1.Differentiate between GET and POST Method.Why is HTTP protocol called stateless protocol? How does it manage its state during item addition in the e-commerce shopping cart?
What do you mean by client side validation and server side validation? Briefly explain with example.
Ans.

Difference between GET and POST Method:

- GET Method:
    - Used to request data from a server.
    - Appends parameters to the URL in the form of a query string.
    - Data is visible in the URL, making it less secure.
    - Typically used for read-only operations like fetching data from a server.
    - Limited data length (URL length constraints).
    - Can be cached, bookmarked, and remain in browser history.
- POST Method:
    - Used to submit data to be processed to a server.
    - Data is sent in the body of the request, making it more secure than GET.
    - Suitable for operations that modify the server's data (e.g., submitting forms, uploading files).
    - No size restrictions on the data being sent.
    - Cannot be cached or bookmarked.

Ans.
HTTP is called a stateless protocol because each request-response cycle is independent. The server does not retain information about previous requests from the same client. Each request is treated as new, without any memory of past interactions.

Ans.
To manage state, even though HTTP is stateless, cookies, sessions, and local storage are used to maintain state information across multiple requests.

- Cookies: Small pieces of data stored in the user's browser. The server can use cookies to track a user's shopping cart across different requests.
- Session: Server-side mechanism where each user is assigned a unique session ID. The shopping cart data is stored in the session and the session ID is passed back and forth via cookies.
- Local Storage: Client-side storage can store the cart items temporarily until the user completes the transaction.

Ans.

Client-Side Validation:

- Validation done in the user's browser before the data is sent to the server.

- Examples: Checking if all required fields are filled, ensuring an email address is in a correct format.

```
<form onsubmit="return validateForm()">
  <input type="text" id="email" required>
  <input type="submit">
</form>
<script>
  function validateForm() {
    var email = document.getElementById("email").value;
    if (email === "") {
      alert("Email is required!");
      return false;
    }
  }
</script>
```

Server-Side Validation:

- Validation done after the data is submitted to the server, ensuring security.
- Examples: Ensuring that user inputs do not contain malicious code (SQL injection prevention), checking for valid user data after submission.

```
import java.io.IOException;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
```

```
import javax.servlet.http.HttpServletResponse;

@WebServlet("/register")
public class RegistrationServlet extends HttpServlet {

    @Override
    protected void doPost(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
        // Retrieve the email from the request
        String email = request.getParameter("email");

        // Example validation: Check if the email is already registered
        if (isEmailRegistered(email)) {
            response.setContentType("text/html");
            response.getWriter().println("Email is already registered.");
        } else {
            // Process registration (e.g., save email to database)
            response.setContentType("text/html");
            response.getWriter().println("Registration successful.");
        }
    }

    // Simulated method for checking if the email is already registered
    private boolean isEmailRegistered(String email) {
        // In a real application, check against a database
```

```
        return email.equals("existing@example.com");
    }
}
```

Both types of validation are crucial: client-side for user experience, server-side for security.

2. Write an HTML document to create a job registration form.

i. Create text boxes for Name, Address, Phone, Email.

ii. Create two radio buttons for Gender specification.

iii.Create a text area for additional information.

iv.Create username and password for future login.

v.Create submit button.

[Write an external CSS file to style a table so that the odd rows have a yellow background and a header row has style white text and a black background color.]

What are the different types of selectors in CSS?

Ans.

index.html

```
<!DOCTYPE html>
<html>
<head>
    <title>Job Registration Form</title>
    <link rel="stylesheet" href="styles.css">
</head>
<body>
```

```
<div class="container">

    <h1>Job Registration Form</h1>

    <table>

        <tr>

            <th>Name</th>

            <td><input type="text" name="name"></td>

        </tr>

        <tr>

            <th>Address</th>

            <td><input type="text" name="address"></td>

        </tr>

        <tr>

            <th>Phone</th>

            <td><input type="text" name="phone"></td>

        </tr>

        <tr>

            <th>Email</th>

            <td><input type="email" name="email"></td>

        </tr>

        <tr>

            <th>Gender</th>

            <td>

                <input type="radio" name="gender" value="male"> Male
```

```
                    <input type="radio" name="gender" value="female">
Female

                </td>

            </tr>

            <tr>

                <th>Additional Information</th>

                <td><textarea name="additional_info"></textarea></td>

            </tr>

            <tr>

                <th>Username</th>

                <td><input type="text" name="username"></td>

            </tr>

            <tr>

                <th>Password</th>

                <td><input type="password" name="password"></td>

            </tr>

            <tr>

                <td colspan="2"><input type="submit" value="Submit"></td>

            </tr>

        </table>

    </div>

</body>

</html>
```

style.css

```
.container {
    width: 100%;
    max-width: 600px; /* Adjust the maximum width as needed */
    margin: 0 auto;
    padding: 20px;
}

table {
    width: 100%;
    border-collapse: collapse;
}

th {
    background-color: black;
    color: white;
    padding: 8px;
    text-align: left;
}

tr:nth-child(odd) {
    background-color: yellow;
}
```

```
td {
    padding: 8px;
}
```

Ans.

CSS selectors are patterns used to select and style HTML elements. Here are the different types of selectors in CSS:

i. Universal Selector (`*`)

- Selects all elements on the page.

- Example:

```
* {
  margin: 0;
  padding: 0;
}
```

ii. Type Selector (Element Selector)

- Selects all elements of a specific type (e.g., `<p>`, `<h1>`, etc.).
- Example:

```
p {
  color: blue;
}
```

iii. Class Selector (`.classname`)

- Selects all elements with a specific class attribute.
- Example:

```
.highlight {
  background-color: yellow;
}
```

iv. ID Selector (`#idname`)

- Selects an element with a specific ID attribute (IDs should be unique on a page).
- Example:

```
#header {
  font-size: 24px;
}
```

v. Attribute Selector

- Selects elements based on an attribute or attribute value.
- Examples:

```
/* Selects all inputs */
input[type="text"] {
  border: 1px solid black;
```

```
}

/* Selects elements with the `title` attribute */
[title] {
  color: green;
}
```

vi. Group Selector

- Groups multiple selectors together to apply the same styles.
- Example:

```
h1, h2, h3 {
  font-family: Arial;
}
```

vii. Descendant Selector (Space)

- Selects elements that are descendants of a specified element.
- Example:

```
div p {
  color: red;
}
```

viii. Child Selector (`>`)

- Selects elements that are direct children of a specified element.
- Example:

```
div > p {
  font-size: 16px;
}
```

ix. Adjacent Sibling Selector (`+`)

- Selects an element that is directly after another specific element.
- Example:

```
h1 + p {
margin-top: 10px;
}
```

x. General Sibling Selector (`~`)

- Selects all elements that are siblings of a specified element.
- Example:

```
h1 ~ p {
color: green;
}
```

xi. Pseudo-Class Selector

- Selects elements based on their state or position (e.g., `:hover`, `:nth-child`).
- Examples:

```
a:hover {
color: red;
}


li:nth-child(odd) {
background-color: lightgray;
}
```

xii. Pseudo-Element Selector

- Selects specific parts of an element, like the first letter or first line.

- Examples:

```
p::first-letter {
    font-size: 2em;
}
p::before {
    content: "Note: ";
    font-weight: bold;
}
```

3.Explain briefly the structure of a client server architecture.Why is Tiered architecture important and how many Tiers can an architecture have?What are the differences between client side scripting and server side scripting? Give an example of client side scripting.

Ans.

Structure of Client-Server Architecture:

In a client-server architecture, there are two main components:

i. Client:

- The client is typically a user's device (e.g., a web browser, mobile app) that sends requests to the server for specific resources or services.

- The client presents data to the user and interacts with the server through the network.

ii. Server:

- The server is a central system that receives and processes the client's requests.

- The server handles data management, processing, and returning responses to the client.

How It Works:

- The client makes a request (e.g., HTTP request) to the server.

- The server processes the request (e.g., fetching data from a database) and sends a response back to the client.

- The client receives the response and displays it to the user.

Ans.

Importance of Tiered Architecture:

Tiered architecture(or multi-tier architecture) is essential for:

i. Separation of Concerns: Different components (like the user interface, business logic, and data storage) are separated into tiers, making it easier to manage, maintain, and scale each part independently.

ii. Scalability: Each tier can be scaled separately. For example, the database can be scaled differently from the web server depending on the load.

iii. Flexibility: New functionality can be added or updated without affecting the entire system, allowing flexibility in development and deployment.

iv. Security: Different tiers can implement security mechanisms to ensure that sensitive data and business logic are protected.

Ans.

Number of Tiers in an Architecture:

An architecture can have n-tiers, but the most common are:

i. 1-Tier: Everything (UI, logic, database) resides on the same machine (not scalable).

ii. 2-Tier: Client and server (business logic and database) are separated.

iii. 3-Tier: Separates the presentation layer (UI), application layer (business logic), and data layer(database).

iv. n-Tier: More than three tiers where different layers handle different services (e.g., API, authentication, caching layers).

Ans.

Differences Between Client-Side and Server-Side Scripting:

Feature	Client-Side Scripting	Server-Side Scripting
Execution Location	Executes in the browser (client)	Executes on the web server
Response Time	Faster response (no need to	Slower due to server

	communicate with server)	communication
Security	Less secure (code is visible to the user)	More secure (code is hidden from the user)
Language Examples	JavaScript, HTML, CSS	PHP, Python (Django/Flask), Node.js, Ruby
Data Access	Limited (cannot access server databases)	Can access databases and server resources
Use Case	Validation, animations, updating UI	Database operations, authentication

Example of Client-Side Scripting:

JavaScript Example for Form Validation:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <title>Client-Side Validation</title>
    <script>
        function validateForm() {
            var email = document.forms["registration"]["email"].value;
            if (email == "") {
                alert("Email must be filled out");
                return false;
            }
        }
    </script>
</head>
```

```
<body>
   <form name="registration" onsubmit="return validateForm()">
      Email: <input type="text" name="email">
      <input type="submit" value="Submit">
   </form>
</body>
</html>
```

In this example, the form validation happens on the client side before the form is submitted to the server.

4. Design a HTML page containing:1.a heading 2. a link. 3. some contents. Draw an outline to show the layout of your page elements. On clicking the link a new page opens in an iframe in the same page. Also, initially the link colour is green. On clicking the link the colour changes to red. Write the stylesheet code corresponding to the above functionality.

Ans.

index.html

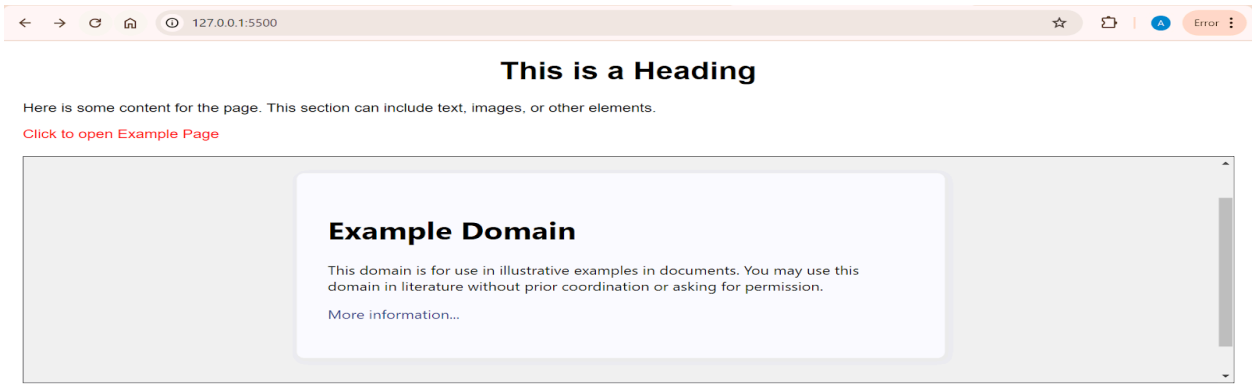
```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Layout with Iframe and Link</title>
    <link rel="stylesheet" href="style.css">
    <script>
        // Function to change link color on click
        function changeLinkColor() {
            document.getElementById("myLink").style.color = "red";
```

```
        }
    </script>
</head>
<body>
    <header>
        <h1>This is a Heading</h1>
    </header>

    <section>
        <p>Here is some content for the page. This section can include text, images, or other elements.</p>

        <!-- Link to open a new page inside iframe -->
        <a id="myLink" href="https://www.example.com" target="iframeArea" onclick="changeLinkColor()">Click to open Example Page</a>
    </section>

    <section>
        <!-- Iframe to load the link -->
        <iframe name="iframeArea" width="100%" height="300px" style="border: 1px solid black;"></iframe>
    </section>
</body>
</html>
```

style.css

```
/* Default link color (green) */
a {
    color: green;
    text-decoration: none;
}

/* On hover, make underline visible */
a:hover {
    text-decoration: underline;
}

/* Styling for the layout */
body {
    font-family: Arial, sans-serif;
    margin: 20px;
}

header {
    text-align: center;
    margin-bottom: 20px;
```

```
}


section {

    margin-bottom: 20px;

}
```



5. Write HTML code for dividing the browser window into two horizontal parts (left part is content and right part is main). Left part should contain the anchors clicking on which corresponding page to be displayed at the right part. Clearly state the assumptions regarding the developed code. Say you have a map of India which is only showing the boundary of each state. Write an HTML page to show that map and when user click within the boundary of West Bengal and

Bihar it will be redirected to the detail map of corresponding state. You may assume arbitrary polygon area in the said boundaries for this purpose.

Ans.

i.html

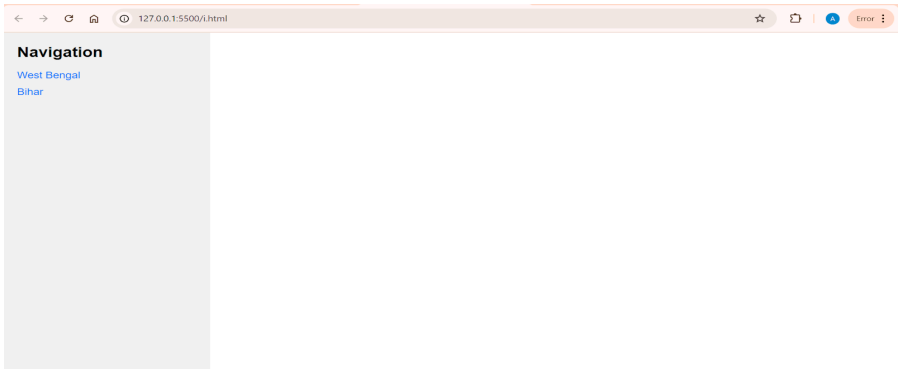
```
<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

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

    <title>India Map with State Details</title>

    <style>

        /* CSS for styling */

        body {

            margin: 0;

            padding: 0;

            font-family: Arial, sans-serif;

        }



        .container {

            display: flex;

            height: 100vh;

        }



        .left-panel {
```

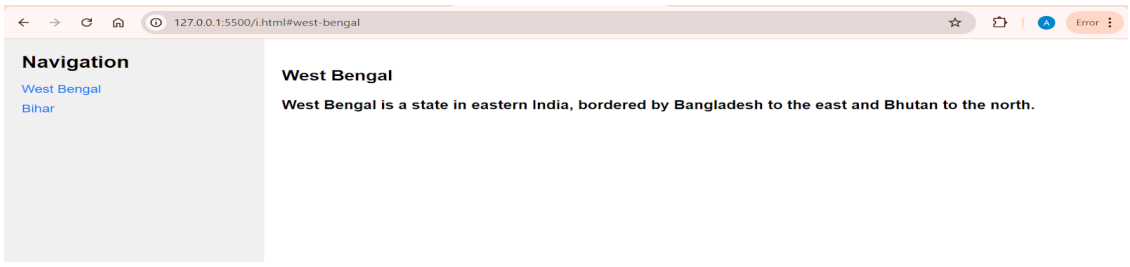
```
        width: 20%; /* Adjust the width as needed */
        background-color: #f0f0f0;
        padding: 20px;
    }

    .right-panel {
        flex: 1;
        padding: 20px;
        background-color: #fff;
    }

    h2 {
        margin: 0 0 10px 0;
        padding: 0;
    }

    ul {
        list-style: none;
        padding: 0;
    }

    li {
        margin: 10px 0;
```

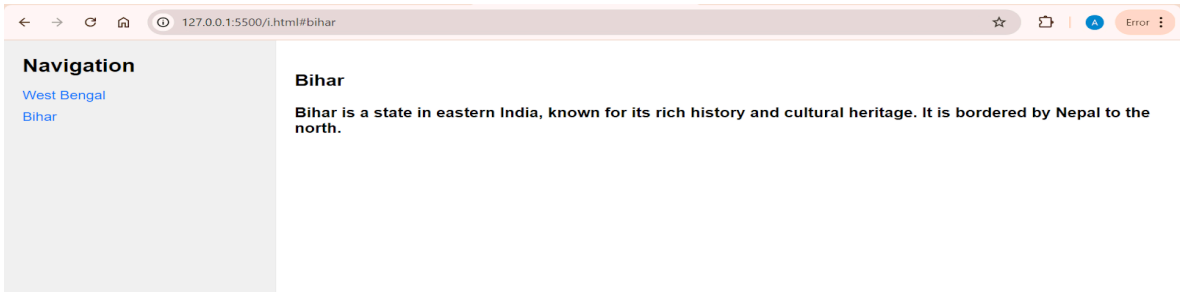
```
        }

        a {
            text-decoration: none;
            color: #007BFF;
        }

        a:hover {
            color: #FF0000;
        }

        #state-details {
            margin-top: 20px;
            font-size: 18px;
            font-weight: bold;
        }
    </style>
</head>
<body>
    <div class="container">
        <div class="left-panel">
            <h2>Navigation</h2>
            <ul>
```

```
                <li><a href="#west-bengal" onclick="showWestBengal()">West
Bengal</a></li>

                <li><a href="#bihar" onclick="showBihar()">Bihar</a></li>

            </ul>

        </div>



        <div class="right-panel">

            <div id="state-details">

            </div>

        </div>

    </div>



    <script>

        // Function to display details for West Bengal

        function showWestBengal() {

            const stateDetails = document.getElementById('state-details');

            stateDetails.innerHTML = `

                <h3>West Bengal</h3>

                <p>West Bengal is a state in eastern India, bordered by
Bangladesh to the east and Bhutan to the north.</p>

            `;

        }



        // Function to display details for Bihar
```

```
        function showBihar() {
            const stateDetails = document.getElementById('state-details');
            stateDetails.innerHTML = `
                <h3>Bihar</h3>
                <p>Bihar is a state in eastern India, known for its rich history and cultural heritage. It is bordered by Nepal to the north.</p>
            `;
        }
    </script>
</body>
</html>
```







Ans.

ii.html

```
<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

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

    <title>India Map with State Details</title>

    <style>

        /* CSS for styling */

        body {

            margin: 0;

            padding: 0;

            font-family: Arial, sans-serif;

        }
```

```
        .container {
            display: flex;
            height: 100vh;
        }

        .left-panel {
            width: 20%; /* Adjust the width as needed */
            background-color: #f0f0f0;
            padding: 20px;
        }

        .right-panel {
            flex: 1;
            padding: 20px;
            background-color: #fff;
        }

        .map-container {
            width: 100%;
            height: auto;
        }

        h2 {
```

```
        margin: 0 0 10px 0;
        padding: 0;
    }

    ul {
        list-style: none;
        padding: 0;
    }

    li {
        margin: 10px 0;
    }

    a {
        text-decoration: none;
        color: #007BFF;
    }

    a:hover {
        color: #FF0000;
    }

    #state-details {
```

```
            margin-top: 20px;
            font-size: 18px;
            font-weight: bold;
        }
    </style>
</head>
<body>
    <div class="container">
        <!-- Left Panel for Navigation -->
        <div class="left-panel">
            <h2>Navigation</h2>
            <ul>
                <li><a href="#west-bengal"
onclick="redirectToWestBengal()">West Bengal</a></li>
                <li><a href="#bihar"
onclick="redirectToBihar()">Bihar</a></li>
            </ul>
        </div>

        <!-- Right Panel with Map and Content -->
        <div class="right-panel">
            <div class="map-container">
                <!-- Map image (replace with the actual image source) -->
```

```
                <img src="india-map.png" alt="India Map" id="india-map"
usemap="#indiaMap" width="500">

                <!-- Define clickable areas on the map -->

                <map name="indiaMap">

                    <area shape="poly"
coords="100,250,350,230,400,280,380,300,330,290,100,250"
href="west-bengal.html" alt="West Bengal">

                    <area shape="poly"
coords="300,250,350,230,400,280,380,300,330,290,300,250" href="bihar.html"
alt="Bihar">

                </map>

            </div>

            <div id="state-details">

                Click on West Bengal or Bihar on the map to view a
detailed map.

            </div>

        </div>

    </div>



    <script>

        // JavaScript functions for redirection when clicking on
navigation links



        function redirectToWestBengal() {

            window.location.href = 'west-bengal.html';

        }
```

```
        function redirectToBihar() {
            window.location.href = 'bihar.html';
        }
    </script>
</body>
</html>
```

west-bengal.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>West Bengal Map</title>
    <style>
        body {
            font-family: Arial, sans-serif;
            text-align: center;
            padding: 20px;
        }
        img {
```

```
            max-width: 100%;
            height: auto;
            border: 2px solid #000;
        }
        .back-link {
            display: inline-block;
            margin-top: 20px;
            padding: 10px 20px;
            background-color: #007BFF;
            color: white;
            text-decoration: none;
            border-radius: 5px;
        }
        .back-link:hover {
            background-color: #0056b3;
        }
    </style>
</head>
<body>
    <h1>Detailed Map of West Bengal</h1>

    <!-- Replace with actual detailed map of West Bengal -->
    <img src="west_bengal_detail.png" alt="Detailed Map of West Bengal">
```

```
    <p>This is the detailed map of West Bengal showing major cities,
rivers, and landmarks.</p>



    <a href="ii.html" class="back-link">Back to India Map</a>

</body>

</html>
```

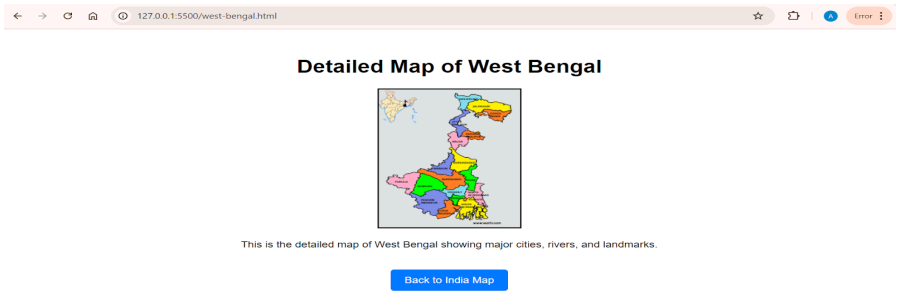
bihar.html

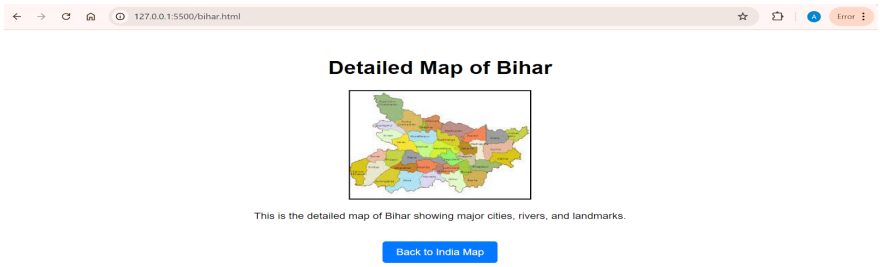
```
<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

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

    <title>Bihar Map</title>

    <style>

        body {

            font-family: Arial, sans-serif;

            text-align: center;

            padding: 20px;

        }

        img {

            max-width: 100%;
```

```
            height: auto;
            border: 2px solid #000;
        }
        .back-link {
            display: inline-block;
            margin-top: 20px;
            padding: 10px 20px;
            background-color: #007BFF;
            color: white;
            text-decoration: none;
            border-radius: 5px;
        }
        .back-link:hover {
            background-color: #0056b3;
        }
    </style>
</head>
<body>
    <h1>Detailed Map of Bihar</h1>

    <!-- Replace with actual detailed map of Bihar -->
    <img src="bihar_detail.png" alt="Detailed Map of Bihar">
```

```
    <p>This is the detailed map of Bihar showing major cities, rivers, and landmarks.</p>



    <a href="ii.html" class="back-link">Back to India Map</a>

</body>

</html>
```

Click on West Bengal or Bihar on the map to view a detailed map.





6. What is the class attribute in HTML? What is inline tag?

Develop a webpage using html that has three boxes and three labels. The captions of the labels are "First Number", "Second Number" and "Result". Put the text boxes beside the labels. Place five command buttons, namely, "Plus", "Minus", "Multiplication", "Division", "Cancel". The form should disappear when the cancel button is pressed. When the plus button is pressed, it should add the first number and the second number and the result should be displayed in the appropriate text box. The first number and the second number text boxes should allow only 0 – 9 and decimal.[Explain Box selector with an example.]

Ans.

Class Attribute in HTML:

The class attribute in HTML is used to assign a specific name (or multiple names) to an element, which can then be used for styling with CSS or to manipulate the element with JavaScript. The class attribute is reusable, meaning you can assign the same class name to multiple elements.

<!DOCTYPE html>

<html lang="en">

<head>

  <style>

```
        .highlight {
            background-color: yellow;
        }
    </style>
</head>
<body>
    <p class="highlight">This paragraph is highlighted.</p>
    <p>This paragraph is not highlighted.</p>
    <div class="highlight">This div is also highlighted.</div>
</body>
</html>
```

In this example, both the paragraph and the div with the class `highlight` will have a yellow background.

Inline Tag in HTML:

An inline tag (or inline element) is an HTML element that does not start on a new line and only takes up as much width as necessary. Inline elements are usually used to format parts of text inside block elements without affecting the overall layout. Inline elements can contain text and other inline elements.

Examples of Inline Tags:

- `<span>`
- `<a>`
- `<strong>`
- `<em>`
- `<img>`

`Example:`

```
<p>This is a <strong>bold</strong> text inside a paragraph.</p>
```

In this example, the `<strong>` tag is an inline element, and it bolds only the part of the text inside the paragraph without breaking the flow or starting a new line.

Key Differences:

- Class Attribute: Used to style or manipulate multiple elements with the same properties.
- Inline Tag: Used within a line of text, and it does not affect the layout by adding extra space or starting on a new line.

Ans.

index.html

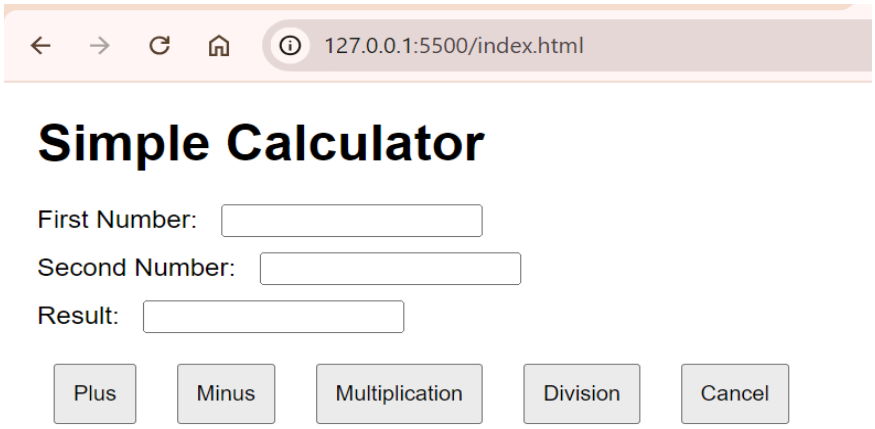
```
<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

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

    <title>Calculator Webpage</title>

    <style>

        body {

            font-family: Arial, sans-serif;

            margin: 20px;

        }

        label {

            margin-right: 10px;

        }

        input[type="text"] {

            width: 150px;

            margin-bottom: 10px;
```

```
        }
        button {
            margin: 10px;
            padding: 10px;
        }
        form {
            display: block;
        }
    </style>
    <script>
        // Function to allow only numeric input and decimal in the
textboxes
        function allowOnlyNumbers(event) {
            const charCode = event.which ? event.which : event.keyCode;
            // Allow numbers (0-9) and decimal point (.)
            if ((charCode < 48 || charCode > 57) && charCode !== 46) {
                event.preventDefault();
            }
        }


        // Function to perform the addition, subtraction, multiplication,
and division
        function calculate(operation) {
```

```
            const num1 =
parseFloat(document.getElementById('firstNumber').value);

            const num2 =
parseFloat(document.getElementById('secondNumber').value);

            let result = 0;



            if (isNaN(num1) || isNaN(num2)) {

                alert("Please enter valid numbers in both fields.");

                return;

            }



            switch (operation) {

                case 'plus':

                    result = num1 + num2;

                    break;

                case 'minus':

                    result = num1 - num2;

                    break;

                case 'multiply':

                    result = num1 * num2;

                    break;

                case 'divide':

                    if (num2 === 0) {

                        alert("Cannot divide by zero!");
```

```
                        return;
                    }
                    result = num1 / num2;
                    break;
            }

            document.getElementById('result').value = result;
        }

        // Function to cancel (hide) the form
        function cancelForm() {
            document.getElementById('calculatorForm').style.display =
'none';
        }
    </script>
</head>
<body>
    <h1>Simple Calculator</h1>

    <!-- Calculator Form -->
    <form id="calculatorForm">
        <!-- First Number Input -->
        <label for="firstNumber">First Number:</label>
```

```
        <input type="text" id="firstNumber"
onkeypress="allowOnlyNumbers(event)"><br>



        <!-- Second Number Input -->

        <label for="secondNumber">Second Number:</label>

        <input type="text" id="secondNumber"
onkeypress="allowOnlyNumbers(event)"><br>



        <!-- Result Box -->

        <label for="result">Result:</label>

        <input type="text" id="result" readonly><br>



        <!-- Buttons -->

        <button type="button" onclick="calculate('plus')">Plus</button>

        <button type="button" onclick="calculate('minus')">Minus</button>

        <button type="button"
onclick="calculate('multiply')">Multiplication</button>

        <button type="button"
onclick="calculate('divide')">Division</button>

        <button type="button" onclick="cancelForm()">Cancel</button>

    </form>

</body>

</html>
```



Ans.

Box Selector in CSS:

The box selector in CSS refers to the layout model that treats every HTML element as a rectangular box. The CSS box model governs the space occupied by these elements and controls the positioning, size, and spacing. It consists of the following components:

1. Content: The actual content of the element (text, images, etc.).
2. Padding: Space between the content and the border.
3. Border: The boundary surrounding the padding and content.
4. Margin: Space outside the border, separating the element from others.

Each part can be controlled individually through CSS properties, allowing developers to adjust the layout and spacing.

Example of CSS Box Model:

```
<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

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

    <title>Box Model Example</title>

    <style>

        .box {
```

```
            width: 200px;
            height: 100px;
            padding: 20px;
            border: 5px solid black;
            margin: 10px;
            background-color: lightblue;
        }
    </style>
</head>
<body>
    <div class="box">
        This is a box with padding, border, and margin.
    </div>
</body>
</html>
```

7. Explain the use of <map> element with a suitable example. What is the function of usemap attribute in <img>element? Briefly explain META tags with example.

Ans.

`<map>` Element in HTML:

The `<map>` element in HTML is used to define an image map, which allows you to create clickable areas (hotspots) on an image. These areas can be associated with links, so when a user clicks on a specific part of the image, they are redirected to another page or action. The `<map>` element contains one or more `<area>` elements, which define the clickable areas on the image.

Function of `usemap` Attribute in `<img>` Element:

The `usemap` attribute in the `<img>` element is used to associate the image with a defined image map. This attribute refers to the `name` or `id` of the `<map>` element. It specifies that the image should be treated as a part of the image map.

Example of `<map>` and `usemap` Attribute:

HTML Example:

```
<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

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

    <title>Image Map Example</title>

</head>

<body>



    <h2>Clickable Map of India</h2>



    <!-- Image with usemap attribute -->

    <img src="india-map.png" alt="Map of India" usemap="#indiaMap" width="500" height="600">



    <!-- Defining the image map -->

    <map name="indiaMap">

        <!-- Defining clickable areas within the map -->

        <area shape="poly" coords="100,150,200,250,300,150" alt="West Bengal"
href="west-bengal.html">

        <area shape="poly" coords="150,300,250,400,350,300" alt="Bihar" href="bihar.html">

    </map>
```

</body>

</html>

Ans.

META Tags in HTML:

The `<meta>` tag provides metadata (data about data) about the HTML document. Metadata will not be displayed on the page, but it plays an important role in SEO, browser behavior, and data handling by search engines. Meta tags are usually placed inside the `<head>` section of the HTML document.

Common Meta Tags:

1. Charset: Defines the character encoding of the document.

2. Viewport: Helps control the layout on mobile devices.

3. Description: Provides a summary of the webpage's content, useful for search engine rankings.

4. Keywords: A list of relevant keywords for the webpage.

5. Author: Defines the name of the author of the webpage.

Example of Meta Tags:

```
<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

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

    <meta name="description" content="A detailed map of India with clickable areas for different states.">

    <meta name="keywords" content="India, Map, States, Geography, Interactive Map">

    <meta name="author" content="John Doe">

    <title>Interactive Map of India</title>
```

```
</head>
<body>
    <h1>Welcome to the Interactive Map of India</h1>
</body>
</html>
```