Web Technology Solution
2021 Fall

Sanjaya (Bir Bikram) Shrestha

February 25, 2023

Contents

1 (Q No.1 A) Discuss the need of web browser and web server in accessing the Internet. Also list the different MIME header and explain them. 3

2 (Q No.1 B) Explain the essential differnce between
i) URL and URI
ii) HTTP and FTP? 4

3 (Q No. 2A) Why table-less design is preferred more than table based design. Write HTML code for the following. 5

4 (Q No. 2B) Explain Standard HTML document Structure. How would you map image in HTML. 6

5 (Q No. 3A) Create a HTML page with a div with some content and two paragraph tags with some content having ID P1 and P2. Write external CSS for the div tag having fixed position with border style solid and width 2px. The P1 should have relative position. The font type of P1 should be Arial and color should be green. The P2 have absolute position with top 0px and left 200px. 8

6 (Q No. 3B) How can you validate form in JavaScript? Write a program to explain it. Your program must contain email, password and submit button. On clicking, validate whether the fields are blank and the format for the email is right. 9

7 (Q No. 4A) What do you mean by Events and Events handling? How will you locate mouse cursor using JS. Explain with code. 10

8 (Q No. 4B) What is DOM? Write a code synopsis to represent the dragging and dropping of elements in JavaScript. 11

9 (Q No. 5A) How can you define variable in PHP? Define any two variable of String types and display their results after concatenation. 12

10 (Q No. 5B) How is Array implemented in PHP? Differentiate between foreach() loop and each() function with example. 13

11 (Q No. 6A) Make a login form with email and password. Write a program to make dynamic login using PHP and MySQL query. Your program must contain database connection file from PHP to MySQL. 13

12 (Q No. 6B) Describe Starting, Storing and Destroying a Session. Explain with sample code. Also differentiate between Session and Cookies. 15

13 (Q No. 7A) DOM Hierarchy in JavaScript 16

14 (Q No. 7B) Client Side vs Server Side scripting 17

15 (Q No. 7C) Ordered list vs Unordered list 18

1 (Q No.1 A) Discuss the need of web browser and web server in accessing the Internet. Also list the different MIME header and explain them.

Web browsers and web servers are two key components required for accessing the Internet. A web browser is an application that allows users to access and navigate websites, while a web server is a program that hosts and delivers web content over the Internet.

The need for a web browser arises from the fact that websites are designed to be accessed and displayed in a particular format, typically using HTML, CSS, and JavaScript. A web browser interprets this code and presents it to the user in a graphical interface, allowing them to navigate the website and interact with its content. Without a web browser, users would not be able to access websites or view their content.

On the other hand, web servers are responsible for hosting and delivering the content that users access via their web browsers. When a user requests a webpage, their web browser sends a request to the web server hosting that page, and the server responds by sending the necessary files to the browser. This exchange of information is what allows users to access websites and view their content.

In summary, web browsers and web servers are essential components required for accessing the Internet. While web browsers provide users with a graphical interface for accessing and navigating websites, web servers host and deliver the content that users access via their browsers. Without these components, the Internet as we know it today would not be possible. The four types of web hosting are:

1. **Content-Type:** This header specifies the type of data being transmitted. It includes information such as the media type (e.g., text, image, audio), subtype (e.g., HTML, JPEG, MP3), and character set (e.g., UTF-8, ISO-8859-1) of the data.

2. **Content-Disposition:** This header is used to suggest a filename for the data being transmitted, as well as to provide instructions on how to handle the data (e.g., whether to display it inline or prompt the user to download it).

3. **Content-Encoding:** This header specifies the encoding used to compress the data being transmitted (e.g., gzip, deflate).

4. **Content-Language:** This header indicates the language in which the data is written.

5. **Content-Length:** This header specifies the size of the data being transmitted, in bytes.

6. **Content-Location:** This header provides a URL that can be used to access the data being transmitted.

7. **Content-Transfer-Encoding:** This header specifies the encoding used to transmit the data (e.g., base64).

   By including these headers in HTTP requests and responses, MIME allows web browsers and servers to communicate and transmit data in a standardized and efficient manner.

2 (Q No.1 B) Explain the essential differnce between i) URL and URI ii) HTTP and FTP?

i) URL and URI

Here are some differences between URI and URL with examples:

1. **Purpose:** A URI is a string of characters that identifies a resource, while a URL is a specific type of URI that provides the location of a resource.
   Example:
   mailto:example@example.com is a URI that identifies an email address.
   http://www.example.com is a URL that identifies the location of a website.

2. **Structure:** URIs can have various structures, including URLs. A URL must have a specific structure that includes the protocol, domain name, and path.
   Example:
   urn:isbn:0451450523 is a URI that identifies a book by its ISBN number.
   ftp://ftp.example.com/files/document.pdf is a URL that identifies the location of a PDF file on an FTP server.

3. **Scope:** URIs can identify any resource, including physical resources such as books or people. URLs only identify web resources.
   Example:
   tel:+1-202-555-0166 is a URI that identifies a telephone number.
   https://www.example.com is a URL that identifies the location of a website.

ii) HTTP and FTP

FTP (File Transfer Protocol) and HTTP (Hypertext Transfer Protocol) are both protocols used for transferring data over the Internet, here are some key differences.

1. **Purpose:** FTP is primarily used for transferring files between a client and a server, while HTTP is primarily used for accessing web resources, such as HTML pages and images.

2. **Port:** FTP uses port 21 for control messages and port 20 for data transfers, while HTTP uses port 80 for regular connections and port 443 for secure connections (HTTPS).

3. **Connection Type:** FTP establishes a separate connection for each transfer, while HTTP uses a persistent connection for multiple requests.

4. **Authentication:** FTP supports multiple authentication methods, including username/password and anonymous access. HTTP primarily uses cookies and session IDs for authentication.

5. **Encryption:** FTP does not encrypt data in transit by default, while HTTPS encrypts data using SSL/TLS encryption.

6. **Data Transfer:** FTP is optimized for large file transfers, while HTTP is optimized for smaller, more frequent data transfers.

7. **URL Format:** FTP URLs start with "ftp://" followed by the server name and file path, while HTTP URLs start with "http://" or "https://" followed by the server name and resource path.

---

3 (Q No. 2A) Why table-less design is preferred more than table based design. Write HTML code for the following.

Table-less design, also known as "div-based" or "CSS-based" design most popular in HTML and web design. Here are some reasons why table-less design is preferred over table-based design:

1. **Accessibility:** Table-less design is more accessible to users with disabilities, particularly those using screen readers, because it allows for more semantic markup and easier navigation through the content.
2. **Flexibility:** Table-less design allows for more flexibility in terms of layout and design, allowing designers to create more complex and dynamic designs with greater precision and ease.
3. **Loading speed:** Table-less design can lead to faster loading times because it requires less code to render the page and allows for more efficient use of CSS stylesheets.
4. **Search engine optimization (SEO):** Table-less design is more SEO-friendly because it allows search engines to more easily crawl and index the content of the page.
5. **Maintenance:** Table-less design is easier to maintain and update because it separates the content and structure of the page from the presentation and styling, making it easier to make changes to the design without affecting the content.

Overall, table-less design offers a more accessible, flexible, efficient, and maintainable approach to HTML and web design, making it a preferred choice for many web developers and designers.

```
<html>
<head>

    <title>Document</title>
</head>
<body>
    <table border="1" cellspacing="0" cellpadding="5">
        <tr>
            <th rowspan="2">Roll No</th>
            <th rowspan="2">Name</th>
            <th colspan="4">Subjects</th>
        </tr>
        <tr>
            <td>C</td> <td>Web</td> <td>Math</td> <td>C++</td>
        </tr>
        <tr>
            <td>1</td> <td>Abhisek</td> <td>50</td> <td>45</td> <td>60</td> <td>70
    </td>
        </tr>
        <tr>
```

```
            <td>2</td> <td>Suresh</td> <td>50</td> <td>45</td> <td>45</td> <td>NQ</td>
        </tr>
        <tr>
            <td>3</td> <td>Yodhin</td> <td>80</td> <td>85</td> <td>90</td> <td>75</td>
        </tr>
        <tr>
            <td colspan="6"> <b>Contact to department if not qualified (NQ)</b> </td>
        </tr>
    </table>
</body>
</html>
```

4 (Q No. 2B) Explain Standard HTML document Structure. How would you map image in HTML.

A standard HTML document structure includes several elements that help structure the content of the document and provide important information about the document's metadata. Here is an overview of the basic elements that make up a standard HTML document structure:

1. The <!DOCTYPE> declaration: This is the very first line of an HTML document and indicates the document type and version. For example, <!DOCTYPE html> declares that the document is an HTML5 document.
2. **The <html> element:** This is the root element of an HTML document and contains all other elements in the document. The <html> element is followed by two other elements: <head> and <body>.
3. **The <head> element:** This element contains metadata about the document, including the document's title, keywords, and descriptions. It also contains references to external files such as CSS stylesheets and JavaScript files.
4. **The <title> element:** This element is nested within the <head> element and defines the title of the document. The title is displayed in the browser's title bar and is also used by search engines to identify the document.
5. **The <body> element:** This element contains the main content of the document, including text, images, and other media. All visible content of the document is contained within the <body> element.
6. **Structural elements:** Within the <body> element, structural elements such as headings (<h1>, <h2>, etc.), paragraphs (<p>), and lists (<ul>, <ol>, <dl>) are used to organize the content and create a hierarchical structure.
7. **The <footer> element:** This element contains information about the document such as copyright notices, author information, and contact information.

Here is an example of a basic HTML document structure:

```html
<!DOCTYPE html>
<html>
  <head>
    <title>My Document</title>
    <meta name="description" content="A description of my document">
    <link rel="stylesheet" href="style.css">
  </head>
  <body>
    <header>
      <h1>My Document</h1>
    </header>
    <main>
      <article>
        <h2>Introduction</h2>
        <p>This is the introduction to my document.</p>
      </article>
      <article>
        <h2>Conclusion</h2>
        <p>This is the conclusion to my document.</p>
      </article>
      <footer>
        &copy; Copyright
      </footer>
    </main>
  </body>
</html>
```

Listing 1: Basic Html document

To map an image in HTML, you can use the <map> and <area> elements to define clickable regions on the image. Here's an example:

```html
    <img src="https://fakeimg.pl/300/" alt="My Image" usemap="#myMap">

    <map name="myMap">
    <area shape="rect" coords="0,0,100,100" href="page1.html" alt="Page 1">
    <area shape="rect" coords="100,0,200,100" href="page2.html" alt="Page 2">
    <area shape="circle" coords="150,150,50" href="page3.html" alt="Page 3">
    </map>
```

Listing 2: Image map

When the user clicks on one of the mapped areas, the browser will navigate to the URL specified in the "href" attribute. This allows you to create interactive images that can act as navigation menus, image maps, and more.

5 (Q No. 3A) Create a HTML page with a div with some content and two paragraph tags with some content having ID P1 and P2. Write external CSS for the div tag having fixed position with border style solid and width 2px. The P1 should have relative position. The font type of P1 should be Arial and color should be green. The P2 have absolute position with top 0px and left 200px.

Here's an example HTML code that creates a page with a div tag containing some content and two p tags with IDs P1 and P2:

```
<!DOCTYPE html>
<html>
<head>
  <title>My Page</title>
  <link rel="stylesheet" type="text/css" href="styles.css">
</head>
<body>
  <div id="myDiv">
    This is some content inside the div tag.
    <p id="P1">This is the first paragraph.</p>
    <p id="P2">This is the second paragraph.</p>
  </div>
</body>
</html>
```

Listing 3: Html document: index.html

And here is the external css file code with gives styles.

```
#myDiv {
  position: fixed;
  border: 2px solid black;
}

#P1 {
  position: relative;
  font-family: Arial, sans-serif;
  color: green;
}

#P2 {
  position: absolute;
  top: 0px;
  left: 200px;
}
```

Listing 4: External css file: styles.css

6 (Q No. 3B) How can you validate form in JavaScript? Write a program to explain it. Your program must contain email, password and submit button. On clicking, validate whether the fields are blank and the format for the email is right.

To validate a form in JavaScript, we can use the submit event of the form element and write a JavaScript function to check the input values and display error messages if necessary. Here's an example program that demonstrates form validation in JavaScript:

```
<!DOCTYPE html>
<html>
<head>
  <title>Form Validation</title>
</head>
<body>
  <form id="myForm">
    <label>Name:</label>
    <input type="text" id="name" name="name"><br>

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

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

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

  <script>
    document.getElementById("myForm").addEventListener("submit", function(event) {
      event.preventDefault(); // prevent the default form submission

      // get the input values
      var name = document.getElementById("name").value;
      var email = document.getElementById("email").value;
      var password = document.getElementById("password").value;

      // validate the input values
      if (name.trim() == "") {
        alert("Name is required.");
        return false;
      }

      if (email.trim() == "") {
        alert("Email is required.");
        return false;
      }

      // check email format using regular expression
      var emailRegex = /^\w+([\.-]?\w+)*@\w+([\.-]?\w+)*(\.\w{2,3})+$/;
      if (!email.match(emailRegex)) {
        alert("Invalid email format.");
        return false;
      }

      if (password.trim() == "") {
        alert("Password is required.");
```

```
        return false;
      }

      // if all input values are valid, submit the form
      alert("Form submitted successfully.");
      document.getElementById("myForm").submit();
    });
  </script>
</body>
</html>
```

Listing 5: Form Validation

To validate the form, we have added an event listener to the form's submit event using the addEventListener() method. Inside the event listener function, we first prevent the default form submission using the preventDefault() method.

We then get the input values using the value property of each input field's element. We then validate each input value by checking if it is empty or not using the trim() method to remove any leading or trailing spaces. we have added a regular expression emailRegex that matches the valid email format. We then use the match() method to check if the email value matches this regular expression. If any input value is invalid, we display an error message using the alert() method and return false to prevent the form submission.

If all input values are valid, we display a success message using the alert() method and submit the form using the submit() method of the form element.

---

7 (Q No. 4A) What do you mean by Events and Events handling? How will you locate mouse cursor using JS. Explain with code.

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.

To locate the mouse cursor in JavaScript, you can use the mousemove event and the clientX and clientY properties of the event object to get the current position of the cursor.

Here's an example code snippet that demonstrates how to locate the mouse cursor and display its position on the page:

```
<!DOCTYPE html>
<html>
<head>
  <title>Mouse Location</title>
  <style>
    #cursor-location {
      position: fixed;
      top: 10px;
      left: 10px;
    }
  </style>
</head>
<body>
```

```
  <div id="cursor-location"></div>

  <script>
    // add a mousemove event listener to the document
    document.addEventListener("mousemove", function(event) {
      // get the current mouse position
      var x = event.clientX;
      var y = event.clientY;

      // update the text of the cursor location element
      var cursorLocation = document.getElementById("cursor-location");
      cursorLocation.innerHTML = "Mouse position: " + x + ", " + y;
    });
  </script>
</body>
</html>
```

Listing 6: Locate Mouse Cursor

In this example, we add a mousemove event listener to the document element using the addEventListener() method. Whenever the mouse is moved, the function specified in the event listener is called.

Inside the event listener function, we use the clientX and clientY properties of the event object to get the current position of the mouse cursor. We then update the text content of a div element with id=”cursor-location” to display the current position of the mouse cursor.

---

8 (Q No. 4B) What is DOM? Write a code synopsis to represent the dragging and dropping of elements in JavaScript.

The Document Object Model (DOM) is a programming interface for HTML and XML documents. It represents the web page as a structured document that can be accessed and manipulated using programming languages such as JavaScript. When a web page is loaded into a web browser, the browser creates a DOM representation of the page. This representation is organized as a tree-like structure called the DOM tree. The root of the tree is the document object, which represents the entire web page.

```
<!DOCTYPE HTML>
<html>
<head>
<style>
#div1 {
  width: 350px;
  height: 70px;
  padding: 10px;
  border: 1px solid #aaaaaa;
}
</style>
<script>
function allowDrop(ev) {
  ev.preventDefault();
}

function drag(ev) {
  ev.dataTransfer.setData("text", ev.target.id);
```

```
}

function drop(ev) {
  ev.preventDefault();
  var data = ev.dataTransfer.getData("text");
  ev.target.appendChild(document.getElementById(data));
}
</script>
</head>
<body>

<p>Drag the W3Schools image into the rectangle:</p>

<div id="div1" ondrop="drop(event)" ondragover="allowDrop(event)"></div>
<br>
<img id="drag1" src="https://fakeimg.pl/300/" draggable="true" ondragstart="drag(
    event)" width="336" height="69">

</body>
</html>
```

Listing 7: Drag and Drop

To make an element draggable we set the draggable attribute to true of the element. The ondragstart attribute calls a function, drag(event), that specifies what data to be dragged.

The dataTransfer.setData() method sets the data type and the value of the dragged data: The ondragover event specifies where the dragged data can be dropped.

---

9 (Q No. 5A) How can you define variable in PHP? Define any two variable of String types and display their results after concatenation.

In PHP, we can define a variable by using the dollar sign ($) followed by the variable name. PHP is a dynamically-typed language, which means that you don't need to specify the data type of a variable when you declare it.

Here's an example of defining two string variables and displaying their concatenated results:

```
    <?php
    $name1 = "John";
    $name2 = "Doe";
    $fullname = $name1 . " " . $name2;
    echo $fullname;
    ?>
```

In the above code, we have defined two string variables, $name1 and $name2, and assigned them values "John" and "Doe" respectively.
We have then concatenated these two variables using the dot (.) operator and stored the result in a new variable $fullname.
Finally, we have displayed the value of $fullname using the echo statement.
The output of the above code will be:

```
    John Doe
```

---

10 (Q No. 5B) How is Array implemented in PHP? Differentiate between foreach() loop and each() function with example.

In PHP, an array is implemented as an ordered map, which means it is a collection of key-value pairs where each key corresponds to a specific value. PHP arrays can hold values of any type, including integers, floats, strings, booleans, objects, and even other arrays.

There are two main ways to loop through an array in PHP: using a foreach() loop or the each() function. Here's an example of each:

Example using foreach() loop:

```
$fruits = array('apple', 'banana', 'cherry', 'date');
foreach ($fruits as $fruit) {
echo $fruit . '<br>';
}

=> OUTPUT
    apple
    banana
    cherry
    date
```

In this example, we first create an array called $fruits that contains four elements. Then we use a foreach() loop to iterate through each element in the array and print it to the screen. The loop uses a variable called $fruit to represent each individual element in the array as it loops through.

Example using each() function:

```
$fruits = array('apple', 'banana', 'cherry', 'date');
$fruit = each($fruits);
echo $fruit['key'] . ' = ' . $fruit['value'] . '<br>';
$fruit = each($fruits);
echo $fruit['key'] . ' = ' . $fruit['value'] . '<br>';

=> OUTPUT
0 = apple
1 = banana
```

Here we create an array called $fruits with four elements. Then we use the each() function to get the first element in the array and store it in a variable called $fruit. We then print out the key and value of this element using the $fruit variable and the 'key' and 'value' keys of the array returned by the each() function. We then use the each() function again to get the second element in the array and repeat the process.

---

11 (Q No. 6A) Make a login form with email and password. Write a program to make dynamic login using PHP and MySQL query. Your program must contain database connection file from PHP to MySQL.

Here is the source code for login form with email and password.

```
<!DOCTYPE html>
<html>
<head>
  <title>Login Form</title>
```

```
</head>
<body>
  <h2>Login Form</h2>
  <form action="login.php" method="post">
    <label for="email">Email:</label>
    <input type="email" id="email" name="email" required><br><br>
    <label for="password">Password:</label>
    <input type="password" id="password" name="password" required><br><br>
    <input type="submit" value="Login">
  </form>
</body>
</html>
```

Listing 8: Login Form: index.html

Here is the PHP code to connect to MySQL database and validate the user's email and password.

```
<?php
session_start();

// Database connection
$host = "localhost";
$username = "username";
$password = "password";
$database = "userdb";

$conn = mysqli_connect($host, $username, $password, $database);

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

// Get email and password from the form
$email = $_POST['email'];
$password = $_POST['password'];

// Check if the email and password are valid
$sql = "SELECT * FROM users WHERE email='$email' AND password='$password'";
$result = mysqli_query($conn, $sql);

if (mysqli_num_rows($result) == 1) {
  // Authentication successful
  $_SESSION['email'] = $email;
  header("Location: dashboard.php");
} else {
  // Authentication failed
  echo "Invalid email or password";
}

mysqli_close($conn);
?>
```

Listing 9: Login Form: login.php

Here we start a PHP session and establish a connection to the MySQL database using the mysql_connect() function. We then retrieve the email and password values submitted in the login form using the $_POST superglobal. We check if the email and password are valid by running a MySQL query that selects a row from the users table where the email and password match the values submitted in the form. If the query returns one row, we store the email in the $_SESSION

superglobal and redirect the user to a dashboard page. If the query returns zero rows, we display an error message.

---

12 (Q No. 6B) Describe Starting, Storing and Destroying a Session. Explain with sample code. Also differentiate between Session and Cookies.

Starting a Session, Storing and Destroying a Session in PHP:

Starting a Session: In PHP, a session can be started using the session_start() function. This function must be called at the beginning of every page where you want to use session variables.

Example:

$$session\_start();$$

Storing Session Data: Session data can be stored as key-value pairs in the $_SESSION superglobal array. This array can be accessed and modified like any other PHP array. Example:

```
1    $_SESSION['username'] = 'john_doe';
2    $_SESSION['user_id'] = 123;
```

Destroying a Session: A session can be destroyed using the session_destroy() function. This function removes all session variables and ends the session.

Example:

$$session\_destroy();$$

Session and cookies are two ways to store data between HTTP requests in PHP. Here are some key differences:

1. **Storage Location:** Session data is stored on the server, while cookie data is stored on the client (in the browser).
2. **Data Size:** Cookies have a maximum size limit of around 4 KB, while session data can be much larger.
3. **Security:** Session data is generally considered more secure than cookies, as it is stored on the server and cannot be easily tampered with by the client.
4. **Expiration:** Cookies can have an expiration date, while session data is destroyed when the user closes their browser or logs out.
5. **Accessibility:** Cookies can be accessed by both the client-side scripts and server-side scripts, while session data can only be accessed by server-side scripts.
6. Cookies can be set using the setcookie() function. This function takes several parameters including the cookie name, value, expiration time, and path.

```
1        setcookie('username', 'john_doe', time() + 3600, '/');
2
```

For Retrieving data from cookie we use following code:

```php
        // Check if the 'username' cookie is set and retrieve its value
        if (isset($_COOKIE['username'])) {
        $username = $_COOKIE['username'];
        echo "Welcome back, $username!";
        } else {
        echo "Please log in to continue.";
        }

```

For starting session and setting values in it we use following code:

```php
        // Start the session
        session_start();

        // Check if the 'username' session variable is set and retrieve its value
        if (isset($_SESSION['username'])) {
        $username = $_SESSION['username'];
        echo "Welcome back, $username!";
        } else {
        echo "Please log in to continue.";
        }


```

---

13 (Q No. 7A) DOM Hierarchy in JavaScript

The DOM (Document Object Model) hierarchy in JavaScript refers to the structure of HTML elements and their relationships with each other. The DOM represents an HTML document as a tree-like structure, where each node in the tree represents an element in the document. Here's an example of a simple HTML document and its corresponding DOM hierarchy:

```html
<!DOCTYPE html>
<html>
  <head>
    <title>My Page</title>
  </head>
  <body>
    <h1 id="myHeading">Welcome to my page!</h1>
    <p>This is a paragraph.</p>
    <ul>
      <li>List item 1</li>
      <li>List item 2</li>
      <li>List item 3</li>
    </ul>

    <script>
      // Retrieve the h1 element by its ID
      var heading = document.getElementById('myHeading');

      // Change the text content of the h1 element
      heading.textContent = 'Hello, world!';
    </script>
  </body>
</html>
```

In this example, the root node of the DOM hierarchy is the html element. The head and body elements are child nodes of the html element, and the title, h1, p, and ul elements are child nodes of the head and body elements. The li elements are child nodes of the ul element.

In JavaScript, we can use the DOM API to manipulate the DOM hierarchy. To retrieve the h1 element and change its text content: we use the getElementById() method of the document object to retrieve the h1 element by its ID. We then use the textContent property to change the text content of the h1 element.

---

14 (Q No. 7B) Client Side vs Server Side scripting

Client-side scripting and server-side scripting are two different approaches to building dynamic web pages.

Client-side scripting refers to the process of running scripts on the client's (i.e., the user's) web browser. These scripts are usually written in JavaScript and are used to create interactive and dynamic web pages. Client-side scripts can be used to validate user input, update page content without reloading the page, and perform various other functions that improve the user experience.

Server-side scripting, on the other hand, refers to the process of running scripts on the web server. These scripts are usually written in languages such as PHP, Python, or Ruby, and are used to generate dynamic HTML pages that are then sent to the client's browser. Server-side scripts can be used to process user input, access databases, generate dynamic content, and perform various other functions that are not possible with client-side scripting.

Here are some key differences between client-side scripting and server-side scripting:

1. **Execution:** Client-side scripts are executed by the user's browser, while server-side scripts are executed by the web server.
2. **Speed:** Client-side scripts are generally faster than server-side scripts because they don't require a round-trip to the server for every request.
3. **Security:** Server-side scripts are generally more secure than client-side scripts because they can't be modified or manipulated by the user.
4. **Accessibility:** Client-side scripts may not be accessible to users who have disabled JavaScript in their browsers, while server-side scripts are accessible to all users.

Here's an example of a simple client-side script that changes the color of a button when it's clicked:

```
<html>
  <head>
    <title>Client-Side Scripting Example</title>
    <script>
      function changeColor() {
        document.getElementById('myButton').style.backgroundColor = 'red';
      }
    </script>
  </head>
  <body>
    <button id="myButton" onclick="changeColor()">Click me!</button>
  </body>
</html>
```

Here we define a JavaScript function called changeColor() that changes the background color of a button when it's clicked. We then use the onclick attribute to attach the function to the button. Here's an example of a simple server-side script that generates a dynamic web page based on user input:

```
<html>
  <head>
    <title>Server-Side Scripting Example</title>
  </head>
  <body>
    <form method="post">
      <label for="name">Enter your name:</label>
      <input type="text" name="name" id="name">
      <input type="submit" value="Submit">
    </form>

    <?php
      if ($_SERVER['REQUEST_METHOD'] == 'POST') {
        $name = $_POST['name'];
        echo "Hello, $name!";
      }
    ?>
  </body>
</html>
```

Here using PHP we generate a web page that prompts the user to enter their name. When the user submits the form, the server-side script retrieves the user's input using the $_POST superglobal array and uses it to generate a personalized greeting. The resulting HTML page is sent back to the client's browser for display.

---

15 (Q No. 7C) Ordered list vs Unordered list

Ordered List

An ordered list is a list of items that are numbered in a specific order. The <ol> tag is used to create an ordered list in HTML. Each list item is defined using the <li> tag. Here is an example of an ordered list in HTML:

```
    <ol>
        <li>First item</li>
        <li>Second item</li>
        <li>Third item</li>
    </ol>
```

This will generate a numbered list that looks like this:

1. First item
2. Second item
3. Third item

Unordered List

An unordered list is a list of items that are not numbered and are presented in a bulleted list. The <ul> tag is used to create an unordered list in HTML. Each list item is defined using the <li> tag. Here is an example of an unordered list in HTML:

```
    <ul>
        <li>First item</li>
        <li>Second item</li>
        <li>Third item</li>
    </ul>
```

This will generate a bulleted list that looks like this:

- First item
- Second item
- Third item