 ? $num1 / $num2 : 'undefined';

        echo "<h2>Results:</h2>";
        echo "Sum: $sum<br>";
        echo "Difference: $difference<br>";
        echo "Product: $product<br>";
        echo "Quotient: $quotient<br>";
    }
    ?>
</body>
</html>
```

- This HTML form allows the user to enter two numbers and submits them via the POST method. The PHP code processes the input when the form is submitted, calculates the sum, difference, product, and quotient, and displays the results.

12. What are the uses of cookies in web pages? Describe syntax for setting cookies in PHP. How can you access and delete the cookie using setcookie() function?

Uses of Cookies:

- **Session Management**: Cookies can track user sessions, allowing users to stay logged in as they navigate a site.
- **Personalization**: Cookies help store user preferences, such as language or theme settings.
- **Tracking and Analytics**: Websites can use cookies to track user behavior and gather analytics data.

Syntax for Setting Cookies in PHP:

```
setcookie("cookie_name", "cookie_value", time() + (86400 * 30), "/"); //
86400 = 1 day
```

- **Parameters:**
  - `"cookie_name"`: The name of the cookie.
  - `"cookie_value"`: The value to be stored in the cookie.
  - `time() + (86400 * 30)`: Expiration time (in this case, 30 days from the current time).
  - `"/"`: Path on the server where the cookie is available. `/` means the cookie is available throughout the site.

Accessing and Deleting a Cookie Using `setcookie()`:

- **Accessing a Cookie:**

```
if (isset($_COOKIE["cookie_name"])) {
    echo "Value of cookie_name: " . $_COOKIE["cookie_name"];
} else {
    echo "Cookie is not set.";
}
```

- **Deleting a Cookie:**

```
setcookie("cookie_name", "", time() - 3600, "/"); // Set expiration time
to the past
```

- To delete a cookie, you set its value to an empty string and its expiration time to a time in the past.

13. What is a PHP session, explain how to start, modify and destroy session variables with example.

A **PHP session** is a way to store information (in variables) to be used across multiple pages. Unlike cookies, session data is stored on the server and is more secure.

How to Start, Modify, and Destroy Session Variables

Starting a Session:

```php
<?php
session_start(); // Start the session
$_SESSION['username'] = 'JohnDoe'; // Set session variable
?>
```

Modifying Session Variables:

```php
<?php
session_start(); // Start the session
$_SESSION['username'] = 'JaneDoe'; // Modify session variable
?>
```

Destroying a Session:

```php
<?php
session_start(); // Start the session
session_unset(); // Unset all session variables
session_destroy(); // Destroy the session
?>
```

- `session_start()` : Initializes a new session or resumes an existing one.
- `$_SESSION` : An associative array to store session variables.
- `session_unset()` : Removes all session variables.
- `session_destroy()` : Deletes the session.

14. Explain CREATE, INSERT and SELECT operations on MySQL table with example.

CREATE

The **CREATE** operation is used to create a new table in the database.

Example:

```
CREATE TABLE users (
    id INT(6) UNSIGNED AUTO_INCREMENT PRIMARY KEY,
    username VARCHAR(30) NOT NULL,
    email VARCHAR(50) NOT NULL
);
```

INSERT

The **INSERT** operation adds new records to a table.

Example:

```
INSERT INTO users (username, email) VALUES ('JohnDoe', 'john@example.com');
```

SELECT

The **SELECT** operation retrieves data from a table.

Example:

```
SELECT * FROM users;
```

- The `CREATE` statement defines a new table with columns.
- The `INSERT` statement adds a new row to the table.
- The `SELECT` statement retrieves data from the table.

15. Explain form processing in PHP. Design an HTML form for entering a number by the user. Write a PHP code to display a message indicating, whether the number is prime or not, when clicking on the submit button.

Form Processing in PHP

Form processing in PHP involves collecting user input from an HTML form, validating it, and performing actions based on the input. The input is sent to the PHP script using HTTP methods such as GET or POST.

Steps for Form Processing in PHP

- Create an HTML Form
  - Use an HTML `<form>` element with input fields and a submit button.
- Submit Form Data
  - Use the action attribute to specify the PHP script that will handle the data. The method attribute determines whether the data is sent via GET or POST.
- Process Form Data in PHP
  - Use `$_GET` or `$_POST` superglobals to retrieve the submitted data and perform operations like validation or calculations.

Prime Number Checker

HTML Form

```
<!DOCTYPE html>
<html>
<head>
    <title>Prime Number Checker</title>
</head>
<body>
    <h1>Check if a Number is Prime</h1>
    <form method="post" action="">
        <label for="number">Enter a number:</label>
        <input type="number" id="number" name="number" required>
        <button type="submit">Check</button>
    </form>
</body>
</html>
```

PHP Script to Process the Form

```
<?php
function isPrime($number) {
    if ($number <= 1) {
        return false; // 0 and 1 are not prime numbers
    }
    for ($i = 2; $i <= sqrt($number); $i++) {
        if ($number % $i == 0) {
            return false; // Not a prime number
```

```
        }
    }
    return true; // Prime number
}

// Process the form submission
if ($_SERVER["REQUEST_METHOD"] == "POST") {
    $number = $_POST['number']; // Get the input number

    if (isPrime($number)) {
        echo "<p>$number is a prime number.</p>";
    } else {
        echo "<p>$number is not a prime number.</p>";
    }
}
?>
```

- The function checks if the number is prime by dividing it by all integers up to its square root.