Web Technology Solution
2021 Spring

Sanjaya (Bir Bikram) Shrestha

February 25, 2023

Contents

1 (Q No.1 A) What is web hosting? Explain any 4 types of web hosting.

Web hosting is a service that stores your website or web application and makes it easily accessible across different devices such as desktop, mobile, and tablets. Any web application or website is typically made of many files, such as images, videos, text, and code, that you need to store on special computers called servers. The web hosting service provider maintains, configures, and runs physical servers that you can rent for your files.

The four types of web hosting are:

1. Shared hosting

   Shared hosting is a type of hosting service where the server resources like disk, memory, cpu are shared among others users using the same web server. Only the resources are shared among the users but not the website content, files, directory.

   Advantages of shared hosting

   (a) Since the resources are shared among users, the cost of hosting will be cheap.
   (b) Multiple websites can be hosted from single account
   (c) Ease of setup and restart the services.

   Disadvantage of shared hosting

   (a) If one website get the high number of traffic then it will consume most of resources affecting other website performance.
   (b) Limited access to configuraiton and installation of 3rd party software.
   (c) Cost may increase substantialy as traffic increases

2. VPS hosting

   Virtual private server (VPS) hosting providers use a virtual private server to give more exclusive access to server resources. VPS hosting technology partitions the physical server to create small virtual servers that for lease. With virtual private servers, we can have exclusive access to both space and computing resources, such as memory and processing power, on the physical server. We can install an operating system on your virtual server and have greater control of your server environment. It is very suitable for:

   (a) medium sized business with high traffic
   (b) websites requires other software installation

   Advantages of VPS hosting

   (a) Provides complete controle over the server.
   (b) Greater control over website performance

   Disavantages of VPS hosting

   (a) Requires inhouse technical experts to maintaing the servers.

3. Dedicated hosting

Dedicated hosting give access to the complete physical server with exclusive access to a dedicated server. We can optimize and control the environment as per requirements. We can partition this server to host multiple domains on the same physical server. This is suitable for:

(a) large enterprises and business
(b) business owners with multiple services
(c) Web application with heavy processing powers required

Advantages of Dedicated hosting

(a) Very reliable and secured
(b) Access to server optimization capablitlies
(c) Customization of server conf

Disadvantages of Dedicated hosting

(a) Very expensive to lease and maintain
(b) Required inhouse technical experts

4. Cloud hosting Cloud hosting is a hosting service which uses resource sharing and scale to decrease web hosting costs and improve website performance. We can get shared access to a cluster of servers, and the cloud hosting provider automatically replicates website files across several servers. If one of the cloud servers is busy, the hosting provider automatically routes your traffic to another server. This greatly improves website performance without adding to your ongoing costs. This is suitable for:

(a) small and medium business
(b) Complex web application with multiple services
(c) Large enterprises with several domain

Advantages of Cloud hosting

(a) Greater availability and reliability for your website users.
(b) Automatically scales up or down as your requirements change.
(c) High flexibility—you pay for only what you use.

Disadvantages of Cloud hosting

(a) Vulnerable to attack
(b) Requires technical expert to maintain the servers requirement

2 (Q No.1 B) What is mime type specification in request/response transaction between browser and server?

In a request/response transaction between a browser and server, the MIME type specification is used to indicate the type of data that is being transmitted in the HTTP message. The MIME (Multipurpose Internet Mail Extensions) type is a standardized way to specify the format of data in Internet messages, and it helps the browser to know how to handle the content it receives from the server.

When a browser sends a request to a server, it includes an Accept header in the request message, which lists the MIME types that the browser can handle. The server then sends the response back to the browser, along with a Content-Type header that specifies the MIME type of the data being sent.

For example, if a user requests a webpage that contains an image, the browser will send a request to the server asking for the image file. The server will respond with the image data and a Content-Type header that specifies the MIME type of the image, such as "image/png" for a PNG file.

Similarly, when a user submits a form with data, the browser will send a request to the server with a Content-Type header that specifies the MIME type of the data being sent, such as "application/x-www-form-urlencoded" or "multipart/form-data".

By using MIME type specification, browsers and servers can communicate effectively and ensure that the data being transferred is interpreted correctly.

A MIME type is a string identifier composed of two parts: a type and a subtype.

1. "type" refers to a logical grouping of many MIME types that are closely related to each other; it's no more than a high level category.
2. "subtypes" are specific to one file type within the "type".

Examples of MIME types are:

1. text/html for normal web pages
2. text/plain for plain text
3. application/octet-stream meaning "download this file"
4. application/x-java-applet for Java™ applets
5. application/pdf for Adobe® PDF documents.

3 (Q No. 2A) What are the different typs of list available in HTML? Explain each with suitable availabe code.

There are 3 types of list in HTML:

Unordered List or Bulleted List (ul) This list is used when there is no specific order or sequence. An unordered list is also called a Bulleted list, as the items are marked with bullets. It begins with the ⟨ul⟩ tag and and closes with a ⟨/ul⟩ tag. The list items begin with the ⟨li⟩ tag and end with ⟨/li⟩ tag.

```
        <ul>
        <li>Apple</li>
        <li>Mango</li>
        <li>Banana</li>
        <li>Grapes</li>
        <li>Orange</li>
    </ul>

```

Listing 1: Unordered list example

Ordered List or Numbered List (ol) The list items in an ordered list are marked with numbers by default instead of bullets. An HTML ordered list starts with the ⟨ol⟩ tag and ends with the ⟨/ol⟩ tag. The list items start with the ⟨li⟩ tag and end with ⟨/li⟩ tag.

```
        <ol>
        <li>Apple</li>
        <li>Mango</li>
        <li>Banana</li>
        <li>Grapes</li>
        <li>Orange</li>
    </ol>

```

Listing 2: Ordered list example

Description List or Definition List (dl) the list items are listed like a dictionary or encyclopedia. Each item in the description list has a description. You can use a description list to display items like a glossary. You will need the following HTML tags to create a description list:

1. ⟨dl⟩ (Definition list) tag - Start tag of the definition list
2. ⟨dt⟩ (Definition Term) tag - It specifies a term (name)
3. ⟨dd⟩ tag (Definition Description) - Specifies the term definition
4. ⟨/dl⟩ tag (Definition list) - Closing tag of the definition list

```
    <dl>
    <dt><b>Apple</b></dt>
    <dd>A red colored fruit</dd>
    <dt><b>Honda</b></dt>
    <dd>A brand of a car</dd>
    <dt><b>Spinach</b></dt>
```

```
    <dd>A green leafy vegetable</dd>
 </dl>
```

Listing 3: Description list example

4 (Q No. 2B) Write Html tags to generate following form.

```
<html>
    <head> </head>
    <body>
    <table border="1" cellspacing="0">
    <tr><th>List with links </th> <th> Images </th><tr>
    <tr>
        <td>
            <ul type="none">
                <li>
                    Karnali
                    <ul>
                        <li>Bardiya</li>
                        <li>Banke</li>
                        <li>Jumla</li>
                    </ul>
                </li>

            </ul>
        </td>
        <td>
            <img src="./karnali.jpg" alt="Karnali">
        </td>
    </tr>
    <tr>
        <td>
            <ul type="none">
                <li>
                    Bagmati
                    <ul>
                        <li>Kathmandu</li>
                        <li>Bhaktapur</li>
                        <li>lalitpur</li>
                    </ul>
                </li>

            </ul>
        </td>
        <td>
            <img src="./bagmati.jpg" alt="Bagmati">
        </td>
    </tr>
    </table>
    </body>
    </html>
```

5 (Q No. 3A) What are the different level of stylesheets with their hierarchy of implementation? Explain the float and clear properties with clear and valid example?

There are 3 levels of stylesheets in CSS:

1. Inline Styles: These styles are applied directly to an HTML element using the style attribute. Inline styles have the highest specificity, meaning they override any other styles applied to the same element.

```
        <p style="color: red;">This text is red.</p>

```

Listing 4: Inline Style

2. Internal Styles: These styles are defined within the head section of an HTML document using the style tag. Internal styles have a higher specificity than external styles, but lower than inline styles.

```

        <head>
    <style>
        p {
        color: blue;
        }
    </style>
    </head>
    <body>
    <p>This text is blue.</p>
    </body>

```

Listing 5: Internal Style

3. External Styles: These styles are defined in an external CSS file and are linked to an HTML document using the link tag. External styles have the lowest specificity, but they are also the most reusable and easier to maintain.

```

    <head>
  <link rel="stylesheet" href="style.css">
</head>
<body>
  <p class="red-text">This text is red.</p>
</body>
```

Listing 6: External Style

Float is a CSS property used to move an element to the left or right of its container, allowing other elements to flow around it. The clear property is used to force an element to be placed below any floated elements.

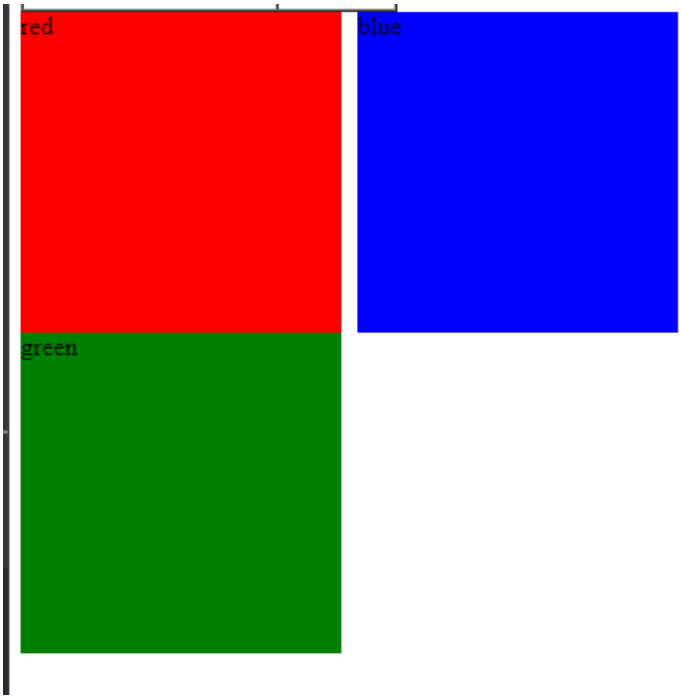
```
<style>
  .box {
    width: 200px;
```

```
    height: 200px;
    float: left;
    margin-right: 10px;
  }
  .clear {
    clear: both;
  }
</style>

<div class="box" style="background-color: red;">red</div>
<div class="box" style="background-color: blue;">blue</div>
<div class="clear"></div>
<div class="box" style="background-color: green;">green</div>
```

Listing 7: Internal Style

Here in the example above, we have three boxes floated to the left. The second box is positioned next to the first box because of the float property. However, we want the third box to be placed below the floated boxes, so we add a div with the clear property to force it to the next line.



6 (Q No. 3B) What are the logical and physical tags? Give examples also.

In Html there are 2 types of tags:

1. Logical tags are used to describe the purpose or meaning of the content. They are used to tell the browser to tell what kind of text is written inside the tags. Logical tags are also called Structural tags. Logical tags are used to indicate to the visually impaired person that there is something more important in the text or to emphasize the text ie, logical tags can be used for styling purposes as well as to give special importance to text content.

Here are some examples of logical tags:

```
<h1>, <h2>, <h3>, <p>, <ul>, <ol>, <li>, <table>, <form>, etc.
```

Listing 8: logical tags

Here are the usecases of logical tags

(a) To write code on our website.
(b) To Emphasize some text.
(c) To display the abbreviation on the Web page.
(d) To display some famous quotation on our web page.
(e) To write some mathematical formula in terms of variables.

Logical tags are used to structure the content of a web page and give it semantic meaning. For example, the ⟨h1 ⟩tag is used to indicate the main heading of a page, while the ⟨p ⟩tag is used to define a paragraph of text.

2. Physical tags: Physical tags are used to indicate that how specific characters are to be formatted or indicated using HTML tags. Any physical style tag may contain any item allowed in text, including conventional text, images, line breaks, etc. Physical tags can only be used for styling purposes for specific elements. Although each physical tag has a defined style, you can override that style by defining your own look for each tag. All physical tags require ending tags Here are the few physical tags.

```
<b>, <i>, <u>, <strike>, <font>, <center>, <br>, <hr>, etc.

```

Listing 9: physical tags

Here are the usecases of Physical tags:

(a) They are extremely straightforward.
(b) They are used to highlighting important sentences.
(c) Physical Text Styles indicate the specific type of appearance for a section e.g., bold, italics, etc.
(d) Physical Styles are rendered in the same manner by all browsers.

Physical tags are used to define the appearance of the content on a web page. For example, the ⟨b ⟩tag is used to make text bold, while the ⟨i ⟩tag is used to make text italic. The ⟨br ⟩tag is used to insert a line break, while the ⟨hr ⟩tag is used to insert a horizontal line.

7 (Q No. 3C) What do you mean by table-less design and how will you achieve it using CSS?

Table-less design, also known as CSS-based design, is an approach to web design that avoids using HTML tables for layout purposes. Instead, it uses CSS (Cascading Style Sheets) to control the layout and presentation of the content on a web page.

The primary advantage of table-less design is that it allows for more flexible and accessible web design, as well as faster load times. It also separates the content and presentation of a web page, making it easier to update and maintain.

To achieve table-less design using CSS, here are some key steps:

1. Using semantic HTML: Using logical tags such as

```
<header>, <main>, <nav>, <section>, and <article>
```

   to structure the content of your web page, helps to separate the content and structure from the presentation.

2. Using CSS layout techniques: Using CSS layout techniques such as Flexbox and CSS Grid to control the layout of the content on the web page. These techniques allow for more flexibility and control over the positioning of elements on the page.

3. Use CSS for styling: Use CSS to control the presentation of the content on your web page. This includes setting fonts, colors, borders, and other visual properties.

4. Avoid using HTML tables for layout: Use tables only for tabular data, not for layout purposes. Instead, use CSS layout techniques to achieve the desired layout.

5. Test for accessibility: Test your web page for accessibility using tools such as screen readers and keyboard navigation. Ensure that the content is accessible to all users, including those with disabilities.

```html
<html>
  <head>
    <title>Table-less Design Example</title>
    <link rel="stylesheet" type="text/css" href="style.css">
    <style>
        body {
            margin: 0;
            padding: 0;
            }
        #header {
        background-color: #ccc;
        }
        #content {
        padding: 10px;
        }
        #footer {
        background-color: #ccc;
        }
    </style>
  </head>
  <body>
    <div id="header">
      <h1>Header</h1>
    </div>
    <div id="content">
      <h2>Content</h2>
      <p>Hello world</p>
    </div>
    <div id="footer">
      <p>Copyright &copy; 2023</p>
    </div>
  </body>
</html>
```

8 (Q No. 4A) What do you mean my DOM in javascript? How do you get elements from document to javascript? Write example of all methods.

The Document Object Model (DOM) is a programming interface for web documents. It represents the page so that programs can change the document structure, style, and content. With the DOM, JavaScript can access and manipulate the elements of an HTML or XML document.

Here are the methods to get elements from the document to JavaScript:

1. **document.getElementById():** This method returns the element with the specified ID. For example, to get the element with ID "myElement", we can get using the code

```
var myElement = document.getElementById("myElement");

```

2. **document.getElementsByClassName():** This method returns a collection of elements with the specified class name. This method returns a collection, so we need to loop through the collection to access each element. For example, to get all elements with class name "myClass", we can use the following code:

```
var myElements = document.getElementsByClassName("myClass");

```

3. **document.getElementsByTagName():** This method returns a collection of elements with the specified tag name. This method also returns a collection. For example, to get all p elements on the page, we can use the following code:

```
var myElements = document.getElementsByTagName("p");

```

4. **document.querySelector():** This method returns the first element that matches a specified CSS selector. This method only returns the first matching element. For example, to get the first p element on the page, we can use the following code:

```
var myElement = document.querySelector("p");

```

5. **document.querySelectorAll():** This method returns a collection of elements that match a specified CSS selector. This method also returns a collection. For example, to get all p elements with class "myClass", we can use the following code:

```
var myElements = document.querySelectorAll("p.myClass");


```

With these methods, we can easily get elements from the document into JavaScript and manipulate them as needed.

9 (Q No. 4B) Write JavaScript program which validates the user data(name, email, contact no) and displays the success message if validated successfully. Also create HTML form for entering data.

```html
<!DOCTYPE html>
<html>
<head>
  <title>User Data Validation</title>
</head>
<body>
  <form action="qno5b.php" method="post">
    <label for="name">Name:</label>
    <input type="text" id="name" name="name"><br>

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

    <label for="contact">Contact No:</label>
    <input type="text" id="contact" name="contact"><br>

    <input type="submit" value="Submit" onclick="validateForm()">
  </form>

  <div id="message"></div>

  <script>
    function validateForm() {
  // Get form elements
  var nameInput = document.getElementById("name");
  var emailInput = document.getElementById("email");
  var contactInput = document.getElementById("contact");
  var messageDiv = document.getElementById("message");

  // Get input values
  var name = nameInput.value.trim();
  var email = emailInput.value.trim();
  var contact = contactInput.value.trim();

  // Validate input
  if (name === "") {
    messageDiv.innerHTML = "Please enter your name.";
    nameInput.focus();
    return false;
  }

  if (email === "") {
    messageDiv.innerHTML = "Please enter your email address.";
    emailInput.focus();
    return false;
  } else if (!validateEmail(email)) {
    messageDiv.innerHTML = "Please enter a valid email address.";
    emailInput.focus();
    return false;
  }

  if (contact === "") {
    messageDiv.innerHTML = "Please enter your contact number.";
    contactInput.focus();
    return false;
  } else if (!validateContact(contact)) {
    messageDiv.innerHTML = "Please enter a valid contact number.";
    contactInput.focus();
    return false;
```

```
  }

  // If all inputs are valid, display success message
  messageDiv.innerHTML = "Your data has been validated successfully!";
}

function validateEmail(email) {
  // Regular expression for email validation
  var emailRegex = /^[^\s@]+@[^\s@]+\.[^\s@]+$/;
  return emailRegex.test(email);
}

function validateContact(contact) {
  // Regular expression for contact number validation
  var contactRegex = /^\d{10}$/;
  return contactRegex.test(contact);
}

  </script>
</body>
</html>
```

10 (Q No. 5A) What do you mean by events and events handling? Write HTML and JS code to display digital clock.

An event is an action or occurrence that takes place in the web browser, such as a user clicking a button or a page finishing loading. Event handling is the process of writing code that responds to these events.

In HTML and JavaScript, we can use event listeners to handle events. An event listener is a function that is called when a particular event occurs. For example, we might attach an event listener to a button element, so that when the button is clicked, the function is executed.

```
<!DOCTYPE html>
<html>
<head>
  <title>Digital Clock</title>
</head>
<body>
  <h1>Digital Clock</h1>
  <p id="clock"></p>

  <script>
    // Get the clock element
var clockElement = document.getElementById("clock");

// Update the clock every second
setInterval(function() {
  var date = new Date();

  // Get hours, minutes, and seconds
  var hours = date.getHours();
  var minutes = date.getMinutes();
  var seconds = date.getSeconds();

  // Convert to 12-hour format
  var ampm = hours >= 12 ? 'pm' : 'am';
  hours = hours % 12;
```

```
  hours = hours ? hours : 12;

  // Add leading zeros
  if (minutes < 10) {
    minutes = '0' + minutes;
  }
  if (seconds < 10) {
    seconds = '0' + seconds;
  }

  // Display the time
  var time = hours + ':' + minutes + ':' + seconds + ' ' + ampm;
  clockElement.innerHTML = time;
}, 1000);

  </script>
</body>
</html>
```

11 (Q No. 5B) What are the different file handling methods available in PHP? After validating form value in 5(A) write the contents to the users.txt with time of login. Also redirect home.php after writing contents to the file. The content of file should look like this. Name: abs Sharma, Email: testuser@gmail.com Logged in on: Fri Feb 01 2020 17:00:00 GMT +0545 (Nepal Time)

PHP provides various file handling methods to read, write, and manipulate files. Here are some of the most commonly used file handling methods in PHP:

1. **fopen():** This function is used to open a file in PHP. It returns a file pointer, which is used by other file handling functions to perform operations on the file.
2. **fclose():** This function is used to close a file that has been opened using fopen().
3. **fread():** This function is used to read data from a file. It takes two parameters: the file pointer returned by fopen(), and the number of bytes to read.
4. **fwrite():** This function is used to write data to a file. It takes two parameters: the file pointer returned by fopen(), and the data to write.
5. **feof():** This function is used to check if the end of the file has been reached.
6. **fgets():** This function is used to read a line from a file. It takes the file pointer returned by fopen() as a parameter.

Here is the PHP code to achieve what is requested writing form data to file, with timestamp.

```
<?php
// Validate form data
$name = $_POST['name'];
$email = $_POST['email'];
```

```php
$contact = $_POST['contact'];

  // Form data is valid, proceed to write to file
  $time = date("D M d Y H:i:s e0");
  $content = "Name: " . $name . ", Email: " . $email . ", Contact No.: " .
    $contact . " Logged in on: " . $time . " using PHP\n";

  // Open the file for writing
  $file = fopen("users.txt", "a");

  // Write the content to the file
  fwrite($file, $content);

  // Close the file
  fclose($file);

  // Redirect to home page
  header("Location: home.php");
  exit();

?>
```

12 (Q No. 6A) Explain the use of PHP $_POST, $_GET and $_REQUEST variables.

In PHP, $_POST, $_GET, and $_REQUEST are superglobal variables used to collect data submitted in HTML forms, as well as data appended to the URL. Here's an explanation of each variable:

1. **$_POST:** This variable is used to collect data that is submitted through an HTML form with the HTTP POST method. The data is encoded in the message body of the HTTP request. The data is accessible using the name attribute of the input field in the form. For example:

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

```

Listing 10: form.html

   In the example above, the $_POST variable can be used in process.php to get the values of the username and password fields:

```php
        $username = $_POST['username'];
        $password = $_POST['password'];

```

Listing 11: process.php

2. **$_GET:** This variable is used to collect data that is appended to the URL with the HTTP GET method. The data is visible in the URL query string, and can be accessed using the key-value pairs in the query string. For example:

```
<a href="profile.php?username=johndoe">View profile</a>
```

Listing 12: link.html

In the example above, the $_GET variable can be used in profile.php to get the value of the username parameter:

```
$username = $_GET['username'];
```

Listing 13: profile.php

3. **$_REQUEST:** This variable is used to collect data submitted through either $_POST or $_GET. The data can be accessed using the name attribute of the input field in the form or the key-value pairs in the query string. For example:

```
$username = $_REQUEST['username'];
```

Listing 14: process.php

In the example above, the $_REQUEST variable can be used to get the value of the username parameter regardless of whether it was submitted through $_POST or $_GET.

13 (Q No. 5B) Write PHP-MySQL code validate the user login data (email and password). That is check whether the user data is available in database or not. Also in another file write code to display the user information (name and email). (Assume database name to "testdb" and table name is "users")

We first connect to the MySQL database using

$$mysqli\_connect\,()$$

function. We then check if the form has been submitted using

$$\$\_SESSION\,["REQUEST\_METHOD"]$$

variable. If it has, we get the values of email and password fields using $_POST variable.

We then validate the user login data against the database by querying the users table using

$$mysqli\_query\,()$$

function. If the query returns a row, we start a session and set the email value to

$$\$\_SESSION\,["email"]$$

. We then redirect to userinfo.php using

$$header()$$

function. If the query returns no rows, we display an error message. Here is the login.php file source code for logging in users.

```php
<?php
// Connect to MySQL database
$servername = "localhost";
$username = "root";
$password = "";
$dbname = "testdb";

$conn = mysqli_connect($servername, $username, $password, $dbname);

// Check connection
if (!$conn) {
  die("Connection failed: " . mysqli_connect_error());
}

// Check if form submitted
if ($_SERVER['REQUEST_METHOD'] === 'POST') {
  $email = $_POST['email'];
  $password = $_POST['password'];

  // Validate user login data against database
  $sql = "SELECT * FROM users WHERE email = '$email' AND password = '$password'";
  $result = mysqli_query($conn, $sql);

  if (mysqli_num_rows($result) > 0) {
    // Login successful
    session_start();
    $_SESSION['email'] = $email;

    // Redirect to user information page
    header("Location: userinfo.php");
    exit();
  } else {
    // Login unsuccessful
    echo "Invalid email or password";
  }
}

mysqli_close($conn);
?>
```

Listing 15: login.php

We then check if the user is logged in by checking if

$$\$\_SESSION\,[\text{"}email\text{"}]$$

is set. If it is not set, we redirect to login.php using header function.

If the user is logged in, we get the user information from the database by querying the users table using

$$mysqli\_query\,()$$

function. If the query returns a row, we display the user information. If the query returns no rows, we display an error message

Here's an example PHP-MySQL code to display user information based on the email value stored in the session:

```php
<?php
// Connect to MySQL database
```

```php
$servername = "localhost";
$username = "root";
$password = "";
$dbname = "testdb";

$conn = mysqli_connect($servername, $username, $password, $dbname);

// Check connection
if (!$conn) {
  die("Connection failed: " . mysqli_connect_error());
}

// Start session
session_start();

// Check if user logged in
if (!isset($_SESSION['email'])) {
  header("Location: login.php");
  exit();
}

// Get user information from database
$email = $_SESSION['email'];
$sql = "SELECT * FROM users WHERE email = '$email'";
$result = mysqli_query($conn, $sql);

if (mysqli_num_rows($result) > 0) {
  $row = mysqli_fetch_assoc($result);
  $name = $row['name'];
  $email = $row['email'];

  // Display user information
  echo "Name: $name<br>";
  echo "Email: $email";
} else {
  echo "User not found";
}

mysqli_close($conn);
?>
```

Listing 16: userinfo.php

14 (Q No. 7A) Hyperlinks and Hypermedia

Hyperlinks and hypermedia are concepts related to linking and navigating between web pages and resources.

Hyperlinks are links that allow users to navigate from one web page to another by clicking on them. Hyperlinks can be represented as text, images, or other types of media and are usually highlighted or underlined to indicate that they are clickable. For example, clicking on a hyperlink in a blog post may take you to a related article or another website.

Hypermedia, on the other hand, refers to the use of different types of media, such as text, images, video, and audio, to link and navigate between resources. In addition to linking to other web pages, hypermedia can also link to different types of media files, such as images or videos. For example, a hypermedia document may contain clickable images that take users to related articles

or videos.

Hyperlinks and hypermedia are fundamental concepts in web design and are used to create intuitive and interactive user experiences. They allow users to navigate between related content and resources, helping them to discover and consume information more easily. Here are some example of hyperlink and hypermedia

1. **Hyperlink** This creates hyperlink to a website.

   $$< ahref = "https://www.example.com" > ClickheretovisitExamplewebsite < /a >$$

2. **Image Hyperlink** This code creates a clickable image that takes users to the website "https://www.example.com" when clicked. The image file "example.png" is displayed as the clickable element.

   $$< ahref = "https://www.example.com" >< imgsrc = "example.png" alt = "Examplewebsite" >< /a >$$

3. **Video Hyperlink** This code creates a hyperlink that links to a video file "video.mp4" located at "https://www.example.com". When clicked, the video will start playing.

   $$< ahref = "https://www.example.com/video.mp4" > Watchthevideohere < /a >$$

4. **Hypermedia** This code creates a clickable element that includes an image and text. When clicked, users will be taken to the website "https://www.example.com". The image and text provide additional context and information about where the hyperlink will take them.

   $$< ahref = "https://www.example.com" >< imgsrc = "example.png" alt = "Examplewebsite" >< p > C$$

15 (Q No. 7B) Pop up boxes in JavaScript

Pop up boxes in JavaScript are used to display messages or alerts to users. There are three types of pop up boxes in JavaScript: alert boxes, confirm boxes, and prompt boxes.

1. **Alert Boxes** Alert boxes are used to display a message to the user. They are useful for displaying important information that requires the user's attention. Here's an example:

```
        alert("Hello, World!");

```

2. **Confirm Boxes:** Confirm boxes are used to ask the user to confirm or cancel an action. They are useful for getting the user's input before performing an action that could have consequences. Here's an example:

```
        if (confirm("Are you sure you want to delete this item?")) {
        // Perform the deletion
      } else {
        // Cancel the deletion
      }

```

   When this code is executed, a confirm box will pop up with the message "Are you sure you want to delete this item?". If the user clicks the "OK" button, the code inside the if statement will be executed. If the user clicks the "Cancel" button, the code inside the else statement will be executed.

3. **Prompt Boxes:** Prompt boxes are used to ask the user to enter some text. They are useful for getting input from the user before performing an action. Here's an example:

```
        var name = prompt("Please enter your name:");
        alert("Hello, " + name + "!");

```

When this code is executed, a prompt box will pop up with the message "Please enter your name:". The user can enter their name and click the "OK" button. The code then uses the entered name to display a personalized message.

16 (Q No. 7C) Types of array in PHP

In PHP, there are three types of arrays:

1. **Indexed arrays:** An indexed array stores a collection of values, each of which is assigned a numeric index. The index starts at 0 and increases by 1 for each element in the array. Indexed arrays can be created using the array() function or using the shorthand [] syntax. Here's an example:

```
        $fruits = array("apple", "banana", "orange");
        // or
        $fruits = ["apple", "banana", "orange"];

```

2. **Associative arrays:** An associative array stores a collection of key-value pairs, where each key is a string and each value can be of any type. Associative arrays can be created using the

$$array()$$

function or using the shorthand [] syntax, but the key-value pairs must be explicitly defined using the =¿ operator. Here's an example:

```
        $person = array("name" => "John", "age" => 30, "email" => "
    john@example.com");
        // or
        $person = ["name" => "John", "age" => 30, "email" => "john@example.com
    "];


```

3. **Multi dimensional array:** Multidimensional array is an array that contains one or more arrays as its elements. Each element of a multidimensional array can be an array itself, which can contain further arrays as its elements.

   Multidimensional arrays are useful for representing complex data structures, such as tables or matrices. They can be indexed using multiple sets of keys, which allows for flexible and efficient data access.

```
        \$data = array(
            array("John", "Doe", 25),
            array("Jane", "Smith", 30),
            array("Bob", "Johnson", 40)
        );

```