



You:

1) State the usage of following server's variables: a. `$_SERVER['PHP_SELF']` b. `$_SERVER['SERVER_NAME']` c. `$_SERVER['REQUEST_TIME']` d. `$_SERVER['QUERY_STRING']` e. `$_SERVER['HTTP_ACCEPT']` g. `$_SERVER['SCRIPT_NAME']`

-----



ChatGPT:

Certainly! The `$_SERVER` superglobal in PHP is an array containing information about the server environment and request. Here's the usage of the specified variables:

- a. `$_SERVER['PHP_SELF']`: This variable contains the filename of the currently executing script, relative to the document root.
- b. `$_SERVER['SERVER_NAME']`: It holds the server's host name or IP address.
- c. `$_SERVER['REQUEST_TIME']`: Represents the timestamp of the start of the request. It's useful for measuring script execution time.
- d. `$_SERVER['QUERY_STRING']`: Contains the query string, if any, via which the page was accessed.
- e. `$_SERVER['HTTP_ACCEPT']`: This variable holds the value of the Accept HTTP header, indicating the MIME types the client can understand.
- g. `$_SERVER['SCRIPT_NAME']`: It contains the path of the current script. It is useful for self-referencing scripts.

Keep in mind that the information provided by `$_SERVER` variables can be manipulated or modified by the client, so they should be used with caution, especially in security-sensitive contexts.

-----



You:

3) State the difference between Get, Post and Request methods. <https://www.guru99.com/difference-get-post-http.html>

-----



ChatGPT:

Certainly! The GET, POST, and REQUEST methods are different HTTP methods used to send data between the client (browser) and the server. Here's a summary of the differences:

#### 1. GET Method:

- Parameters are included in the URL.
- Limited data can be sent, typically less than 2KB.
- Visible in the browser's address bar.
- Not suitable for sensitive data, as it's visible in the URL.
- Caching is possible, as data is part of the URL.
- Used for requesting data from a specified resource.

#### 2. POST Method:

- Parameters are included in the request body.
- Can handle larger amounts of data.
- Not visible in the browser's address bar.
- Suitable for sensitive data, as it's not exposed in the URL.
- Not cached by default.
- Used for submitting data to be processed to a specified resource.

#### 3. REQUEST Method:

- It is a PHP superglobal variable that can be used to collect data from both GET and POST methods.
- The data from GET and POST requests can be accessed using `$_REQUEST`.
- It is not recommended to use `$_REQUEST` as it may lead to security issues and is generally considered less secure than using `$_GET` or `$_POST` explicitly.

The choice between GET and POST depends on the nature of the data being sent and the security considerations of the application. GET is typically used for non-sensitive data retrieval, while POST is used for data submission and more secure data transfer. Always validate and sanitize user input to prevent security vulnerabilities like SQL injection or cross-site scripting (XSS).

-----



You:

4) Write a PHP script to submit login credentials to the server and get the response of either "Welcome to UTU" or "You are not the legitimate user of the website".

-----



Certainly! Below is a simple example of a PHP script for submitting login credentials to the server and processing the response.

```

php
<?php
// Assuming you have a login verification logic, replace the following with your actual validation code.
function verifyCredentials($username, $password) {
    // Replace this with your actual authentication logic.
    // For demonstration purposes, let's assume a simple check.
    $validUsername = "your_username";
    $validPassword = "your_password";

    if ($username === $validUsername && $password === $validPassword) {
        return true; // Valid credentials
    } else {
        return false; // Invalid credentials
    }
}

// Check if form is submitted
if ($_SERVER['REQUEST_METHOD'] === 'POST') {
    // Get the submitted credentials
    $username = isset($_POST['username']) ? $_POST['username'] : '';
    $password = isset($_POST['password']) ? $_POST['password'] : '';

    // Verify the credentials
    if (verifyCredentials($username, $password)) {
        echo "Welcome to UTU";
    } else {
        echo "You are not the legitimate user of the website";
    }
}
?>

<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Login Form</title>
</head>
<body>
    <form method="post" action="">
        <label for="username">Username:</label>
        <input type="text" name="username" required><br>

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

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

```

Please note that this is a basic example, and in a real-world scenario, you should use a secure password hashing mechanism, implement proper validation and sanitation, and consider other security measures to protect against common web vulnerabilities.



You:

5) Explain any six server variables.



Certainly! Here are explanations for six commonly used server variables in PHP:

#### 1. `$_SERVER['PHP_SELF']`:

- **Usage:** Contains the filename of the currently executing script, relative to the document root.
- **Example:** If your script is located at `http://example.com/myapp/index.php`, `$_SERVER['PHP_SELF']` would be `"/myapp/index.php"`.

#### 2. `$_SERVER['SERVER_NAME']`:

- **Usage:** Holds the server's host name or IP address.
- **Example:** If your website is hosted at [www.example.com](http://www.example.com), `$_SERVER['SERVER_NAME']` would be "[www.example.com](http://www.example.com)".

### 3. `$_SERVER['REQUEST_TIME']`:

- **Usage:** Represents the timestamp of the start of the request, useful for measuring script execution time.
- **Example:** Contains the Unix timestamp when the request started.

### 4. `$_SERVER['QUERY_STRING']`:

- **Usage:** Contains the query string of the URL, if any, through which the page was accessed.
- **Example:** If the URL is <http://example.com/page.php?param1=value1>, `$_SERVER['QUERY_STRING']` would be "param1=value1".

### 5. `$_SERVER['HTTP_ACCEPT']`:

- **Usage:** Holds the value of the Accept HTTP header, indicating the MIME types the client can understand.
- **Example:** It may contain values like "text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,/;q=0.8", indicating the client's preferred content types.

### 6. `$_SERVER['SCRIPT_NAME']`:

- **Usage:** Contains the path of the current script, useful for self-referencing scripts.
- **Example:** If your script is located at <http://example.com/myapp/index.php>, `$_SERVER['SCRIPT_NAME']` would be "/myapp/index.php".

These server variables provide valuable information about the server environment and the current request, allowing developers to access and manipulate various aspects of the script's execution.



You:

6) Write a PHP script to get the browser type and display belowmessage accordingly. "You are using <browser-type>!!"



ChatGPT:

Certainly! You can use the `$_SERVER['HTTP_USER_AGENT']` variable to get information about the user's browser. Here's a simple PHP script that displays a message based on the detected browser type:

```
php
<?php
// Get the user agent string
$userAgent = $_SERVER['HTTP_USER_AGENT'];

// Check for common browser patterns
if (strpos($userAgent, 'MSIE') !== false || strpos($userAgent, 'Trident') !== false) {
    $browserType = 'Internet Explorer';
} elseif (strpos($userAgent, 'Firefox') !== false) {
    $browserType = 'Firefox';
} elseif (strpos($userAgent, 'Chrome') !== false) {
    $browserType = 'Google Chrome';
} elseif (strpos($userAgent, 'Safari') !== false) {
    $browserType = 'Safari';
} elseif (strpos($userAgent, 'Opera') !== false || strpos($userAgent, 'OPR') !== false) {
    $browserType = 'Opera';
} else {
    $browserType = 'an unknown browser';
}

// Display the message
echo "You are using $browserType!!";
?>
```

This script checks for common patterns in the user agent string to identify the browser type. Keep in mind that the user agent can be manipulated or spoofed, so this method is not foolproof and may not always accurately reflect the user's actual browser. It provides a general idea based on common user agent patterns.



You:

7) Describe PHP filters to validate data user input.



ChatGPT:

PHP filters provide a convenient and powerful way to validate and sanitize user input. They are part of the PHP Filter Extension and are used to validate and sanitize data based on various pre-defined filters. These filters help ensure that input data meets specific criteria, reducing the risk of security vulnerabilities and improving data integrity. Here are some commonly used PHP filters:

### 1. Validation Filters:

- `FILTER_VALIDATE_EMAIL`: Validates an email address.
- `FILTER_VALIDATE_INT`: Validates an integer.
- `FILTER_VALIDATE_FLOAT`: Validates a floating-point number.
- `FILTER_VALIDATE_BOOLEAN`: Validates a boolean value (true/false).

#### Example:

```
php
$email = "user@example.com";
if (filter_var($email, FILTER_VALIDATE_EMAIL)) {
    echo "Valid email address";
} else {
    echo "Invalid email address";
}
```

### 2. Sanitization Filters:

- `FILTER_SANITIZE_STRING`: Strips tags, optionally strips or encodes special characters.
- `FILTER_SANITIZE_EMAIL`: Removes all illegal characters from an email address.
- `FILTER_SANITIZE_NUMBER_INT`: Removes all characters except digits and the plus/minus sign.
- `FILTER_SANITIZE_NUMBER_FLOAT`: Removes all characters except digits, period (.) and comma (,).

#### Example:

```
php
$input = "<p>Hello, world!</p>";
$sanitizedInput = filter_var($input, FILTER_SANITIZE_STRING);
echo $sanitizedInput; // Output: Hello, world!
```

### 3. Custom Validation with Callbacks:

- `FILTER_CALLBACK`: Allows the use of a custom function for validation.

#### Example:

```
php
function customValidation($value) {
    // Your custom validation logic here
    return ($value > 0 && $value < 100);
}

$number = 42;
if (filter_var($number, FILTER_CALLBACK, array('options' => 'customValidation'))) {
    echo "Valid number";
} else {
    echo "Invalid number";
}
```

Using PHP filters helps in creating more secure and robust applications by ensuring that input data is properly validated and sanitized before being used in the application logic. Always validate user input based on the specific requirements of your application to prevent security vulnerabilities like SQL injection or cross-site scripting (XSS).



You:

8) Explain various HTTP variables. in details



ChatGPT:

HTTP variables, often found in the `$_SERVER` superglobal in PHP, provide information about the server environment and the current HTTP request. Here are some of the key HTTP variables and their explanations:

#### 1. `$_SERVER['HTTP_USER_AGENT']`:

- **Usage**: Contains the user agent string of the browser making the request.
- **Example**: "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/91.0.4472.124 Safari/537.36".
- **Purpose**: It helps identify the type and version of the browser being used.

#### 2. `$_SERVER['HTTP_REFERER']`:

- **Usage:** Contains the URL of the referring page from which the current request was made (if available).
- **Example:** "<http://previouspage.com>".
- **Purpose:** Useful for tracking where the user came from.

### 3. \$\_SERVER['HTTP\_ACCEPT\_LANGUAGE']:

- **Usage:** Contains the preferred language of the user's browser.
- **Example:** "en-US,en;q=0.9".
- **Purpose:** It helps determine the language preferences of the user.

### 4. \$\_SERVER['HTTP\_ACCEPT\_ENCODING']:

- **Usage:** Contains the accepted content encodings (e.g., gzip, deflate) by the browser.
- **Example:** "gzip, deflate, br".
- **Purpose:** Useful for understanding which content encodings the client supports.

### 5. \$\_SERVER['HTTP\_HOST']:

- **Usage:** Contains the host header from the current request.
- **Example:** "[www.example.com](http://www.example.com)".
- **Purpose:** It helps identify the host to which the request is directed.

### 6. \$\_SERVER['REMOTE\_ADDR']:

- **Usage:** Contains the IP address of the client making the request.
- **Example:** "192.168.1.1".
- **Purpose:** Useful for tracking the IP address of the client.

### 7. \$\_SERVER['REQUEST\_METHOD']:

- **Usage:** Contains the request method used by the client (e.g., GET, POST, HEAD).
- **Example:** "GET".
- **Purpose:** Helps determine the type of request being made.

### 8. \$\_SERVER['SERVER\_PROTOCOL']:

- **Usage:** Contains the name and version of the communication protocol.
- **Example:** "HTTP/1.1".
- **Purpose:** Identifies the version of the HTTP protocol being used.

These HTTP variables provide valuable information about the client, the request, and the server environment. They are often used in web applications to customize responses based on the client's capabilities and preferences or for logging and debugging purposes. Keep in mind that information received from the client, such as user agent strings, can be manipulated and should not be solely relied upon for security-sensitive decisions.



You:

9) Write a PHP script to submit a registration form having following fields and print the response. Name: Gender: Area of Interest: (Networking, Security, Data Science, Cloud Computing) Address:



ChatGPT:

Certainly! Here's a basic example of a PHP script for handling a registration form with the specified fields:

```
php
<?php
// Initialize variables to store form data
$name = $gender = $interest = $address = '';

// Check if the form is submitted
if ($_SERVER['REQUEST_METHOD'] === 'POST') {
    // Retrieve form data
    $name = isset($_POST['name']) ? $_POST['name'] : '';
    $gender = isset($_POST['gender']) ? $_POST['gender'] : '';
    $interest = isset($_POST['interest']) ? $_POST['interest'] : '';
    $address = isset($_POST['address']) ? $_POST['address'] : '';

    // Display the submitted data
    echo "<strong>Registration Details:</strong><br>";
    echo "Name: $name<br>";
    echo "Gender: $gender<br>";
    echo "Area of Interest: $interest<br>";
    echo "Address: $address<br>";
}
?>

<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
```

```

<meta name="viewport" content="width=device-width, initial-scale=1.0">
<title>Registration Form</title>
</head>
<body>
<h2>Registration Form</h2>
<form method="post" action="">
  <label for="name">Name:</label>
  <input type="text" name="name" required><br>

  <label for="gender">Gender:</label>
  <select name="gender" required>
    <option value="male">Male</option>
    <option value="female">Female</option>
    <option value="other">Other</option>
  </select><br>

  <label for="interest">Area of Interest:</label>
  <select name="interest" required>
    <option value="Networking">Networking</option>
    <option value="Security">Security</option>
    <option value="Data Science">Data Science</option>
    <option value="Cloud Computing">Cloud Computing</option>
  </select><br>

  <label for="address">Address:</label>
  <textarea name="address" rows="4" required></textarea><br>

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

```

This script defines an HTML form with fields for name, gender, area of interest, and address. When the form is submitted, the PHP script retrieves the form data using the `$_POST` superglobal and prints the registration details. Note that this is a simple example, and in a real-world scenario, you would likely want to perform additional validation and sanitation of user input for security and data integrity reasons.



You:

10) Write a PHP script to check if user entered required data to the registration form of an e-commerce website.



ChatGPT:

Certainly! Below is a PHP script that checks if the required data is entered in the registration form for an e-commerce website. It includes basic validation to ensure that the required fields are not empty.

```

php
<?php
// Initialize variables to store form data and error messages
$name = $gender = $interest = $address = '';
$errors = [];

// Check if the form is submitted
if ($_SERVER['REQUEST_METHOD'] === 'POST') {
  // Retrieve form data
  $name = isset($_POST['name']) ? $_POST['name'] : '';
  $gender = isset($_POST['gender']) ? $_POST['gender'] : '';
  $interest = isset($_POST['interest']) ? $_POST['interest'] : '';
  $address = isset($_POST['address']) ? $_POST['address'] : '';

  // Validate required fields
  if (empty($name)) {
    $errors[] = "Name is required";
  }

  if (empty($gender)) {
    $errors[] = "Gender is required";
  }

  if (empty($interest)) {
    $errors[] = "Area of Interest is required";
  }

  if (empty($address)) {
    $errors[] = "Address is required";
  }
}

```

```
// Display registration details or errors
if (empty($errors)) {
    echo "<strong>Registration Details:</strong><br>";
    echo "Name: $name<br>";
    echo "Gender: $gender<br>";
    echo "Area of Interest: $interest<br>";
    echo "Address: $address<br>";
} else {
    echo "<strong>Error(s):</strong><br>";
    foreach ($errors as $error) {
        echo "- $error<br>";
    }
}
}
?>

<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Registration Form</title>
</head>
<body>
    <h2>Registration Form</h2>
    <form method="post" action="">
        <label for="name">Name:</label>
        <input type="text" name="name" required><br>

        <label for="gender">Gender:</label>
        <select name="gender" required>
            <option value="male">Male</option>
            <option value="female">Female</option>
            <option value="other">Other</option>
        </select><br>

        <label for="interest">Area of Interest:</label>
        <select name="interest" required>
            <option value="Networking">Networking</option>
            <option value="Security">Security</option>
            <option value="Data Science">Data Science</option>
            <option value="Cloud Computing">Cloud Computing</option>
        </select><br>

        <label for="address">Address:</label>
        <textarea name="address" rows="4" required></textarea><br>

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

This script includes basic checks for empty values in the required fields. If any required field is not filled in, it will display an error message. Otherwise, it will display the registration details. For a more robust solution, additional validation and sanitation should be implemented based on the specific requirements of your e-commerce application.



You:

11) Differentiate GET and POST methods. <https://www.guru99.com/difference-get-post-http.html>



ChatGPT:

Certainly! Here's a differentiation between the GET and POST methods:

### 1. Data Submission:

- **GET:** Appends data to the URL as a query string. Limited data can be sent (usually less than 2KB).
- **POST:** Sends data in the body of the HTTP request. Can handle larger amounts of data compared to GET.

### 2. Security:

- **GET:** Parameters are visible in the URL, making it less secure. Not suitable for sensitive data.
- **POST:** Parameters are not visible in the URL, enhancing security. Suitable for sensitive data like passwords.

### 3. Caching:

- **GET:** Data is cached by the browser and server because it's part of the URL.
- **POST:** Data is not cached by default, reducing the chance of sensitive data exposure.

#### 4. Bookmarking:

- **GET:** Parameters are visible in the URL, making it easy to bookmark.
- **POST:** Parameters are not visible in the URL, so it's not bookmark-friendly.

#### 5. Idempotence:

- **GET:** Considered idempotent; multiple identical requests should have the same effect as a single request.
- **POST:** Not necessarily idempotent; multiple identical requests might have different effects.

#### 6. Use Cases:

- **GET:** Suitable for data retrieval, like search queries or navigating to different pages.
- **POST:** Suitable for data submission, like form submissions, where data is sent to be processed.

#### 7. Visibility:

- **GET:** Parameters are visible in the URL and can be seen by the user.
- **POST:** Parameters are not visible in the URL, enhancing privacy.

#### 8. Encoding:

- **GET:** Parameters are URL-encoded and appended to the URL.
- **POST:** Parameters can be URL-encoded or sent in a more secure way (multipart/form-data).

#### 9. Browser Back/Forward Buttons:

- **GET:** Can be used with browser back/forward buttons, and the URL changes accordingly.
- **POST:** Navigating back/forward might prompt a browser warning due to resubmitting data.

For a more detailed explanation and examples, you can refer to the provided link: [Difference Between GET and POST Methods](#).



You:

12) Write a PHP script to upload a file and display the content of it.



ChatGPT:

Certainly! Below is a basic example of a PHP script that allows users to upload a file and then displays the content of the uploaded file. Note that this example assumes the uploaded file contains text data.

```

php
<?php
$uploadDirectory = "uploads/";

// Check if the form is submitted
if ($_SERVER['REQUEST_METHOD'] === 'POST' && isset($_FILES['file'])) {
    // Handle file upload
    $uploadedFile = $_FILES['file'];

    // Check for errors during upload
    if ($uploadedFile['error'] === UPLOAD_ERR_OK) {
        // Move the uploaded file to the desired directory
        $uploadPath = $uploadDirectory . basename($uploadedFile['name']);
        move_uploaded_file($uploadedFile['tmp_name'], $uploadPath);

        // Display the content of the uploaded file
        echo "<strong>Content of the uploaded file:</strong><br>";
        $fileContent = file_get_contents($uploadPath);
        echo nl2br(htmlspecialchars($fileContent)); // Display content with line breaks and HTML special characters escaped
    } else {
        echo "Error uploading file. Please try again.";
    }
}
?>

<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>File Upload and Display</title>
</head>
<body>
    <h2>Upload a File</h2>

```



```
<form method="post" action="" enctype="multipart/form-data">
  <label for="file">Select a file:</label>
  <input type="file" name="file" accept=".txt"> <!-- Accepts only text files for simplicity -->

  <input type="submit" value="Upload and Display">
</form>
</body>
</html>
```

This script includes an HTML form with a file input for selecting a file to upload. When the form is submitted, the PHP script handles the file upload, moves the uploaded file to the "uploads/" directory, and then displays the content of the uploaded file. This example is for educational purposes, and in a production environment, you should implement additional security measures and validations.



13) Write a PHP script to display a clickable image and redirect it to other websites based on the co-ordinates fetched.



Certainly! Below is a simple example of a PHP script that displays a clickable image and redirects to other websites based on the coordinates clicked on the image.

```
php
<?php
// Check if coordinates are provided in the query string
if (isset($_GET['x']) && isset($_GET['y'])) {
    // Extract coordinates from the query string
    $xCoordinate = $_GET['x'];
    $yCoordinate = $_GET['y'];

    // Check if coordinates fall within a specific region
    if ($xCoordinate >= 0 && $xCoordinate <= 100 && $yCoordinate >= 0 && $yCoordinate <= 100) {
        // Redirect to Website A
        header("Location: https://www.websiteA.com");
        exit();
    } elseif ($xCoordinate > 100 && $xCoordinate <= 200 && $yCoordinate >= 0 && $yCoordinate <= 100) {
        // Redirect to Website B
        header("Location: https://www.websiteB.com");
        exit();
    }
    // Add more conditions for additional regions as needed
}

// HTML content to display the clickable image
?>
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Clickable Image</title>
</head>
<body>
    <h2>Click on the image to redirect</h2>
    

    <map name="imagemap">
        <!-- Define clickable regions with coordinates -->
        <area shape="rect" coords="0,0,100,100" href="?x=50&y=50" alt="Website A">
        <area shape="rect" coords="101,0,200,100" href="?x=150&y=50" alt="Website B">
        <!-- Add more areas as needed -->
    </map>
</body>
</html>
```

This script uses an HTML image map with clickable areas defined by coordinates. When a user clicks on a specific region, the script redirects them to a corresponding website based on the provided coordinates. Adjust the coordinates and URLs as needed for your specific use case.



14) Discuss about \$\_FILES array of PHP to handle uploaded files.



In PHP, the `$_FILES` superglobal array is used to handle uploaded files in the context of form submissions with `enctype="multipart/form-data"`. When a user submits a form that includes a file input field, the uploaded file information is stored in the `$_FILES` array. This array provides details about the uploaded file, such as its name, type, size, and temporary location on the server.

Here are some key elements of the `$_FILES` array and how it works:

1. `$_FILES['userfile']['name']`:

- Contains the original name of the uploaded file.
- Example: `$fileName = $_FILES['userfile']['name'];`

2. `$_FILES['userfile']['type']`:

- Contains the MIME type of the uploaded file.
- Example: `$fileType = $_FILES['userfile']['type'];`

3. `$_FILES['userfile']['size']`:

- Contains the size of the uploaded file in bytes.
- Example: `$fileSize = $_FILES['userfile']['size'];`

4. `$_FILES['userfile']['tmp_name']`:

- Contains the temporary filename of the uploaded file on the server.
- Example: `$tempFileName = $_FILES['userfile']['tmp_name'];`

5. `$_FILES['userfile']['error']`:

- Contains an error code, if any, during the file upload process.
- Example: `$uploadError = $_FILES['userfile']['error'];`
- `UPLOAD_ERR_OK` (value 0) indicates a successful upload.

6. Handling File Upload:

- When a file is uploaded, it is typically moved from the temporary location to a permanent location on the server using `move_uploaded_file()` function.
- Example: `move_uploaded_file($_FILES['userfile']['tmp_name'], 'uploads/' . $_FILES['userfile']['name']);`

Here's a basic example of a form that allows file uploads:

```
html
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>File Upload Form</title>
</head>
<body>
  <form action="upload.php" method="post" enctype="multipart/form-data">
    <label for="file">Choose a file:</label>
    <input type="file" name="userfile" id="file">
    <br>
    <input type="submit" value="Upload File">
  </form>
</body>
</html>
```

And the corresponding PHP script (upload.php):

```
php
<?php
if ($_SERVER['REQUEST_METHOD'] === 'POST') {
  // Check if the file is uploaded successfully
  if ($_FILES['userfile']['error'] === UPLOAD_ERR_OK) {
    $fileName = $_FILES['userfile']['name'];
    $tempFileName = $_FILES['userfile']['tmp_name'];

    // Move the uploaded file to a permanent location
    move_uploaded_file($tempFileName, 'uploads/' . $fileName);

    echo "File uploaded successfully!";
  } else {
    echo "Error uploading file. Please try again.";
  }
}
```

```
}
?>
```

This is a basic example, and in a real-world scenario, you should perform additional validations, such as checking file types, file size limits, and implementing security measures to prevent potential security vulnerabilities.



You:

15) Write a PHP script to display following message on the browser. The <a> tag defines a hyperlink. Example <a href="http://www.utu.ac.in">Uka Tarsadia University</a>. To learn more about <a> tag click on the below link. HTML Tutorial



ChatGPT:

Certainly! Below is a PHP script that echoes the specified message to display it in the browser:

```
php
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>HTML Tutorial</title>
</head>
<body>
  <?php
    // PHP script to display the message
    echo "The <a href=\"http://www.utu.ac.in\">Uka Tarsadia University</a>. To learn more about <a href=\"#\">&lt;a&gt; t
    echo "HTML Tutorial";
  ?>
</body>
</html>
```

In this script, the PHP echo statement is used to output the specified HTML message. The <a> tags are properly escaped within the PHP script to ensure they are treated as plain text and not interpreted as HTML tags by the PHP parser.



You:

16) Write a PHP script to ensure that user entered age as integer only and name as text only.



ChatGPT:

Certainly! You can use PHP to validate the user-entered data and ensure that the age is an integer and the name is text only. Here's an example:

```
php
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>User Data Validation</title>
</head>
<body>

  <?php
    // Initialize variables
    $name = $age = '';
    $errors = [];

    // Check if the form is submitted
    if ($_SERVER['REQUEST_METHOD'] === 'POST') {
      // Validate name
      $name = isset($_POST['name']) ? $_POST['name'] : '';
      if (!preg_match("/^[a-zA-Z ]+$/", $name)) {
        $errors[] = "Name should contain only letters and spaces.";
      }

      // Validate age
```

```

$age = isset($_POST['age']) ? $_POST['age'] : '';
if (!ctype_digit($age)) {
    $errors[] = "Age should be an integer.";
}

// Display validation result
if (empty($errors)) {
    echo "Name: $name<br>";
    echo "Age: $age<br>";
    echo "Validation successful!";
} else {
    echo "<strong>Error(s):</strong><br>";
    foreach ($errors as $error) {
        echo "- $error<br>";
    }
}
}
?>

<form method="post" action="">
    <label for="name">Name:</label>
    <input type="text" name="name" value="<?php echo htmlspecialchars($name); ?>" required><br>

    <label for="age">Age:</label>
    <input type="text" name="age" value="<?php echo htmlspecialchars($age); ?>" required><br>

    <input type="submit" value="Submit">
</form>

</body>
</html>

```

In this script:

- The `preg_match` function is used to ensure that the name contains only letters and spaces.
- The `ctype_digit` function is used to check if the age is an integer.
- The `htmlspecialchars` function is used to prevent HTML injection by encoding special characters in the user-entered data.

Adjust the validation rules according to your specific requirements. This is a basic example, and in a real-world scenario, you might want to implement more comprehensive validation and sanitation based on your application's needs.



You:

17) Explain `header()` function of PHP along with demonstrative PHPscript.



ChatGPT:

The `header()` function in PHP is used to send raw HTTP headers to the browser. It is often used to send additional information to the browser before any actual content is sent. The `header()` function must be called before any actual output is sent to the browser, including HTML tags and whitespace.

The basic syntax of the `header()` function is as follows:

```

php
header("headerName: headerValue");

```

Here, "headerName: headerValue" represents the HTTP header that you want to send. Common headers include "Location" for redirection, "Content-Type" for specifying the type of content, and others.

Now, let's demonstrate the `header()` function with a simple example:

```

php
<?php
// Redirect the user to a different page after 5 seconds
header("Refresh: 5; URL=https://www.example.com");

// Set a custom HTTP response header
header("X-My-Header: Hello, World!");

// Output some HTML content
?>
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">

```

```

<meta name="viewport" content="width=device-width, initial-scale=1.0">
<title>Header Function Demo</title>
</head>
<body>
  <h1>Welcome to my website!</h1>
  <p>This page will redirect in 5 seconds.</p>
</body>
</html>

```

In this example:

- The Refresh header is used to instruct the browser to reload or redirect after a specified number of seconds. In this case, it redirects to "<https://www.example.com>" after 5 seconds.
- The X-My-Header header is a custom header that we set to "Hello, World!".

Remember that the header() function must be called before any actual output is sent to the browser. If there is any output (such as HTML content) before calling header(), it will result in an error.

The header() function is commonly used for tasks like redirection, setting content type, preventing caching, and more in PHP applications. However, be cautious when using it for security-sensitive operations, and ensure proper validation and sanitation of input to prevent header injection vulnerabilities.



You:

18) Write a PHP script to upload and download a .pdf and .php file.



ChatGPT:

Uploading and downloading files in a PHP script involves handling the file upload process, storing the files on the server, and providing a way for users to download them. Below is an example script that allows users to upload PDF and PHP files, stores them on the server, and provides download links:

### upload\_download.php

```

php
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>File Upload and Download</title>
</head>
<body>

<?php
$uploadDirectory = "uploads/";

// Handle file upload
if ($_SERVER['REQUEST_METHOD'] === 'POST' && isset($_FILES['userfile'])) {
    $uploadedFile = $_FILES['userfile'];

    // Check for errors during upload
    if ($uploadedFile['error'] === UPLOAD_ERR_OK) {
        // Move the uploaded file to the desired directory
        $uploadPath = $uploadDirectory . basename($uploadedFile['name']);
        move_uploaded_file($uploadedFile['tmp_name'], $uploadPath);

        echo "File uploaded successfully!";
    } else {
        echo "Error uploading file. Please try again.";
    }
}

?>

<h2>Upload File</h2>
<form method="post" action="" enctype="multipart/form-data">
  <label for="file">Choose a file (PDF or PHP):</label>
  <input type="file" name="userfile" accept=".pdf, .php">
  <br>
  <input type="submit" value="Upload File">
</form>

<hr>

<h2>Download Files</h2>

```

```
<?php
// List files in the upload directory for download
$files = scandir($uploadDirectory);
foreach ($files as $file) {
    if ($file !== "." && $file !== "..") {
        echo '<p><a href="download.php?filename=' . urlencode($file) . '"> . $file . '</a></p>';
    }
}
?>

</body>
</html>
```

### download.php

```
php
<?php
$uploadDirectory = "uploads/";

// Check if the filename is provided in the query string
if (isset($_GET['filename'])) {
    $filename = $_GET['filename'];

    // Construct the full path to the file
    $filePath = $uploadDirectory . $filename;

    // Check if the file exists
    if (file_exists($filePath)) {
        // Set appropriate headers for file download
        header('Content-Description: File Transfer');
        header('Content-Type: application/octet-stream');
        header('Content-Disposition: attachment; filename="' . $filename . '"');
        header('Expires: 0');
        header('Cache-Control: must-revalidate');
        header('Pragma: public');
        header('Content-Length: ' . filesize($filePath));

        // Read the file and output its contents
        readfile($filePath);
        exit();
    } else {
        echo "File not found.";
    }
} else {
    echo "Filename not provided.";
}
?>
```

Explanation:

#### 1. upload\_download.php:

- Handles file upload and displays a form to upload PDF or PHP files.
- Lists the files available for download.

#### 2. download.php:

- Checks if a filename is provided in the query string.
- If the file exists, sets appropriate headers for file download and outputs the file contents.

Note: Be careful when allowing users to upload and download PHP files, as it may pose security risks. Ensure proper validation, sanitization, and implement security measures to prevent unauthorized access or execution of PHP files.