COMMITTED TO EXCELLENCE IN EDUCATION

निर्मलस्नेह उत्तम सेवाधर्म

SHRI G.P.M. DEGREE COLLEGE OF SCIENCE & COMMERCE

SHRI G.P.M. DEGREE COLLEGE OF SCIENCE & COMMERCE.

(COMMITED TO EXCELLENCE IN EDUCATION)

CERTIFICATE

This is to certify that Mr/Ms.____

a student of **SY.BSC-CS** Roll no: ________has completed the required number of practicals in the subject of ____

as prescribed by the UNIVERSITY OF MUMBAI under my supervision during the academic year 2024-2025.

…………………..	………………………
Prof. Incharge	Principal
………………………	………………………
External Examiner	Course Co-ordinator
………………………..	…………………………
Date	College Stamp

Prof. name : Ms. Akanksha Mishra		Class/SEM : S.Y.BSC.CS / SEM III (2024-2025)		
Course code : USCSP306		**Subject Name : Web Technologies**		
Sr. No	**Date**	**Topics**	**Page No.**	**Signature**
1.	17/06/24	Design a webpage that makes use of a. Document Structure Tags b. Various Text Formatting Tags c. List Tags d. Image and Image Maps		
2.	24/06/24	Design a webpage that makes use of a. Table tags b. Form Tags (forms with various form elements) c. Navigation across multiple pages d. Embedded Multimedia elements		
3.	01/07/24	Design a webpage that make use of Cascading Style Sheets with a. CSS properties to change the background of a Page b. CSS properties to change Fonts and Text Styles c. CSS properties for positioning an element		
4.	22/07/24	Write JavaScript code for a. Performing various mathematical operations such as calculating factorial b. Validating the various Form Elements		
5.	29/07/24	Write JavaScript code for a. Demonstrating different JavaScript Objects such as String, RegExp, Math, Date b. Demonstrating different JavaScript Objects such as Window, Navigator, History, Location, Document c. Storing and Retrieving Cookies		
6.	05/08/24	Create a XML file with Internal / External DTD and display it using a. CSS b. XSL		
7.	12/08/24	Design a webpage to handle asynchronous requests using AJAX on a. Mouseover b. Button click		
8.	26/08/24	Write PHP scripts for a. Retrieving data from HTML forms b. Performing certain mathematical operations such as finding Fibonacci Series c. Working with Arrays d. Working with Files (Reading / Writing)		
9.	02/09/24	Write PHP scripts for a. Working with Databases (Storing Records / Retrieving Records and Display them) b. Storing and Retrieving Cookies c. Storing and Retrieving Sessions		
10.	16/09/24	Design a webpage with some jQuery animation effects.		

Experiment No.—1: HTML basics: Structure, Formatting, Lists, and Multimedia

Aim: Design a webpage that makes use of

a. Document Structure Tags
b. Various Text Formatting Tags
c. List Tags
d. Image and Image Maps

Tools & Technologies used: VS Code, HTML

Learning Objectives:

1. Create structured HTML documents using key tags
2. Create Interactive Elements with Images and Maps

Theory:

1. Document Structure Tags: It define the overall layout and hierarchy of a webpage.

Tags are:

i. <html>: The root element that wraps the entire HTML document.
ii. <head>: Contains meta-information about the document, such as the title and linked stylesheets.
iii. <title>: Specifies the title displayed in the browser tab.
iv. <body>: Contains the content of the webpage that users see.

2. Various Text Formatting Tags : used to enhance the visual presentation of text.

Tags are:

i. <h1>, <h2>, ... <h6>: Header tags that define headings of different levels.
ii. <p>: Defines a paragraph of text.
iii. <strong>: Emphasizes text, typically rendered in bold.
iv. <em>: Italicizes text to denote emphasis.

3. List Tags

List tags organize information in a clear and structured manner.

Types:

i. Unordered Lists (<ul>): Items are displayed with bullet points.
ii. Ordered Lists (<ol>): Items are numbered, indicating a specific sequence.

4. Image and Image Maps

Images enhance the visual appeal of a webpage.

The <img> tag is used to embed images:

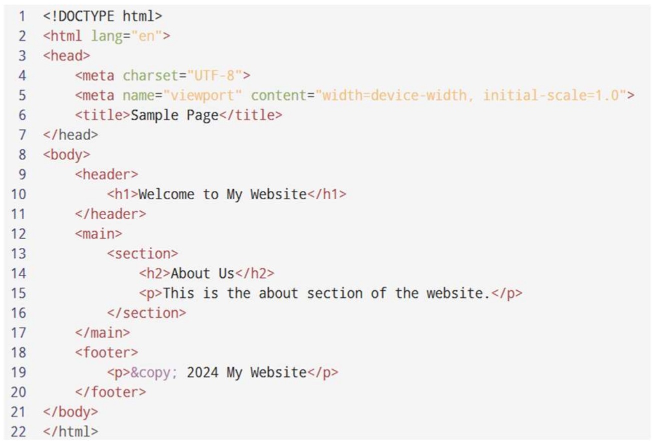
Attributes: src (source), alt (alternative text), width, and height

Image maps allow users to click on specific areas of an image to navigate to different links. This is achieved using:

<map>: Defines the image map.

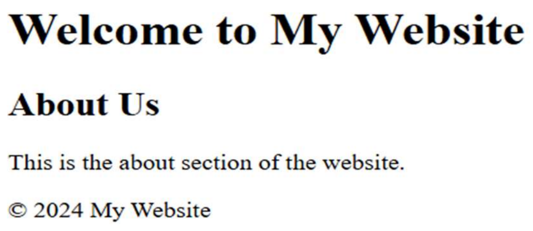
<area>: Specifies clickable areas within the map.

Code1: Document Structure Tags



```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Sample Page</title>
</head>
<body>
    <header>
        <h1>Welcome to My Website</h1>
    </header>
    <main>
        <section>
            <h2>About Us</h2>
            <p>This is the about section of the website.</p>
        </section>
    </main>
    <footer>
        <p>&copy; 2024 My Website</p>
    </footer>
</body>
</html>
```

Output :



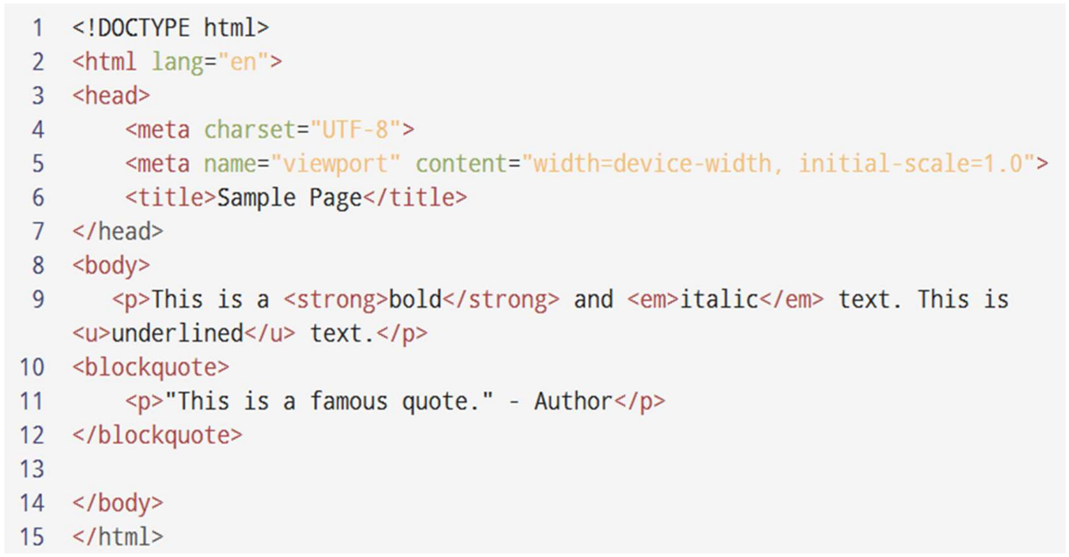
Welcome to My Website

About Us

This is the about section of the website.

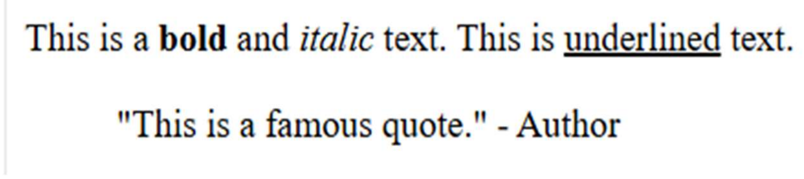
© 2024 My Website

Code2: Various Text Formatting Tags



```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Sample Page</title>
</head>
<body>
   <p>This is a <strong>bold</strong> and <em>italic</em> text. This is <u>underlined</u> text.</p>
<blockquote>
    <p>"This is a famous quote." - Author</p>
</blockquote>

</body>
</html>
```

Output 2 :



This is a **bold** and *italic* text. This is underlined text.

"This is a famous quote." - Author

Code3: List Tags

```html
<!DOCTYPE html>
<html lang="en">
<body>
  <ul>
    <li>Apple</li>
    <li>Banana</li>
    <li>Cherry</li>
</ul>
<ol>
    <li>First item</li>
    <li>Second item</li>
    <li>Third item</li>
</ol>
</body>
</html>
```

Output 3 :

```
• Apple
• Banana
• Cherry

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

Code4: Image and Image Maps

```html
<!DOCTYPE html>
<html lang="en">
<body>
  <img src="worldmap.jpg" usemap="#map1" alt="World Map">
<map name="map1">
    <area shape="rect" coords="34,44,270,350" alt="Europe" href="europe.html">
    <area shape="circle" coords="300,300,50" alt="Asia" href="asia.html">
</map>

</body>
</html>
```

Output 4 :

RUSSIA
ATLANTIC
OCEAN
PACIFIC
OCEAN
INDIAN OCEAN

Learning Outcomes:

1. Mastered HTML document structure and formatting.
2. Created lists and interactive image maps.

Course Outcomes:

1. Develop Comprehensive Web Pages Using HTML
2. Demonstrate Proficiency in Web Design

Conclusion:

Viva Questions:

1. How do you create an ordered list in HTML?
2. What attributes are required for the `<img>` tag?
3. What is the function of the `<map>` tag?
4. What are the different text formatting tags?

For Faculty use:

Correction Parameters	Formative Assessment[40%]	Timely Completion of Practical[40%]	Attendance Learning Attitude[20%]

Experiment No.—2: HTML : Tables, Forms, Navigation, and Multimedia

Aim: Design a webpage that makes use of

- **a.** Table tags
- **b.** Form Tags (forms with various form elements)
- **c.** Navigation across multiple pages
- **d.** Embedded Multimedia elements

Tools & Technologies used: VS Code, HTML

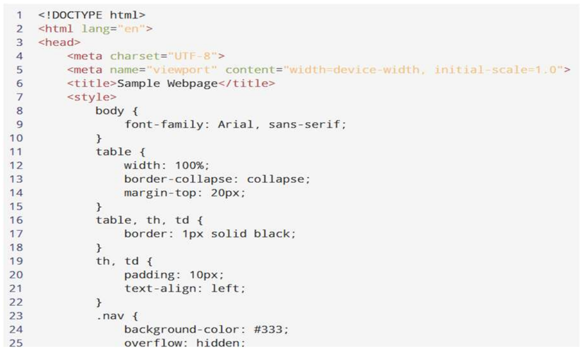
Learning Objectives:

1. Learn to design interactive forms.
2. Master table structures and navigation.

Theory:

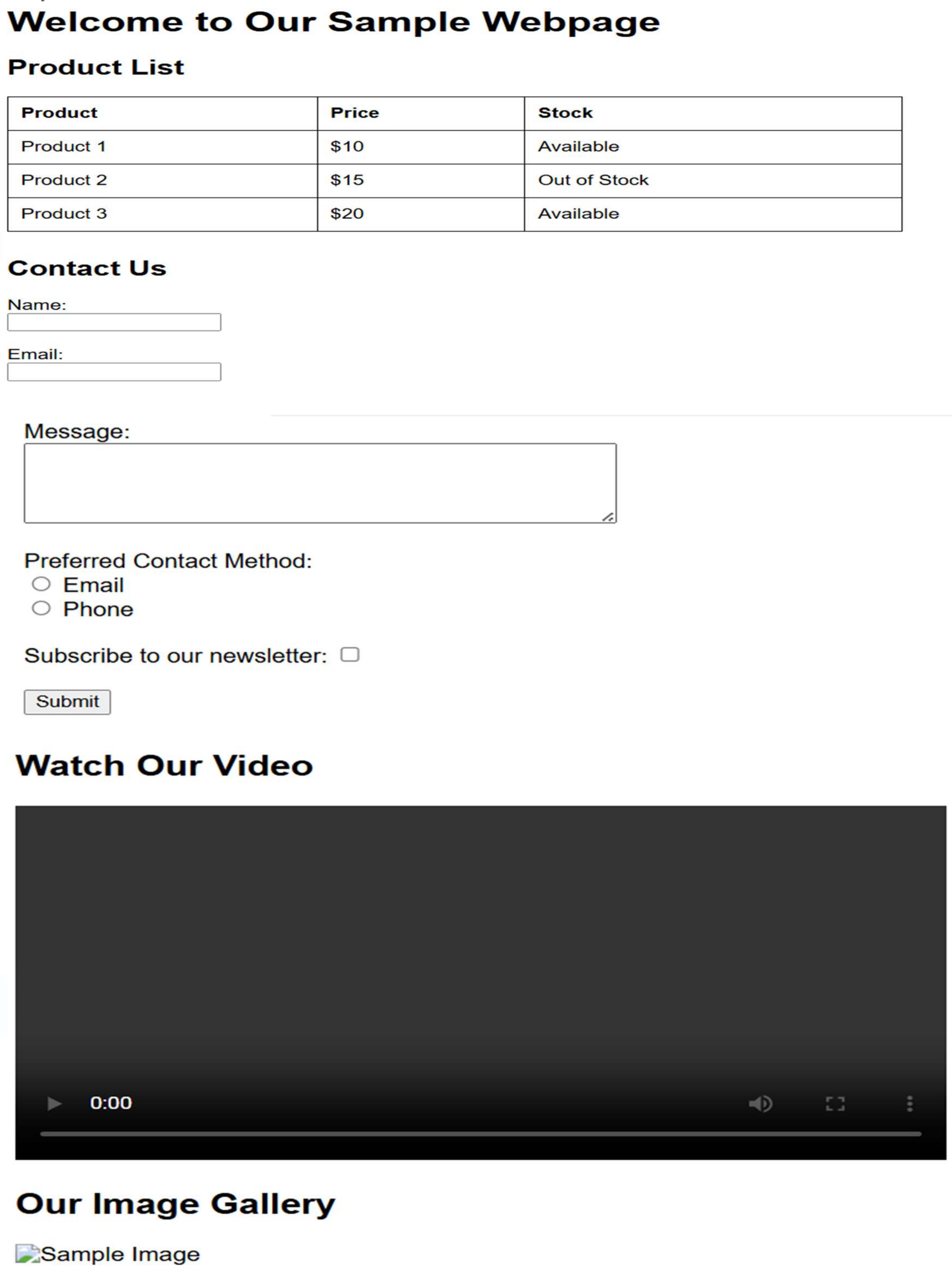
1. Table Tags
   - The <table> tag is used to create tables with rows and columns.
   - Elements like <tr>, <th>, and <td> define table rows, headers, and data.
2. Form Tags
   - The <form> tag is used to collect user input through elements like <input>, <textarea>, and <button>.
   - Forms send data using the action and method attributes.
3. Navigation Across Multiple Pages
   - Navigation is created using the <a> (anchor) tag with the href attribute linking to other pages or sections.
   - It enables internal and external page linking.
4. Embedded Multimedia Elements
   - Multimedia such as images, videos, and audio is embedded using <img>, <video>, and <audio> tags.
   - These elements enrich the content with visuals and media.

Code1:



```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Sample Webpage</title>
    <style>
        body {
            font-family: Arial, sans-serif;
        }
        table {
            width: 100%;
            border-collapse: collapse;
            margin-top: 20px;
        }
        table, th, td {
            border: 1px solid black;
        }
        th, td {
            padding: 10px;
            text-align: left;
        }
        .nav {
            background-color: #333;
            overflow: hidden;
        }
        .nav a {
            color: white;
            padding: 14px 20px;
            text-decoration: none;
            float: left;
        }
        .nav a:hover {
            background-color: #ddd;
            color: black;
        }
        .container {
            padding: 20px;
        }
        .form-container {
            margin-top: 30px;
        }
        .multimedia-container {
            margin-top: 30px;
        }
    </style>
</head>
<body>

    <!-- Navigation Bar -->
    <div class="nav">
        <a href="index.html">Home</a>
        <a href="about.html">About</a>
        <a href="contact.html">Contact</a>
    </div>

    <div class="container">
        <h1>Welcome to Our Sample Webpage</h1>

        <!-- Table -->
        <h2>Product List</h2>
        <table>
            <tr>
                <th>Product</th>
                <th>Price</th>
                <th>Stock</th>
            </tr>
            <tr>
                <td>Product 1</td>
                <td>$10</td>
                <td>Available</td>
            </tr>
            <tr>
                <td>Product 2</td>
                <td>$15</td>
                <td>Out of Stock</td>
            </tr>
            <tr>
                <td>Product 3</td>
                <td>$20</td>
                <td>Available</td>
            </tr>
        </table>

        <!-- Form -->
        <div class="form-container">
            <h2>Contact Us</h2>
            <form action="submit_form.php" method="POST">
                <label for="name">Name:</label><br>
                <input type="text" id="name" name="name" required><br><br>

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

                <label for="message">Message:</label><br>
                <textarea id="message" name="message" rows="4" cols="50" required></textarea><br><br>

                <label>Preferred Contact Method:</label><br>
                <input type="radio" id="email_contact" name="contact_method" value="email">
                <label for="email_contact">Email</label><br>
                <input type="radio" id="phone_contact" name="contact_method" value="phone">
                <label for="phone_contact">Phone</label><br><br>

                <label for="subscribe">Subscribe to our newsletter:</label>
                <input type="checkbox" id="subscribe" name="subscribe" value="yes"><br><br>

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

        <!-- Embedded Multimedia -->
        <div class="multimedia-container">
            <h2>Watch Our Video</h2>
            <video width="600" controls>
                <source src="sample_video.mp4" type="video/mp4">
                Your browser does not support the video tag.
            </video>

            <h2>Our Image Gallery</h2>
            <img src="sample_image.jpg" alt="Sample Image" width="500">
        </div>
    </div>

</body>
</html>
```

Output 1 :



Welcome to Our Sample Webpage

Product List

Product	Price	Stock
Product 1	$10	Available
Product 2	$15	Out of Stock
Product 3	$20	Available

Contact Us

Name:

Email:

Message:

Preferred Contact Method:
○ Email
○ Phone

Subscribe to our newsletter: ☐

Submit

Watch Our Video

Our Image Gallery

Sample Image

Learning Outcomes:

1. Understand table and form creation
2. Implement navigation and multimedia embedding

Course Outcomes:

1. Build web pages with tables/forms
2. Integrate multimedia and multi-page navigation

Conclusion:

Viva Questions:

1. What tag is used to create a table in HTML?
2. Which tag is used to collect user input in HTML?
3. What attribute is used to link pages in HTML?

For Faculty use:

Correction Parameters	Formative Assessment[40%]	Timely Completion of Practical[40%]	Attendance Learning Attitude[20%]

Experiment No.—3: CSS

Aim: Design a webpage that make use of Cascading Style Sheets with

- **a.** CSS properties to change the background of a Page
- **b.** CSS properties to change Fonts and Text Styles
- **c.** CSS properties for positioning an element

Tools & Technologies used: VS Code, HTML, CSS, Chrome

Learning Objectives:

1. Learn to modify page backgrounds
2. Master text styles and element positioning

Theory:

CSS Properties to Change the Background of a Page:

- The background-color property is used to change the background color of an element or page.
- The background-image property allows you to set an image as the background of a page or element.
- The background-size property allows you to control the size of the background image (e.g., cover or contain).
- The background-position property defines the starting position of the background image (e.g., top, center, bottom).

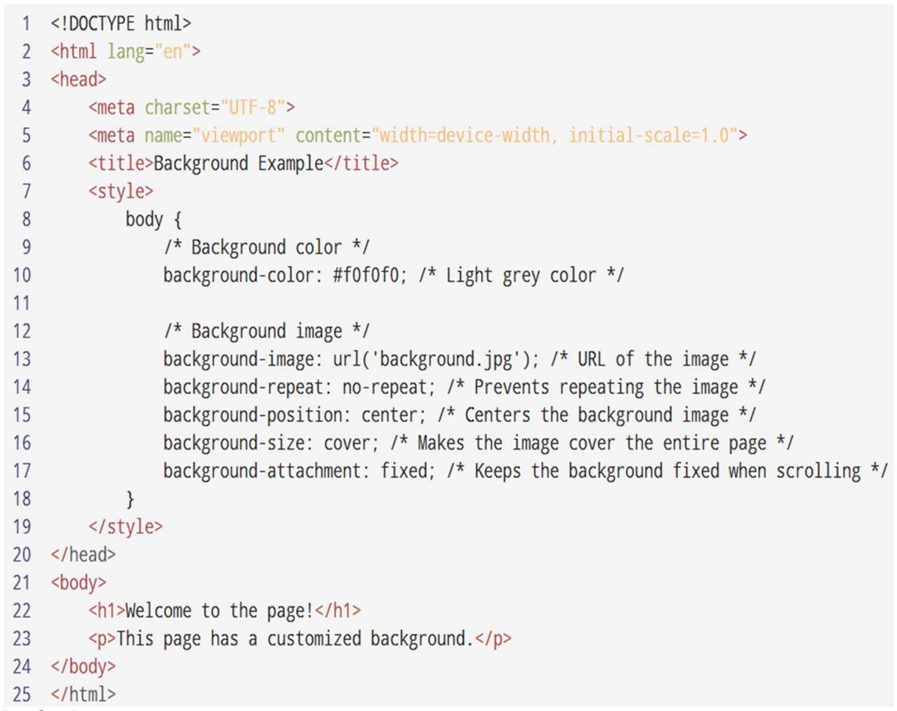
CSS Properties to Change Fonts and Text Styles:

- The font-family property specifies the font style for the text (e.g., Arial, Times New Roman).
- The font-size property controls the size of the text.
- The font-weight property adjusts the thickness of the text (e.g., normal, bold).
- The text-align property defines the alignment of text within an element (e.g., left, center, right).

CSS Properties for Positioning an Element:

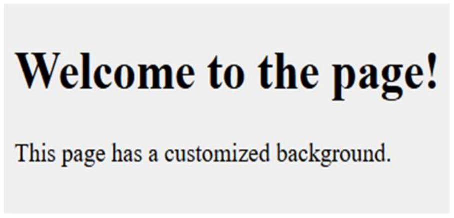
- The position property defines how an element is positioned in the document flow (e.g., static, relative, absolute, fixed, sticky).
- The top, right, bottom, and left properties are used to offset elements when the position is set to relative, absolute, or fixed.
- The z-index property determines the stack order of positioned elements (higher values are in front).
- The display property can be used to control the layout behavior of elements (e.g., block, inline, flex).

Code1:



```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Background Example</title>
    <style>
        body {
            /* Background color */
            background-color: #f0f0f0; /* Light grey color */

            /* Background image */
            background-image: url('background.jpg'); /* URL of the image */
            background-repeat: no-repeat; /* Prevents repeating the image */
            background-position: center; /* Centers the background image */
            background-size: cover; /* Makes the image cover the entire page */
            background-attachment: fixed; /* Keeps the background fixed when scrolling */
        }
    </style>
</head>
<body>
    <h1>Welcome to the page!</h1>
    <p>This page has a customized background.</p>
</body>
</html>
```

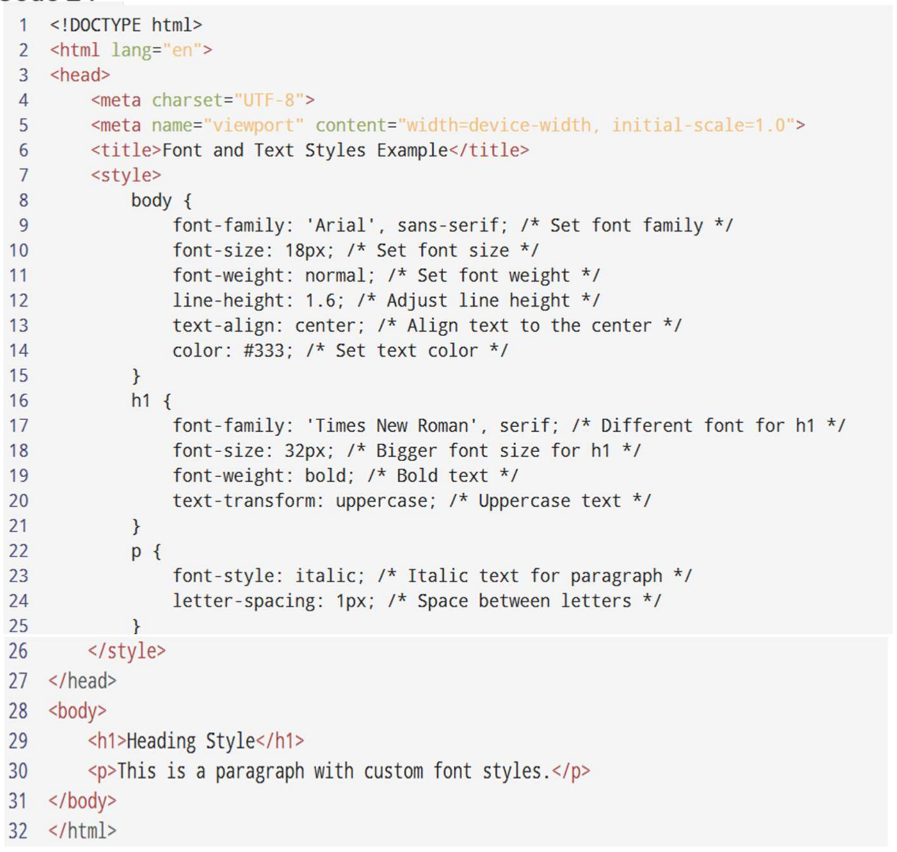
Output 1 :



Welcome to the page!

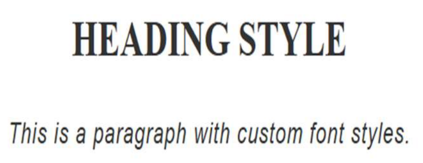
This page has a customized background.

Code 2 :



```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Font and Text Styles Example</title>
    <style>
        body {
            font-family: 'Arial', sans-serif; /* Set font family */
            font-size: 18px; /* Set font size */
            font-weight: normal; /* Set font weight */
            line-height: 1.6; /* Adjust line height */
            text-align: center; /* Align text to the center */
            color: #333; /* Set text color */
        }
        h1 {
            font-family: 'Times New Roman', serif; /* Different font for h1 */
            font-size: 32px; /* Bigger font size for h1 */
            font-weight: bold; /* Bold text */
            text-transform: uppercase; /* Uppercase text */
        }
        p {
            font-style: italic; /* Italic text for paragraph */
            letter-spacing: 1px; /* Space between letters */
        }
    </style>
</head>
<body>
    <h1>Heading Style</h1>
    <p>This is a paragraph with custom font styles.</p>
</body>
</html>
```

Output 2 :



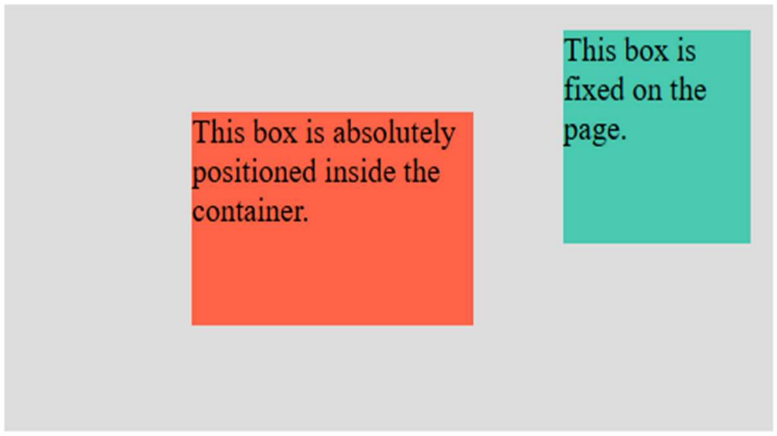
HEADING STYLE

This is a paragraph with custom font styles.

Code 3 :

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Positioning Example</title>
    <style>
        /* Positioning a div element */
        .container {
            position: relative; /* Relative positioning */
            width: 100%;
            height: 200px;
            background-color: #ddd;
        }

        .box {
            position: absolute; /* Absolute positioning */
            top: 50px; /* 50px from the top of the container */
            left: 100px; /* 100px from the left of the container */
            width: 150px;
            height: 100px;
            background-color: #ff6347; /* Tomato color */
            z-index: 10; /* Ensure it's on top of other elements */
        }

        .floating {
            position: fixed; /* Fixed position relative to the viewport */
            top: 20px; /* 20px from the top of the page */
            right: 20px; /* 20px from the right edge */
            width: 100px;
            height: 100px;
            background-color: #48c9b0; /* Turquoise color */
        }
    </style>
</head>
<body>
    <div class="container">
        <div class="box">
            This box is absolutely positioned inside the container.
        </div>
    </div>

    <div class="floating">
        This box is fixed on the page.
    </div>
</body>
</html>
```

Output 3 :


This box is fixed on the page.
This box is absolutely positioned inside the container.

Learning Outcomes:

1. Understand CSS styling and layout techniques
2. Apply CSS properties for web design

Course Outcomes:

1. Design web pages with CSS
2. Control layouts and text styling

Conclusion:

Viva Questions:

1. Which CSS property changes the background?
2. Which property changes text font?
3. Which property controls element positioning?

For Faculty use:

Correction Parameters	Formative Assessment[40%]	Timely Completion of Practical[40%]	Attendance Learning Attitude[20%]

Experiment No.—4: JavaScript

Aim: Write JavaScript code for

- **a.** Performing various mathematical operations such as calculating factorial
- **b.** Validating the various Form Elements

Tools & Technologies used: VS Code, HTML, JavaScript, Chrome

Learning Objectives:

1. Learn to perform mathematical operations
2. Master form validation techniques in JavaScript

Theory:

Performing Mathematical Operations in JavaScript:

- JavaScript offers a variety of built-in mathematical operations like addition, subtraction, multiplication, and division using operators (+, -, *, /).
- JavaScript also provides the Math object which includes methods for power, square root, rounding, and other mathematical functions.
- Custom functions in JavaScript are used to perform complex calculations, such as computing the factorial of a number using recursion or loops.

Form Validation in JavaScript:

- JavaScript is widely used for validating form data before it is submitted to a server. This ensures that users provide the required information in the correct format.
- Validation can include checking for empty fields, ensuring proper email format, validating numeric input, and ensuring that all required checkboxes are checked.
- JavaScript provides methods like getElementById(), value, and submit() to access form elements and validate them using conditional statements.

Common Form Validation Techniques:

- Empty Field Validation: Ensures that mandatory fields are filled out before submission.
- Email Validation: Ensures that the email follows a correct format (e.g., user@example.com).
- Numeric Validation: Ensures that numerical fields contain only numbers within a specified range.

Code1:

```javascript
// Function to calculate factorial
function factorial(n) {
    // Factorial of a negative number is not defined
    if (n < 0) {
        return "Factorial is not defined for negative numbers";
    }

    let result = 1;  // Initializing result to 1, as factorial of 0 is 1
    for (let i = 1; i <= n; i++) {
        result *= i;  // Multiply result by i
    }
    return result;  // Return the calculated factorial
}
```

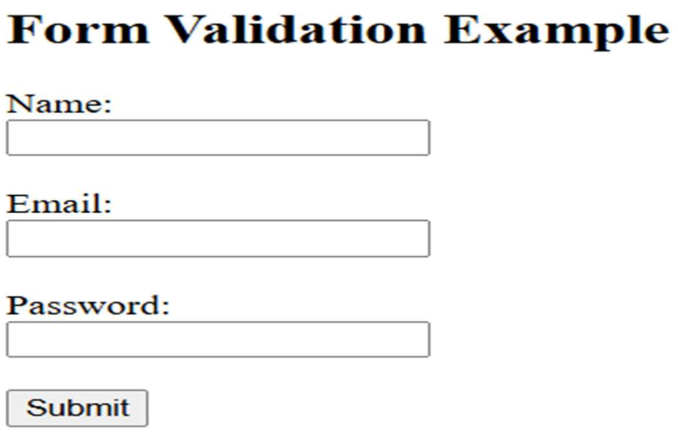
Output 1 :

```
// Example usage
console.log(factorial(5));  // Output: 120
console.log(factorial(0));  // Output: 1
console.log(factorial(-3)); // Output:
Factorial is not defined for negative numbers
```

Code 2 :

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Form Validation</title>
    <script type="text/javascript">
        // Function to validate the form
        function validateForm() {
            // Get the form elements
            var name = document.getElementById('name').value;
            var email = document.getElementById('email').value;
            var password = document.getElementById('password').value;
            var errorMessage = '';

            // Validate Name
            if (name === '') {
                errorMessage += 'Name is required.\n';
            }

            // Validate Email using Regular Expression
            var emailPattern = /^[a-zA-Z0-9._-]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,4}$/;
            if (!email.match(emailPattern)) {
                errorMessage += 'Please enter a valid email.\n';
            }
            // Validate Password (Minimum 8 characters)
            if (password.length < 8) {
                errorMessage += 'Password must be at least 8 characters long.\n';
            }

            // Show error message and prevent form submission if there are errors
            if (errorMessage !== '') {
                alert('Error(s):\n' + errorMessage);
                return false; // Prevent form submission
            }

            return true; // Allow form submission
        }
    </script>
</head>
<body>
    <h2>Form Validation Example</h2>
    <form onsubmit="return validateForm()">
        <label for="name">Name:</label><br>
        <input type="text" id="name" name="name"><br><br>

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

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

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

Output 2:



Form Validation Example

Learning Outcomes:

1. Implement mathematical operations using JavaScript
2. Validate form elements using JavaScript

Course Outcomes:

1. Develop interactive forms with validation
2. Apply mathematical operations using JavaScript

Conclusion:

Viva Questions:

1. Which JavaScript function calculates factorial?
2. Which method is used for form validation?
3. Why validation is used?

For Faculty use:

Correction Parameters	Formative Assessment[40%]	Timely Completion of Practical[40%]	Attendance Learning Attitude[20%]

Experiment No.—5: JavaScript

Aim: Write JavaScript code for

- **a.** Demonstrating different JavaScript Objects such as String, RegExp, Math, Date
- **b.** Demonstrating different JavaScript Objects such as Window, Navigator, History, Location, Document
- **c.** Storing and Retrieving Cookies

Tools & Technologies used: VS Code, HTML, JavaScript, Chrome

Learning Objectives:

1. Learn to work with built-in JavaScript objects
2. Store and retrieve data using cookies

Theory:

JavaScript Objects (String, RegExp, Math, Date):

- String: The String object provides methods for manipulating strings, such as length, slice(), replace(), and toUpperCase(). It allows easy string manipulation and pattern matching.
- RegExp: The RegExp object is used for working with regular expressions, enabling pattern matching in strings. Methods like test() and exec() are used to search for patterns in text.
- Math: The Math object offers mathematical operations like Math.random(), Math.pow(), Math.sqrt(), and constants like Math.PI. It simplifies complex math operations.
- Date: The Date object handles date and time functionality. Methods like getDate(), getMonth(), getFullYear(), and getTime() allow for creating, formatting, and manipulating dates and times.

Browser-Specific JavaScript Objects (Window, Navigator, History, Location, Document):

- Window: The Window object represents the browser window and provides methods for controlling window size, opening new windows (window.open()), and accessing the browser environment.
- Navigator: The Navigator object provides information about the browser, such as the browser name, version, and platform. It is often used for feature detection (e.g., checking if a browser supports cookies).
- History: The History object allows manipulation of the browser history. Methods like back(), forward(), and go() enable navigation through previously visited pages.
- Location: The Location object represents the URL of the current page. It provides methods for retrieving and modifying the current page's URL and navigating to a new URL.
- Document: The Document object represents the web page's structure and provides methods for interacting with the HTML content, such as getElementById() and querySelector().

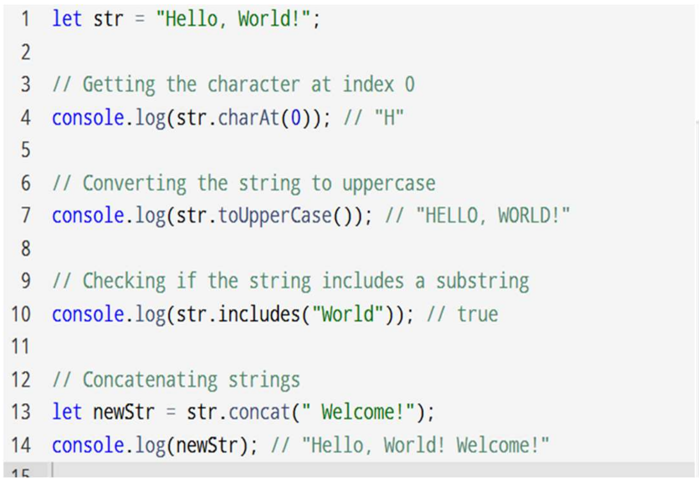
Storing and Retrieving Cookies:

- Cookies are small pieces of data stored in the user's browser that can be accessed by JavaScript. The document.cookie property is used to create, store, and retrieve cookies.

- Setting Cookies: To set a cookie, you can assign a string to document.cookie in the format name=value, and optionally, specify the expiry date and domain.
- Retrieving Cookies: Cookies can be retrieved using document.cookie, but they are returned as a single string, requiring parsing to access individual cookie values.
- Example: document.cookie = "username=JohnDoe; expires=Fri, 31 Dec 2024 23:59:59 GMT"; to store a cookie.

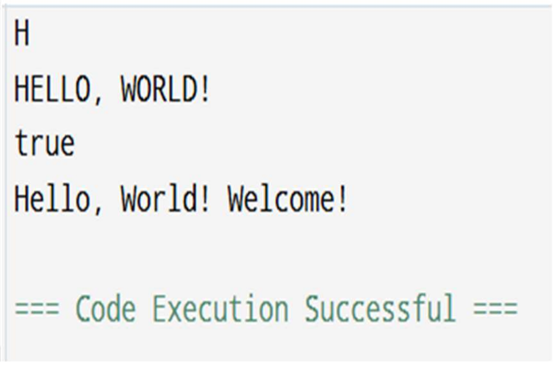
code

Code 1:



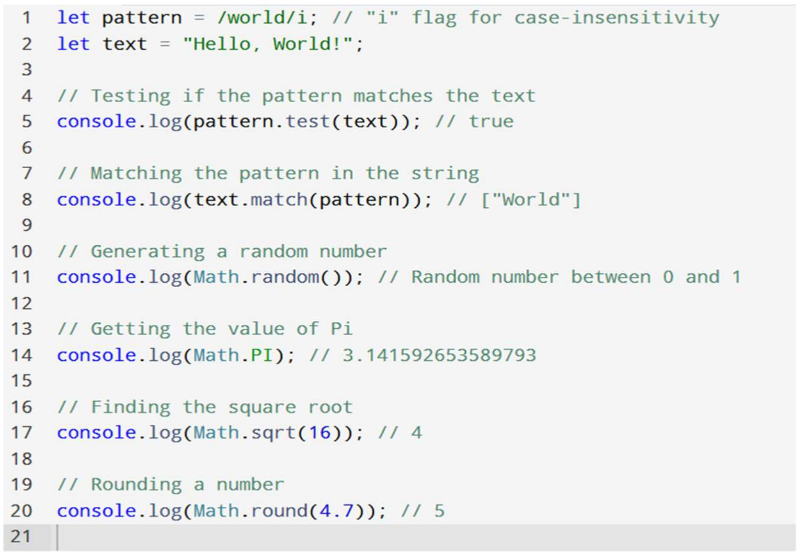
```
let str = "Hello, World!";

// Getting the character at index 0
console.log(str.charAt(0)); // "H"

// Converting the string to uppercase
console.log(str.toUpperCase()); // "HELLO, WORLD!"

// Checking if the string includes a substring
console.log(str.includes("World")); // true

// Concatenating strings
let newStr = str.concat(" Welcome!");
console.log(newStr); // "Hello, World! Welcome!"
```

Output 1:

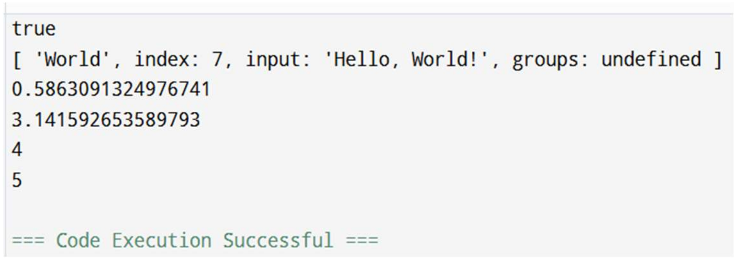


```
H
HELLO, WORLD!
true
Hello, World! Welcome!


=== Code Execution Successful ===
```

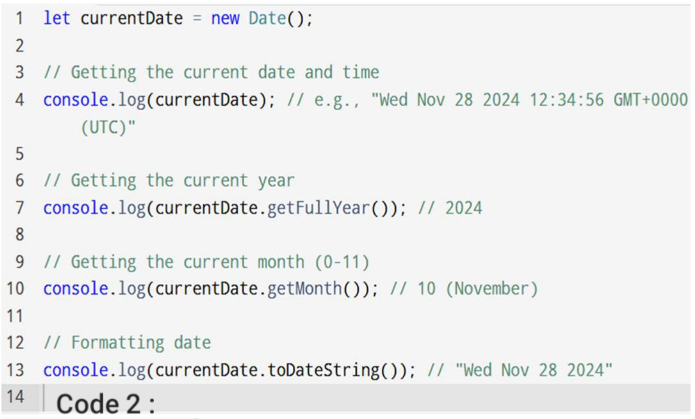
:



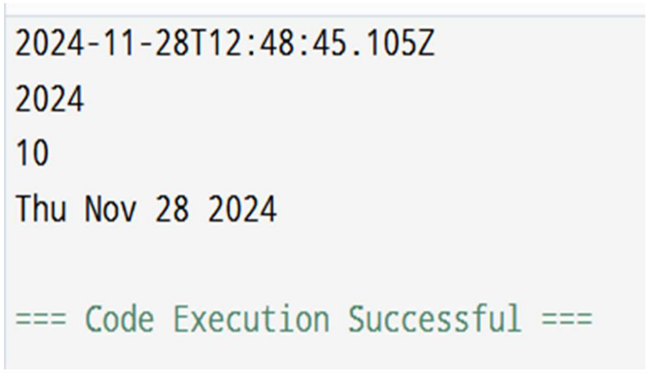
```
let pattern = /world/i; // "i" flag for case-insensitivity
let text = "Hello, World!";

// Testing if the pattern matches the text
console.log(pattern.test(text)); // true

// Matching the pattern in the string
console.log(text.match(pattern)); // ["World"]

// Generating a random number
console.log(Math.random()); // Random number between 0 and 1

// Getting the value of Pi
console.log(Math.PI); // 3.141592653589793

// Finding the square root
console.log(Math.sqrt(16)); // 4

// Rounding a number
console.log(Math.round(4.7)); // 5
```



```
true
[ 'World', index: 7, input: 'Hello, World!', groups: undefined ]
0.5863091324976741
3.141592653589793
4
5

=== Code Execution Successful ===
```

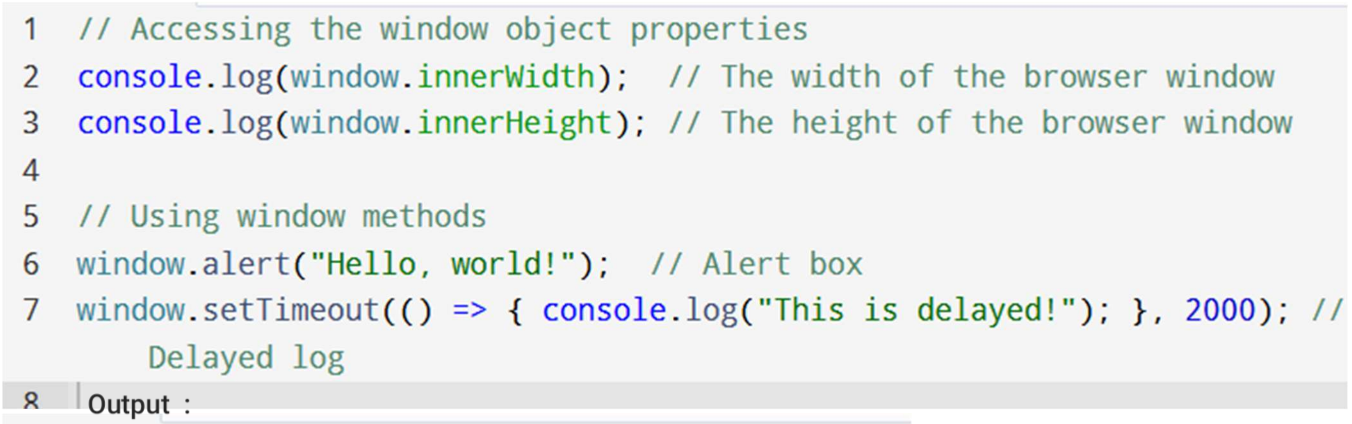


```
let currentDate = new Date();

// Getting the current date and time
console.log(currentDate); // e.g., "Wed Nov 28 2024 12:34:56 GMT+0000
    (UTC)"

// Getting the current year
console.log(currentDate.getFullYear()); // 2024

// Getting the current month (0-11)
console.log(currentDate.getMonth()); // 10 (November)

// Formatting date
console.log(currentDate.toDateString()); // "Wed Nov 28 2024"
```



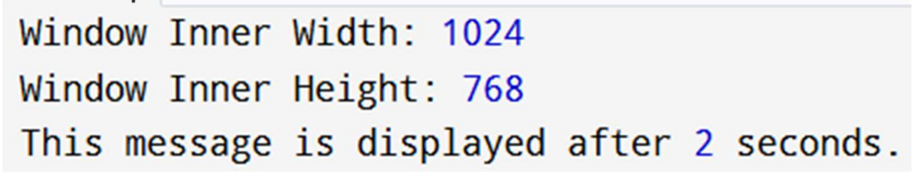
```
2024-11-28T12:48:45.105Z
2024
10
Thu Nov 28 2024


=== Code Execution Successful ===
```

Code 2 :

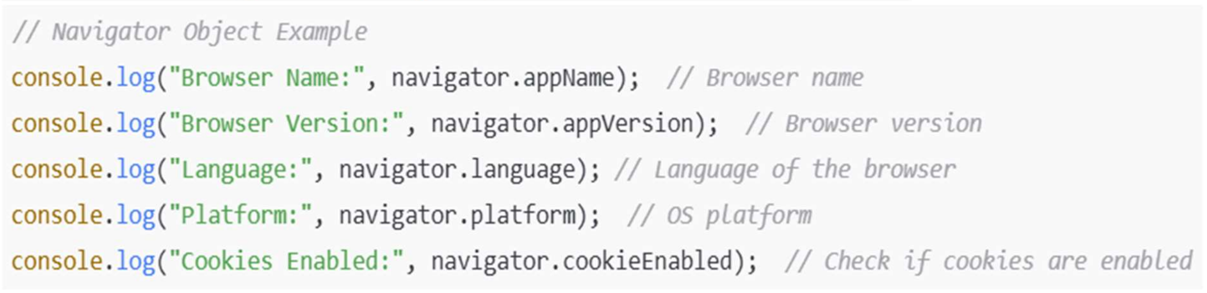


```
// Accessing the window object properties
console.log(window.innerWidth);  // The width of the browser window
console.log(window.innerHeight); // The height of the browser window

// Using window methods
window.alert("Hello, world!");  // Alert box
window.setTimeout(() => { console.log("This is delayed!"); }, 2000); //
    Delayed log
```

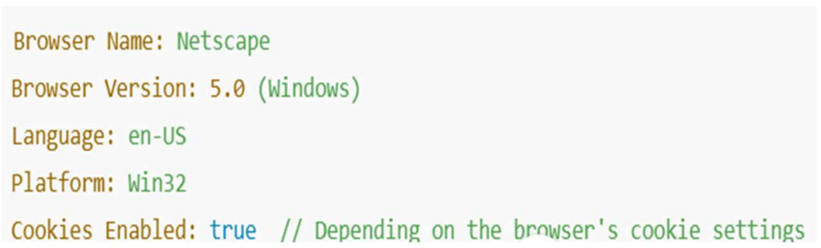
Output :



```
Window Inner Width: 1024
Window Inner Height: 768
This message is displayed after 2 seconds.
```



```
// Navigator Object Example
console.log("Browser Name:", navigator.appName);  // Browser name
console.log("Browser Version:", navigator.appVersion);  // Browser version
console.log("Language:", navigator.language); // Language of the browser
console.log("Platform:", navigator.platform);  // OS platform
console.log("Cookies Enabled:", navigator.cookieEnabled);  // Check if cookies are enabled
```



```
Browser Name: Netscape
Browser Version: 5.0 (Windows)
Language: en-US
Platform: Win32
Cookies Enabled: true  // Depending on the browser's cookie settings
```

```javascript
// History Object Example
console.log("History Length:", history.length);
```

```
History Length: 5
```

```javascript
// Location Object Example
console.log("Current URL:", location.href); // Full URL
console.log("Host:", location.hostname);  // Domain
console.log("Path:", location.pathname);  // Path part of the URL
console.log("Protocol:", location.protocol); // Protocol (e.g., http or https)
```

```
Current URL: https://www.example.com/index.html
Host: www.example.com
Path: /index.html
Protocol: https:
```

Code 3 :

```javascript
// Storing Cookies
const date = new Date();
date.setTime(date.getTime() + (7 * 24 * 60 * 60 * 1000)); // 7 days expiration
document.cookie = "username=JohnDoe; expires=" + date.toUTCString() + "; path=/";  // Cookie with expiration date
```

Output:

```
username=JohnDoe
```

```javascript
// Retrieving a Cookie
function getCookie(name) {
  const cookies = document.cookie.split(';');
  for (let i = 0; i < cookies.length; i++) {
    let cookie = cookies[i].trim();
    if (cookie.startsWith(name + "=")) {
      return cookie.substring(name.length + 1);
    }
  }
  return null;  // Return null if the cookie is not found
}

// Retrieving the 'username' cookie
const username = getCookie("username");
console.log("Retrieved username cookie:", username);
```

```
Retrieved username cookie: JohnDoe
```

```
console.log("All Cookies:", document.cookie);  // Logs all cookies stored
```

```
All Cookies: username=JohnDoe; user_id=12345;
```

Learning Outcomes:

1. Understand JavaScript object handling and usage
2. Manipulate cookies and browser objects in JavaScript

Course Outcomes:

1. Implement JavaScript objects and cookies
2. Access and manipulate browser-specific objects

Conclusion:

Viva Questions:

1. Which object provides date and time functionalities?
2. Which method is used to set a cookie in JavaScript?
3. Which JavaScript object is used for regular expressions?

For Faculty use:

Correction Parameters	Formative Assessment[40%]	Timely Completion of Practical[40%]	Attendance Learning Attitude[20%]

Experiment No.—6: XML

Aim: Create a XML file with Internal / External DTD and display it using

- **a.** CSS
- **b.** XSL

Tools & Technologies used: VS Code, HTML, CSS, XML, XSL, Chrome

Learning Objectives:

1. Learn XML data representation techniques
2. Master XSL for XML transformation

Theory:

XML (Extensible Markup Language):

- XML is a markup language designed to store and transport data in a human-readable format. It allows the definition of custom tags and provides a way to structure and store data hierarchically.
- XML documents consist of elements, attributes, and nested tags to represent data. An XML file is defined with a declaration (<?xml version="1.0" encoding="UTF-8"?>) and is validated using a DTD or XML Schema.

Internal and External DTD (Document Type Definition):

- Internal DTD: A DTD that is embedded within the XML document itself. It is declared in the <!DOCTYPE> declaration at the beginning of the XML document. It defines the structure and legal elements/attributes for the XML document.
- External DTD: A DTD that is stored in a separate file and referenced by the XML document using the <!DOCTYPE> declaration with a SYSTEM or PUBLIC identifier. External DTDs help in reusing the same structure for multiple XML files.

XSL (Extensible Stylesheet Language):

- XSL is a language used to transform and display XML data. It consists of three parts: XSLT (for transforming XML documents), XPath (for navigating XML documents), and XSL-FO (for formatting the output).
- XSLT: XSLT is used to transform an XML document into another format such as HTML, plain text, or even another XML document. It uses templates to match parts of an XML document and apply transformation rules.

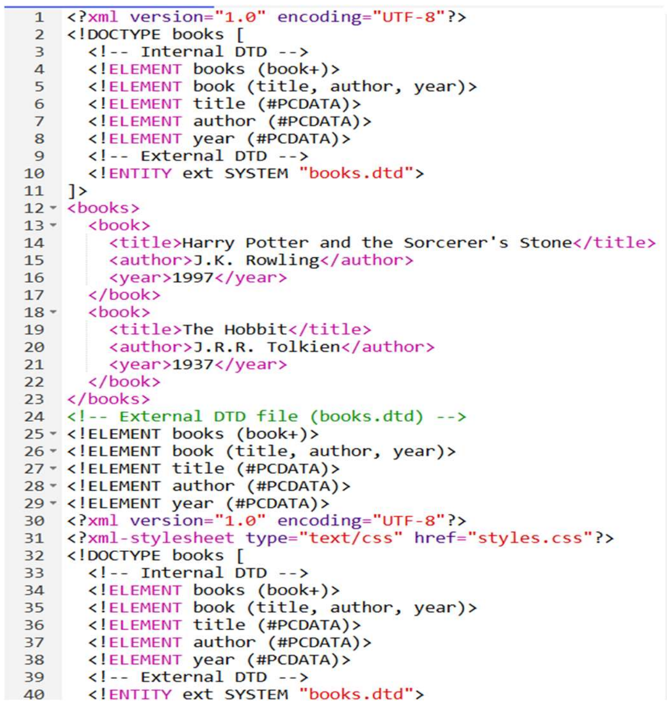
SHRI G.P.M. DEGREE COLLEGE OF SCIENCE & COMMERCE

Department of Computer

Affiliated to University of Mumbai

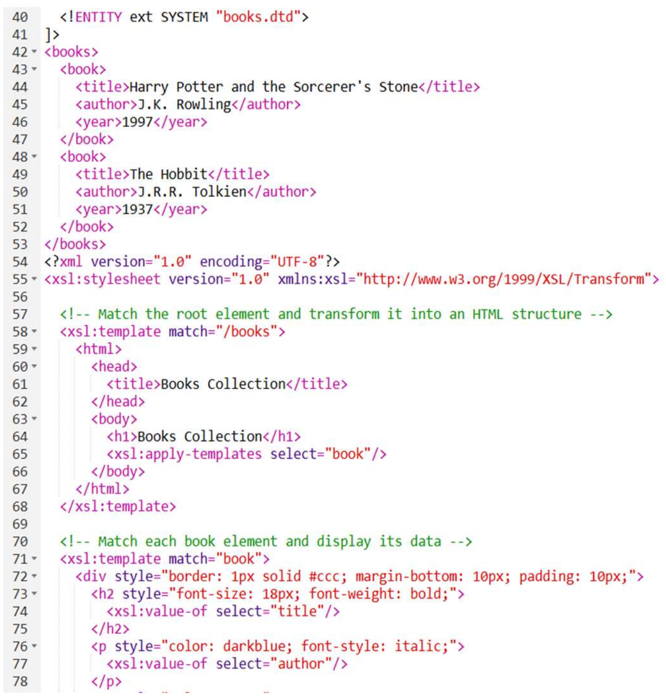
Design..Develop..Deploy..Deliver

Code :

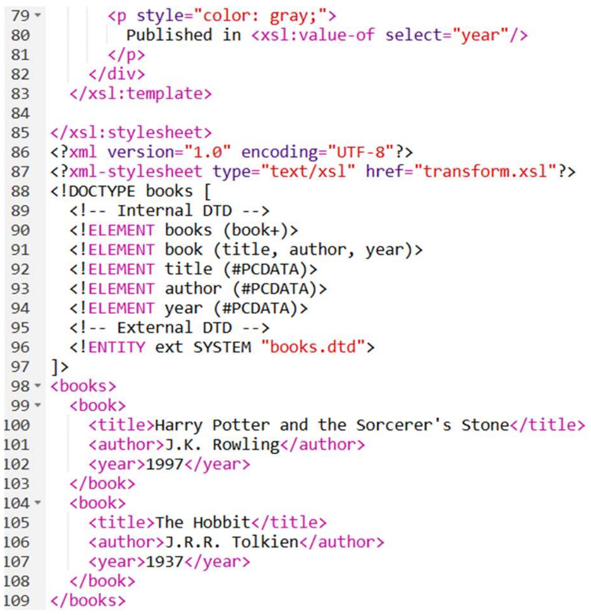


```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE books [
  <!-- Internal DTD -->
  <!ELEMENT books (book+)>
  <!ELEMENT book (title, author, year)>
  <!ELEMENT title (#PCDATA)>
  <!ELEMENT author (#PCDATA)>
  <!ELEMENT year (#PCDATA)>
  <!-- External DTD -->
  <!ENTITY ext SYSTEM "books.dtd">
]>
<books>
  <book>
    <title>Harry Potter and the Sorcerer's Stone</title>
    <author>J.K. Rowling</author>
    <year>1997</year>
  </book>
  <book>
    <title>The Hobbit</title>
    <author>J.R.R. Tolkien</author>
    <year>1937</year>
  </book>
</books>
<!-- External DTD file (books.dtd) -->
<!ELEMENT books (book+)>
<!ELEMENT book (title, author, year)>
<!ELEMENT title (#PCDATA)>
<!ELEMENT author (#PCDATA)>
<!ELEMENT year (#PCDATA)>
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet type="text/css" href="styles.css"?>
<!DOCTYPE books [
  <!-- Internal DTD -->
  <!ELEMENT books (book+)>
  <!ELEMENT book (title, author, year)>
  <!ELEMENT title (#PCDATA)>
  <!ELEMENT author (#PCDATA)>
  <!ELEMENT year (#PCDATA)>
  <!-- External DTD -->
  <!ENTITY ext SYSTEM "books.dtd">
```

Code b:

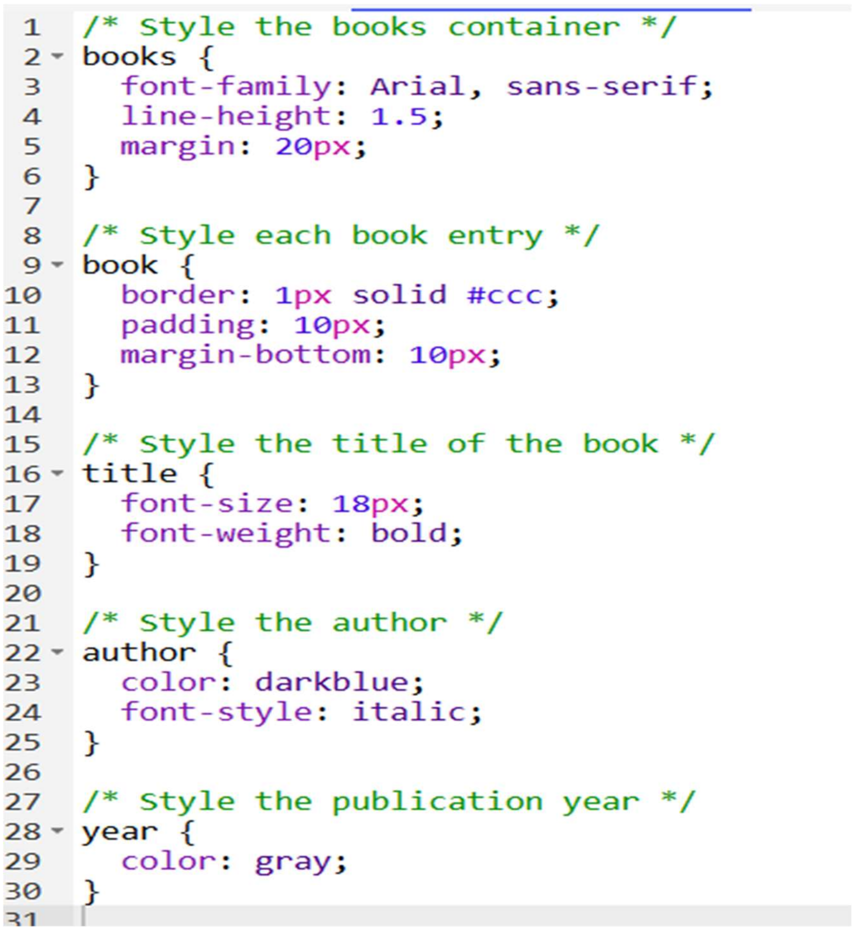


```
  <!ENTITY ext SYSTEM "books.dtd">
]>
<books>
  <book>
    <title>Harry Potter and the Sorcerer's Stone</title>
    <author>J.K. Rowling</author>
    <year>1997</year>
  </book>
  <book>
    <title>The Hobbit</title>
    <author>J.R.R. Tolkien</author>
    <year>1937</year>
  </book>
</books>
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

  <!-- Match the root element and transform it into an HTML structure -->
  <xsl:template match="/books">
    <html>
      <head>
        <title>Books Collection</title>
      </head>
      <body>
        <h1>Books Collection</h1>
        <xsl:apply-templates select="book"/>
      </body>
    </html>
  </xsl:template>

  <!-- Match each book element and display its data -->
  <xsl:template match="book">
    <div style="border: 1px solid #ccc; margin-bottom: 10px; padding: 10px;">
      <h2 style="font-size: 18px; font-weight: bold;">
        <xsl:value-of select="title"/>
      </h2>
      <p style="color: darkblue; font-style: italic;">
        <xsl:value-of select="author"/>
      </p>
```



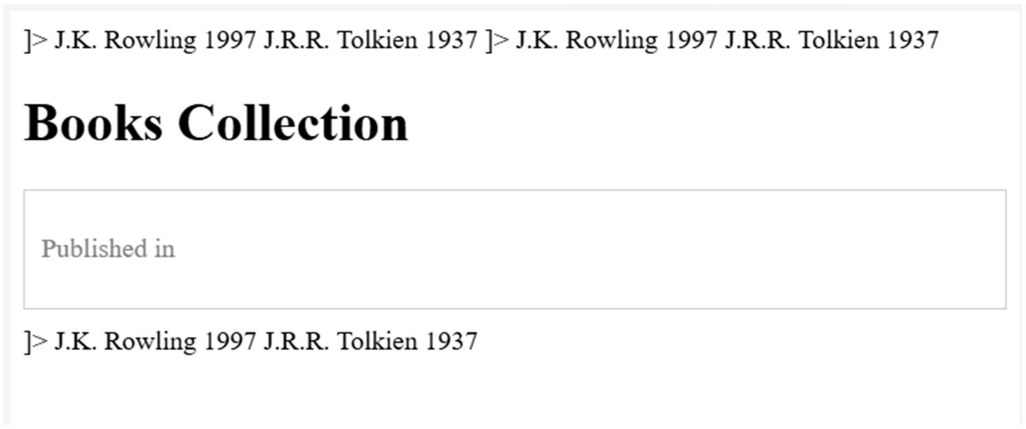
```
      <p style="color: gray;">
        Published in <xsl:value-of select="year"/>
      </p>
    </div>
  </xsl:template>

</xsl:stylesheet>
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet type="text/xsl" href="transform.xsl"?>
<!DOCTYPE books [
  <!-- Internal DTD -->
  <!ELEMENT books (book+)>
  <!ELEMENT book (title, author, year)>
  <!ELEMENT title (#PCDATA)>
  <!ELEMENT author (#PCDATA)>
  <!ELEMENT year (#PCDATA)>
  <!-- External DTD -->
  <!ENTITY ext SYSTEM "books.dtd">
]>
<books>
  <book>
    <title>Harry Potter and the Sorcerer's Stone</title>
    <author>J.K. Rowling</author>
    <year>1997</year>
  </book>
  <book>
    <title>The Hobbit</title>
    <author>J.R.R. Tolkien</author>
    <year>1937</year>
  </book>
</books>
```

Code a:



```
/* Style the books container */
books {
  font-family: Arial, sans-serif;
  line-height: 1.5;
  margin: 20px;
}

/* Style each book entry */
book {
  border: 1px solid #ccc;
  padding: 10px;
  margin-bottom: 10px;
}

/* Style the title of the book */
title {
  font-size: 18px;
  font-weight: bold;
}

/* Style the author */
author {
  color: darkblue;
  font-style: italic;
}

/* Style the publication year */
year {
  color: gray;
}
```

Output :



```
]> J.K. Rowling 1997 J.R.R. Tolkien 1937 ]> J.K. Rowling 1997 J.R.R. Tolkien 1937

Books Collection

Published in

]> J.K. Rowling 1997 J.R.R. Tolkien 1937
```

Learning Outcomes:

1. Understand XML structure and syntax
2. Apply DTD for XML validation

Course Outcomes:

1. Create and validate XML document
2. Transform XML using XSLT

Conclusion:

Viva Questions:

1. What language defines XML structure?
2. Which XSL component transforms XML data?
3. What is used to navigate XML?

For Faculty use:

Correction Parameters	Formative Assessment[40%]	Timely Completion of Practical[40%]	Attendance Learning Attitude[20%]

Experiment No.—7: AJAX

Aim: Design a webpage to handle asynchronous requests using AJAX on

a. Mouseover
b. Button click

Tools & Technologies used: VS Code, HTML, AJAX, Chrome

Learning Objectives:

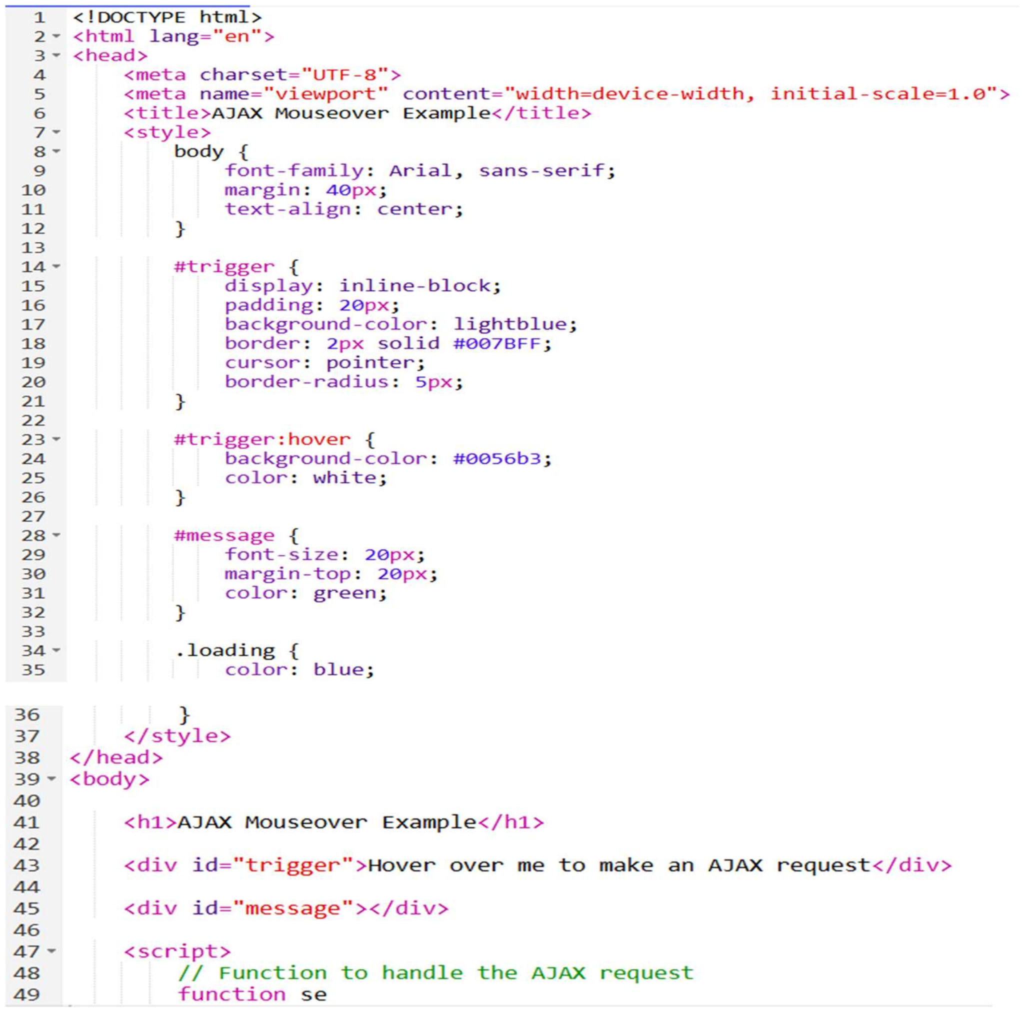
1. Learn AJAX to handle mouseover events
2. Use AJAX for button click interactions

Theory:

AJAX (Asynchronous JavaScript and XML):

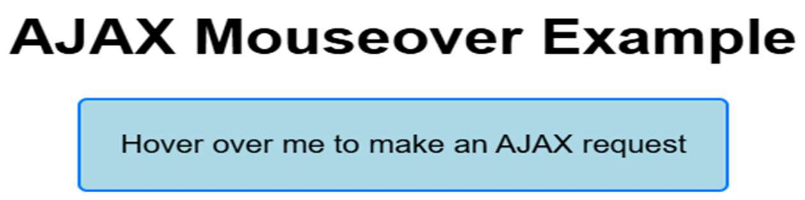
- Asynchronous Data Handling:
  AJAX allows web pages to retrieve data from a server asynchronously without reloading the entire page. This enables dynamic content updates, providing a smoother user experience.
- Uses XMLHttpRequest Object:
  AJAX uses the XMLHttpRequest object (or Fetch API in modern JavaScript) to send and receive data from the server. This communication happens in the background, enabling seamless updates on the page.
- Common Events and Methods:
  AJAX is often triggered by events like onclick, onmouseover, or onload. Data is usually returned in formats like JSON or XML, which are then processed and displayed on the web page.

Code 1 :



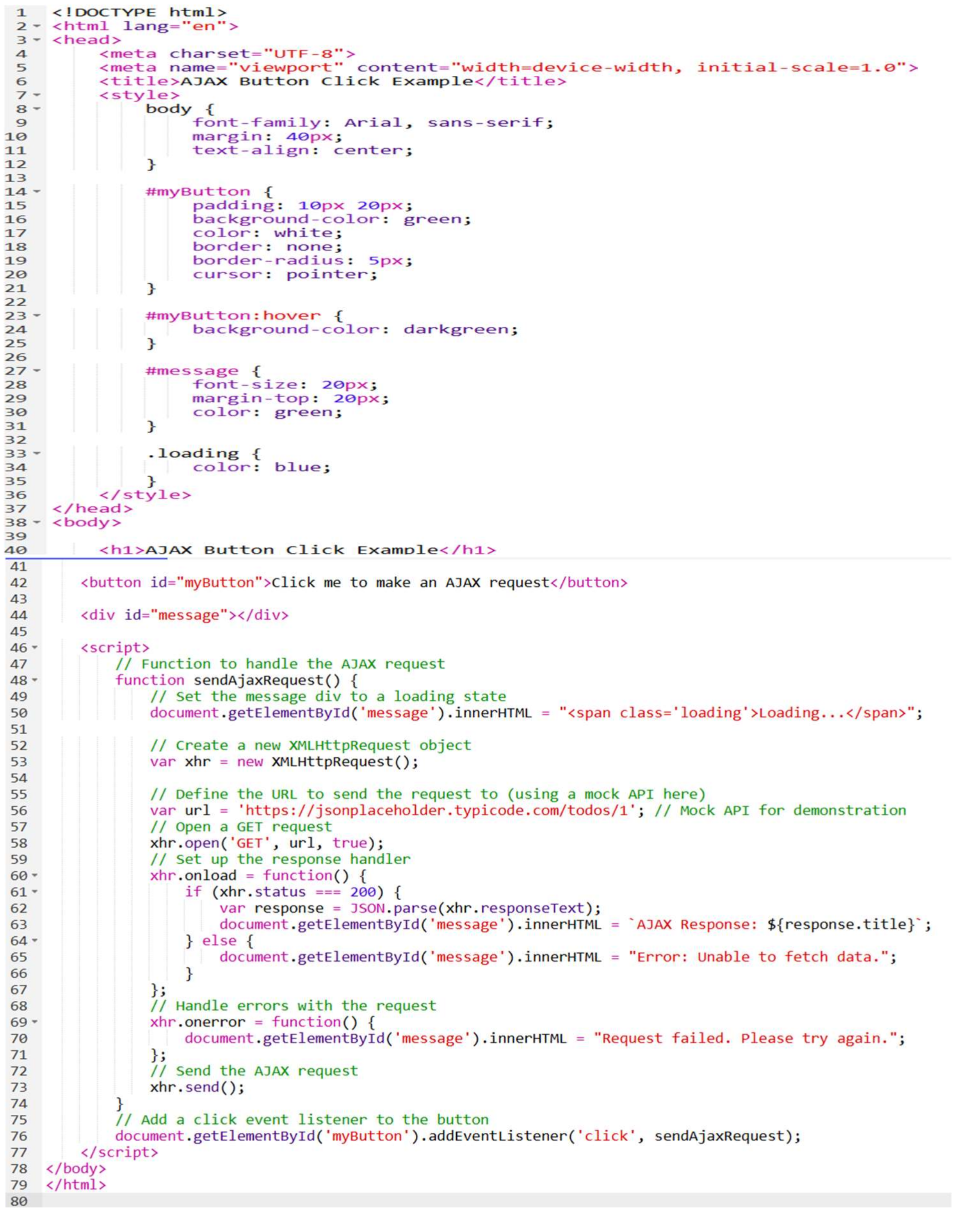
```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>AJAX Mouseover Example</title>
    <style>
        body {
            font-family: Arial, sans-serif;
            margin: 40px;
            text-align: center;
        }

        #trigger {
            display: inline-block;
            padding: 20px;
            background-color: lightblue;
            border: 2px solid #007BFF;
            cursor: pointer;
            border-radius: 5px;
        }

        #trigger:hover {
            background-color: #0056b3;
            color: white;
        }

        #message {
            font-size: 20px;
            margin-top: 20px;
            color: green;
        }

        .loading {
            color: blue;
        }
    </style>
</head>
<body>

    <h1>AJAX Mouseover Example</h1>

    <div id="trigger">Hover over me to make an AJAX request</div>

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

    <script>
        // Function to handle the AJAX request
        function se
```

Output :



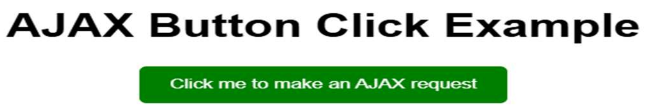
```
AJAX Mouseover Example

Hover over me to make an AJAX request
```

Code2:



```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>AJAX Button Click Example</title>
    <style>
        body {
            font-family: Arial, sans-serif;
            margin: 40px;
            text-align: center;
        }

        #myButton {
            padding: 10px 20px;
            background-color: green;
            color: white;
            border: none;
            border-radius: 5px;
            cursor: pointer;
        }

        #myButton:hover {
            background-color: darkgreen;
        }

        #message {
            font-size: 20px;
            margin-top: 20px;
            color: green;
        }

        .loading {
            color: blue;
        }
    </style>
</head>
<body>

    <h1>AJAX Button Click Example</h1>

    <button id="myButton">Click me to make an AJAX request</button>

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

    <script>
        // Function to handle the AJAX request
        function sendAjaxRequest() {
            // Set the message div to a loading state
            document.getElementById('message').innerHTML = "<span class='loading'>Loading...</span>";

            // Create a new XMLHttpRequest object
            var xhr = new XMLHttpRequest();

            // Define the URL to send the request to (using a mock API here)
            var url = 'https://jsonplaceholder.typicode.com/todos/1'; // Mock API for demonstration
            // Open a GET request
            xhr.open('GET', url, true);
            // Set up the response handler
            xhr.onload = function() {
                if (xhr.status === 200) {
                    var response = JSON.parse(xhr.responseText);
                    document.getElementById('message').innerHTML = `AJAX Response: ${response.title}`;
                } else {
                    document.getElementById('message').innerHTML = "Error: Unable to fetch data.";
                }
            };
            // Handle errors with the request
            xhr.onerror = function() {
                document.getElementById('message').innerHTML = "Request failed. Please try again.";
            };
            // Send the AJAX request
            xhr.send();
        }
        // Add a click event listener to the button
        document.getElementById('myButton').addEventListener('click', sendAjaxRequest);
    </script>
</body>
</html>
```

Output :



Learning Outcomes:

1. Understand AJAX for asynchronous operations
2. Handle events with AJAX requests

Course Outcomes:

1. Design interactive pages with AJAX
2. Apply AJAX to handle dynamic requests

Conclusion:

Viva Questions:

1. What does AJAX stand for?
2. Which event triggers AJAX on mouseover?
3. Which method sends an AJAX request?

For Faculty use:

Correction Parameters	Formative Assessment[40%]	Timely Completion of Practical[40%]	Attendance Learning Attitude[20%]

Experiment No.—8: PHP

Aim: Write PHP scripts for

- **a.** Retrieving data from HTML forms
- **b.** Performing certain mathematical operations such as finding Fibonacci Series
- **c.** Working with Arrays
- **d.** Working with Files (Reading / Writing)

Tools & Technologies used: VS Code, HTML, PHP, Chrome

Learning Objectives:

1. Retrieve and process HTML form data
2. Perform mathematical operations and file handling

Theory:

Introduction to PHP:

- PHP stands for Hypertext Preprocessor (a recursive acronym).
- It is an open-source, server-side scripting language primarily used for web development.
- PHP is embedded in HTML to generate dynamic web pages.

Working with Arrays:

- PHP supports indexed arrays (numeric keys) and associative arrays (string keys).
- Arrays are powerful tools to store multiple values and can be manipulated using functions like array_merge(), array_push(), and array_pop().

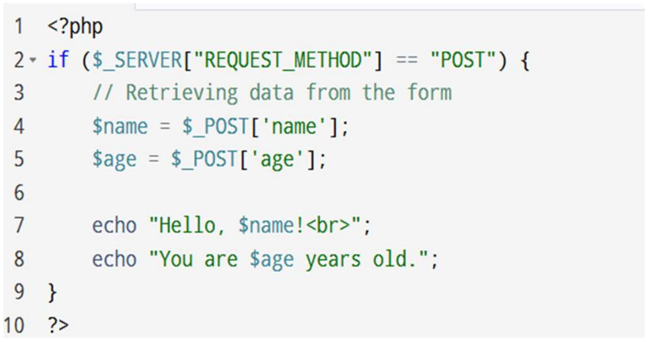
PHP and Forms:

- PHP is commonly used to process form data submitted via HTML forms.
- Data can be retrieved using $_GET (for GET requests) or $_POST (for POST requests).
- Form data can be validated and stored in a database or processed for other purposes.

File Handling:

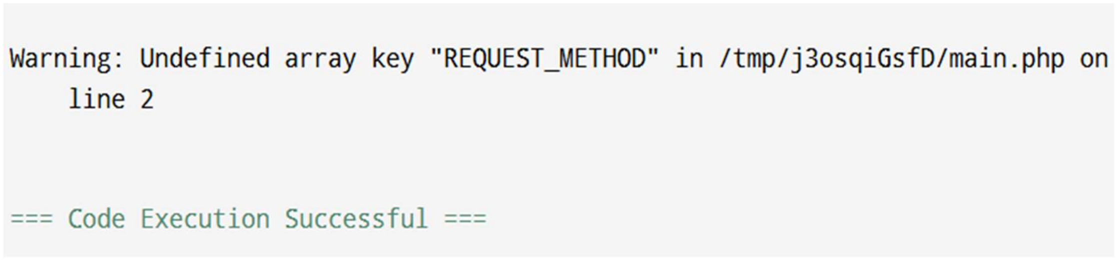
- PHP can handle files for reading, writing, and modifying.
- Key functions include fopen(), fread(), fwrite(), fclose(), and file_get_contents().
- Files can be opened in various modes like r (read), w (write), and a (append).

Code1:



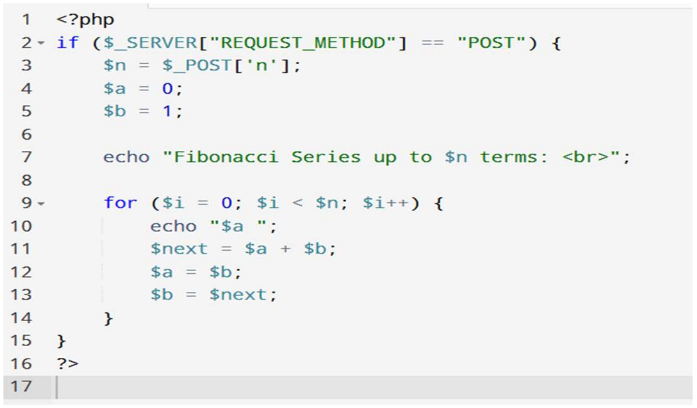
```php
<?php
if ($_SERVER["REQUEST_METHOD"] == "POST") {
    // Retrieving data from the form
    $name = $_POST['name'];
    $age = $_POST['age'];

    echo "Hello, $name!<br>";
    echo "You are $age years old.";
}
?>
```

Output 1:



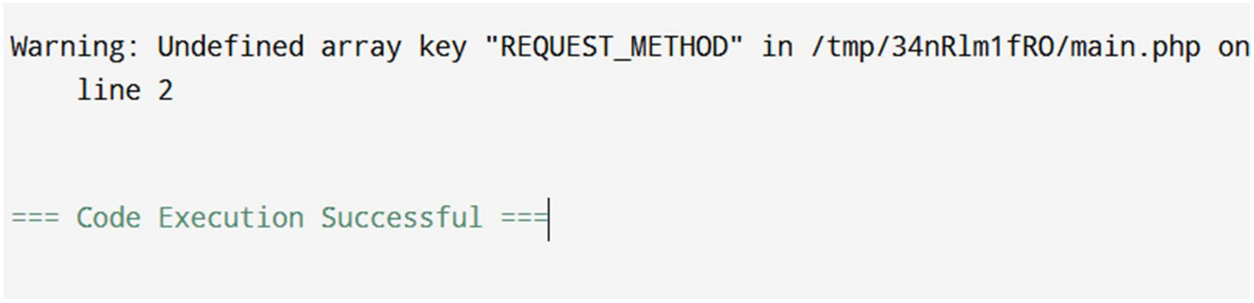
```
Warning: Undefined array key "REQUEST_METHOD" in /tmp/j3osqiGsfD/main.php on
    line 2


=== Code Execution Successful ===
```

Code2:



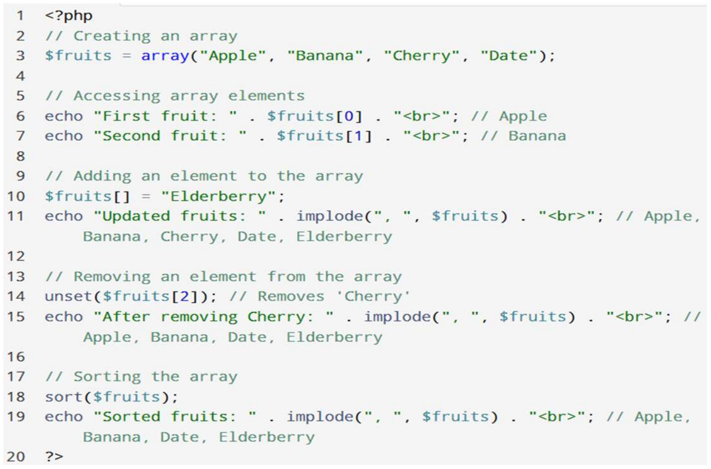
```php
<?php
if ($_SERVER["REQUEST_METHOD"] == "POST") {
    $n = $_POST['n'];
    $a = 0;
    $b = 1;

    echo "Fibonacci Series up to $n terms: <br>";

    for ($i = 0; $i < $n; $i++) {
        echo "$a ";
        $next = $a + $b;
        $a = $b;
        $b = $next;
    }
}
?>

```

Output 2:



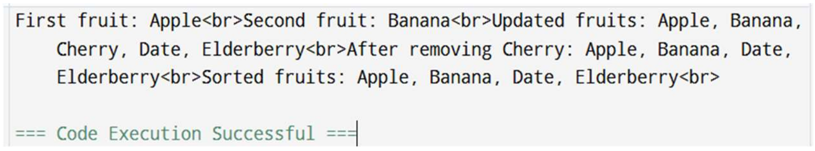
```
Warning: Undefined array key "REQUEST_METHOD" in /tmp/34nRlm1fRO/main.php on
    line 2


=== Code Execution Successful ===
```

Code3:



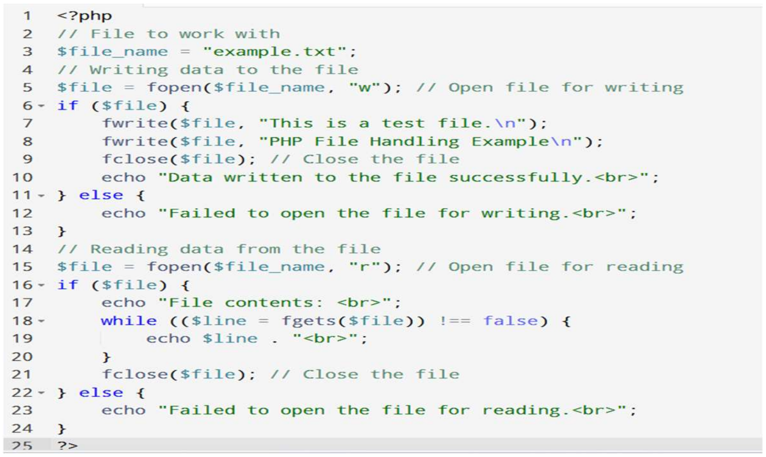
```php
<?php
// Creating an array
$fruits = array("Apple", "Banana", "Cherry", "Date");

// Accessing array elements
echo "First fruit: " . $fruits[0] . "<br>"; // Apple
echo "Second fruit: " . $fruits[1] . "<br>"; // Banana

// Adding an element to the array
$fruits[] = "Elderberry";
echo "Updated fruits: " . implode(", ", $fruits) . "<br>"; // Apple,
    Banana, Cherry, Date, Elderberry

// Removing an element from the array
unset($fruits[2]); // Removes 'Cherry'
echo "After removing Cherry: " . implode(", ", $fruits) . "<br>"; //
    Apple, Banana, Date, Elderberry

// Sorting the array
sort($fruits);
echo "Sorted fruits: " . implode(", ", $fruits) . "<br>"; // Apple,
    Banana, Date, Elderberry
?>
```

Output 3:



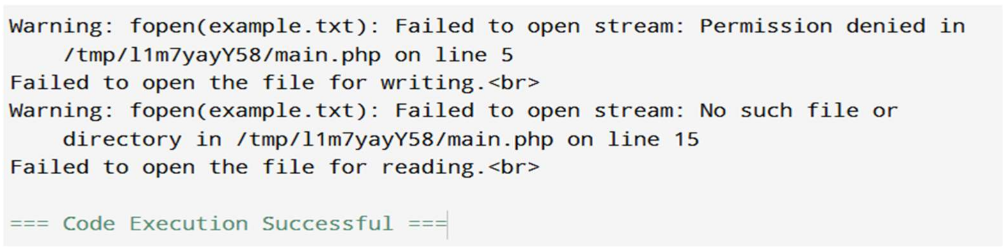
```
First fruit: Apple<br>Second fruit: Banana<br>Updated fruits: Apple, Banana,
    Cherry, Date, Elderberry<br>After removing Cherry: Apple, Banana, Date,
    Elderberry<br>Sorted fruits: Apple, Banana, Date, Elderberry<br>

=== Code Execution Successful ===
```

Code4:



```php
<?php
// File to work with
$file_name = "example.txt";
// Writing data to the file
$file = fopen($file_name, "w"); // Open file for writing
if ($file) {
    fwrite($file, "This is a test file.\n");
    fwrite($file, "PHP File Handling Example\n");
    fclose($file); // Close the file
    echo "Data written to the file successfully.<br>";
} else {
    echo "Failed to open the file for writing.<br>";
}
// Reading data from the file
$file = fopen($file_name, "r"); // Open file for reading
if ($file) {
    echo "File contents: <br>";
    while (($line = fgets($file)) !== false) {
        echo $line . "<br>";
    }
    fclose($file); // Close the file
} else {
    echo "Failed to open the file for reading.<br>";
}
?>
```

Output 4:



```
Warning: fopen(example.txt): Failed to open stream: Permission denied in
    /tmp/l1m7yayY58/main.php on line 5
Failed to open the file for writing.<br>
Warning: fopen(example.txt): Failed to open stream: No such file or
    directory in /tmp/l1m7yayY58/main.php on line 15
Failed to open the file for reading.<br>

=== Code Execution Successful ===
```

Learning Outcomes:

1. Retrieve and process form data in PHP
2. Read and write data from files

Course Outcomes:

1. Demonstrate form data handling with PHP
2. Apply mathematical algorithms and arrays

Conclusion:

Viva Questions:

1. PHP function for retrieving form data?
2. Which function is used to read from a file?

For Faculty use:

Correction Parameters	Formative Assessment[40%]	Timely Completion of Practical[40%]	Attendance Learning Attitude[20%]

Experiment No.—9: PHP

Aim: Write PHP scripts for

a. Working with Databases (Storing Records / Retrieving Records and Display them)
b. Storing and Retrieving Cookies
c. Storing and Retrieving Sessions

Tools & Technologies used: VS Code, HTML, PHP, Chrome

Learning Objectives:

1. Learn to interact with databases in PHP
2. Understand cookies and sessions in PHP

Theory:

Working with Databases in PHP:

- Database Connection:
    - PHP can interact with databases like MySQL, PostgreSQL, etc.
    - To connect to a MySQL database, use the mysqli_connect() function.
    - Example: $conn = mysqli_connect('localhost', 'username', 'password', 'database_name');
- Storing Records:
    - To store data in a database, use SQL INSERT queries.
    - Example: INSERT INTO users (name, email) VALUES ('John Doe', 'john@example.com');
- Retrieving Records:
    - Use SELECT queries to retrieve data.
    - Example: SELECT * FROM users;
    - Use mysqli_fetch_assoc() to fetch the result in associative array form.
- Displaying Data:
    - Display fetched records in HTML by looping through the result set.

Working with Cookies in PHP:

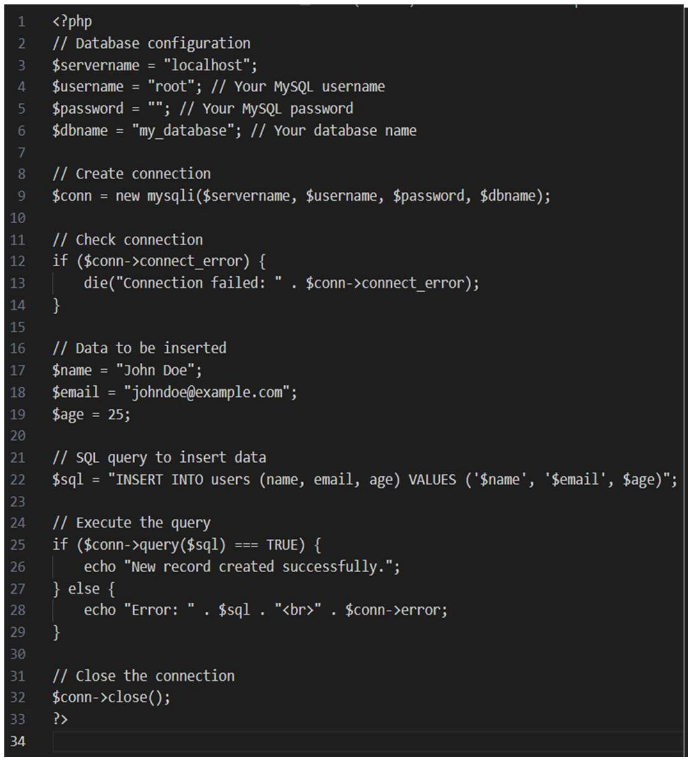
- Setting Cookies:
    - Cookies are stored on the client's browser.
    - Use the setcookie() function to create a cookie.
- Retrieving Cookies:
    - Use the $_COOKIE superglobal to access cookie data.
    - Example: echo $_COOKIE['user'];
- Deleting Cookies:
    - Set the expiration time of the cookie to a past time to delete it.
    - Example: setcookie('user', '', time() - 3600);

Working with Sessions in PHP:

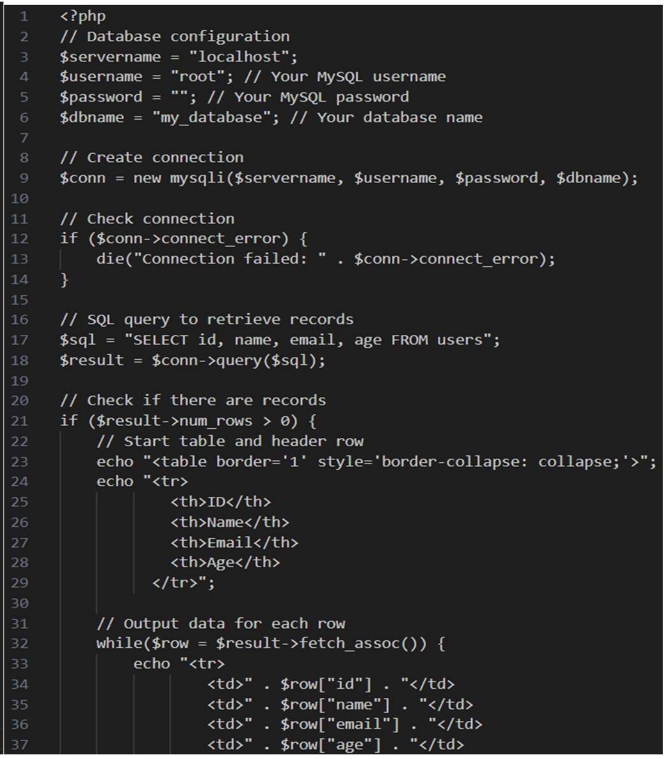
- Starting a Session:

  - Sessions store data on the server-side and persist across multiple pages.
  - To use sessions, call session_start() at the beginning of each PHP script.
  - Example: session_start();
- Storing Data in Sessions:
  - Store session data using the $_SESSION superglobal.
  - Example: $_SESSION['username'] = 'John Doe';
- Retrieving Session Data:
  - Access stored session data using $_SESSION array.
  - Example: echo $_SESSION['username'];
- Ending a Session:
  - Destroy the session using session_destroy() to remove all session variables.
  - Example: session_destroy();

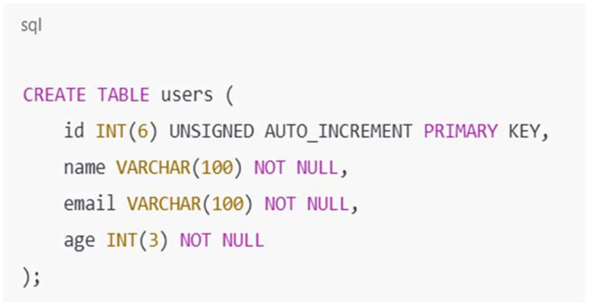
Code1:



```php
<?php
// Database configuration
$servername = "localhost";
$username = "root"; // Your MySQL username
$password = ""; // Your MySQL password
$dbname = "my_database"; // Your database name

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

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

// Data to be inserted
$name = "John Doe";
$email = "johndoe@example.com";
$age = 25;

// SQL query to insert data
$sql = "INSERT INTO users (name, email, age) VALUES ('$name', '$email', $age)";

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

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



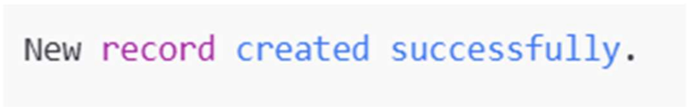
```php
<?php
// Database configuration
$servername = "localhost";
$username = "root"; // Your MySQL username
$password = ""; // Your MySQL password
$dbname = "my_database"; // Your database name

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

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

// SQL query to retrieve records
$sql = "SELECT id, name, email, age FROM users";
$result = $conn->query($sql);

// Check if there are records
if ($result->num_rows > 0) {
    // Start table and header row
    echo "<table border='1' style='border-collapse: collapse;'>";
    echo "<tr>
            <th>ID</th>
            <th>Name</th>
            <th>Email</th>
            <th>Age</th>
          </tr>";

    // Output data for each row
    while($row = $result->fetch_assoc()) {
        echo "<tr>
                <td>" . $row["id"] . "</td>
                <td>" . $row["name"] . "</td>
                <td>" . $row["email"] . "</td>
                <td>" . $row["age"] . "</td>
```



sql

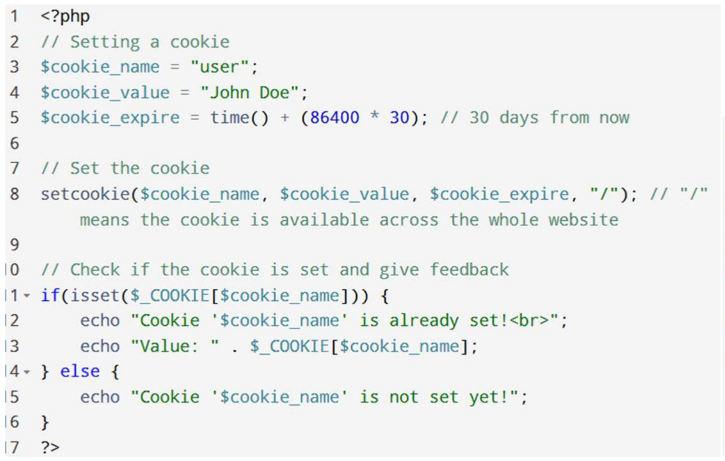
```sql
CREATE TABLE users (
    id INT(6) UNSIGNED AUTO_INCREMENT PRIMARY KEY,
    name VARCHAR(100) NOT NULL,
    email VARCHAR(100) NOT NULL,
    age INT(3) NOT NULL
);
```

Output 1:

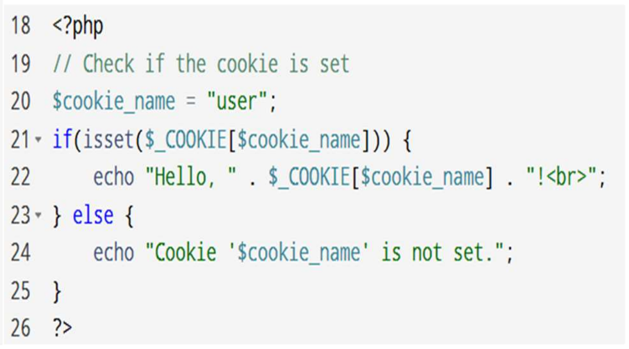


```
New record created successfully.
```

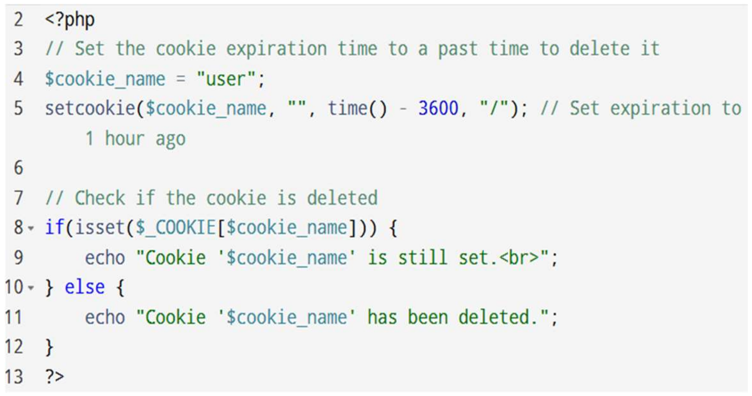
Code2:



```php
<?php
// Setting a cookie
$cookie_name = "user";
$cookie_value = "John Doe";
$cookie_expire = time() + (86400 * 30); // 30 days from now

// Set the cookie
setcookie($cookie_name, $cookie_value, $cookie_expire, "/"); // "/"
    means the cookie is available across the whole website

// Check if the cookie is set and give feedback
if(isset($_COOKIE[$cookie_name])) {
    echo "Cookie '$cookie_name' is already set!<br>";
    echo "Value: " . $_COOKIE[$cookie_name];
} else {
    echo "Cookie '$cookie_name' is not set yet!";
}
?>
```

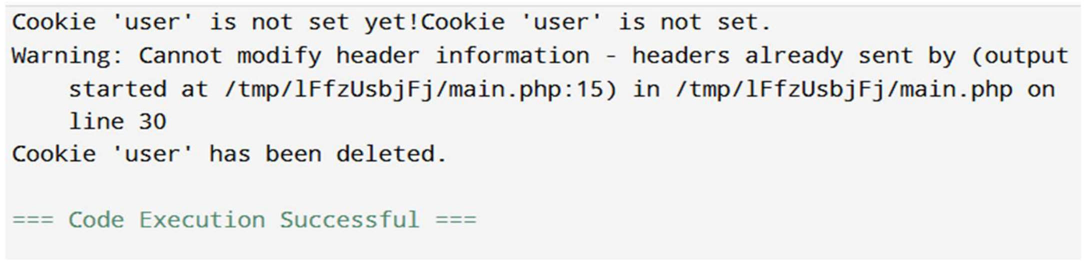


```php
<?php
// Check if the cookie is set
$cookie_name = "user";
if(isset($_COOKIE[$cookie_name])) {
    echo "Hello, " . $_COOKIE[$cookie_name] . "!<br>";
} else {
    echo "Cookie '$cookie_name' is not set.";
}
?>
```



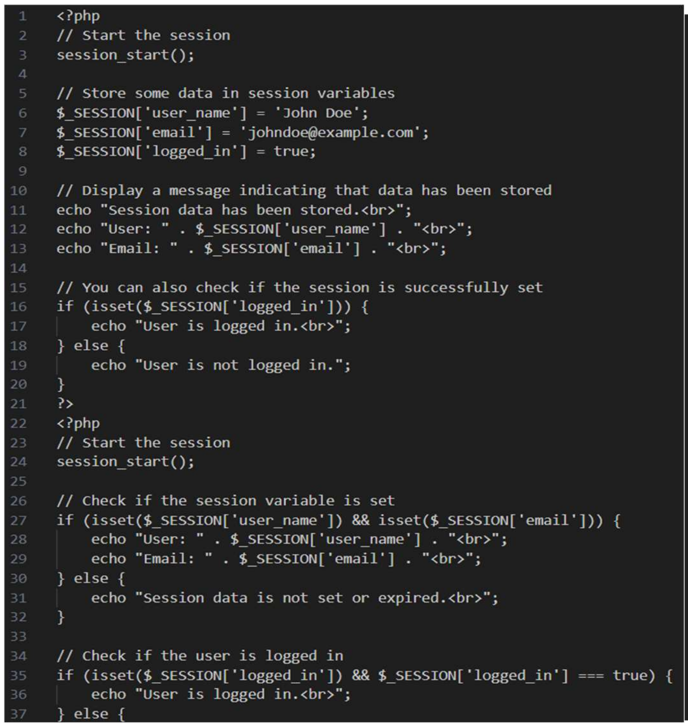
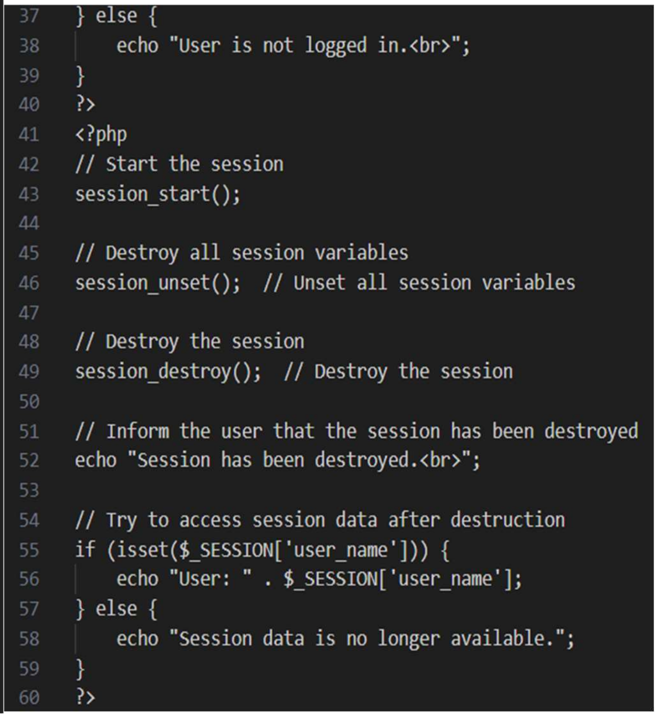
```php
<?php
// Set the cookie expiration time to a past time to delete it
$cookie_name = "user";
setcookie($cookie_name, "", time() - 3600, "/"); // Set expiration to
    1 hour ago

// Check if the cookie is deleted
if(isset($_COOKIE[$cookie_name])) {
    echo "Cookie '$cookie_name' is still set.<br>";
} else {
    echo "Cookie '$cookie_name' has been deleted.";
}
?>
```

Output 2:



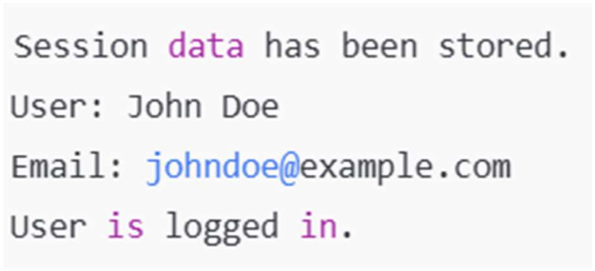
```
Cookie 'user' is not set yet!Cookie 'user' is not set.
Warning: Cannot modify header information - headers already sent by (output
    started at /tmp/lFfzUsbjFj/main.php:15) in /tmp/lFfzUsbjFj/main.php on
    line 30
Cookie 'user' has been deleted.

=== Code Execution Successful ===
```

Code3:

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

// Store some data in session variables
$_SESSION['user_name'] = 'John Doe';
$_SESSION['email'] = 'johndoe@example.com';
$_SESSION['logged_in'] = true;

// Display a message indicating that data has been stored
echo "Session data has been stored.<br>";
echo "User: " . $_SESSION['user_name'] . "<br>";
echo "Email: " . $_SESSION['email'] . "<br>";

// You can also check if the session is successfully set
if (isset($_SESSION['logged_in'])) {
    echo "User is logged in.<br>";
} else {
    echo "User is not logged in.";
}
?>
<?php
// Start the session
session_start();

// Check if the session variable is set
if (isset($_SESSION['user_name']) && isset($_SESSION['email'])) {
    echo "User: " . $_SESSION['user_name'] . "<br>";
    echo "Email: " . $_SESSION['email'] . "<br>";
} else {
    echo "Session data is not set or expired.<br>";
}

// Check if the user is logged in
if (isset($_SESSION['logged_in']) && $_SESSION['logged_in'] === true) {
    echo "User is logged in.<br>";
} else {
    echo "User is not logged in.<br>";
}
?>
<?php
// Start the session
session_start();

// Destroy all session variables
session_unset();  // Unset all session variables

// Destroy the session
session_destroy();  // Destroy the session

// Inform the user that the session has been destroyed
echo "Session has been destroyed.<br>";

// Try to access session data after destruction
if (isset($_SESSION['user_name'])) {
    echo "User: " . $_SESSION['user_name'];
} else {
    echo "Session data is no longer available.";
}
?>
```

Output 3:



```
Session data has been stored.
User: John Doe
Email: johndoe@example.com
User is logged in.
```

Learning Outcomes:

1. Store and retrieve database records in PHP
2. Work with cookies for data storage and Handle user sessions for dynamic data

Course Outcomes:

1. Perform CRUD operations with PHP and MySQL
2. Use cookies for client-side data storage

Conclusion:

Viva Questions:

1. Which function is used to connect to database?
2. Which function is used to set a cookie?
3. Which function is used to start a session?

For Faculty use:

Correction Parameters	Formative Assessment[40%]	Timely Completion of Practical[40%]	Attendance Learning Attitude[20%]

Experiment No.—10: jQuery

Aim: Design a webpage with some jQuery animation effects.

Tools & Technologies used: VS Code, HTML, jQuery

Learning Objectives:

1. Learn to apply jQuery animations on web pages
2. Understand basic jQuery effects and interactions

Theory:

Introduction to jQuery:

- jQuery is a fast, small, and feature-rich JavaScript library.
- It simplifies HTML document traversal, event handling, and animation effects.
- It allows developers to create interactive and dynamic web pages with less code.

Basic jQuery Animation Methods:

- hide() and show(): Used to hide and display HTML elements.
- fadeIn() and fadeOut(): Used to smoothly fade an element in or out.
- slideUp() and slideDown(): Used to slide an element up or down, showing or hiding it.
- animate(): Used for custom animations by changing CSS properties over time.

Creating Animation Effects:

- jQuery allows animation of CSS properties like width, height, margin, opacity, etc.
- Example: $('#box').animate({width: '200px'}, 1000); (animates width over 1 second).
- Multiple animations can be queued to run in sequence.

Chaining Animations:

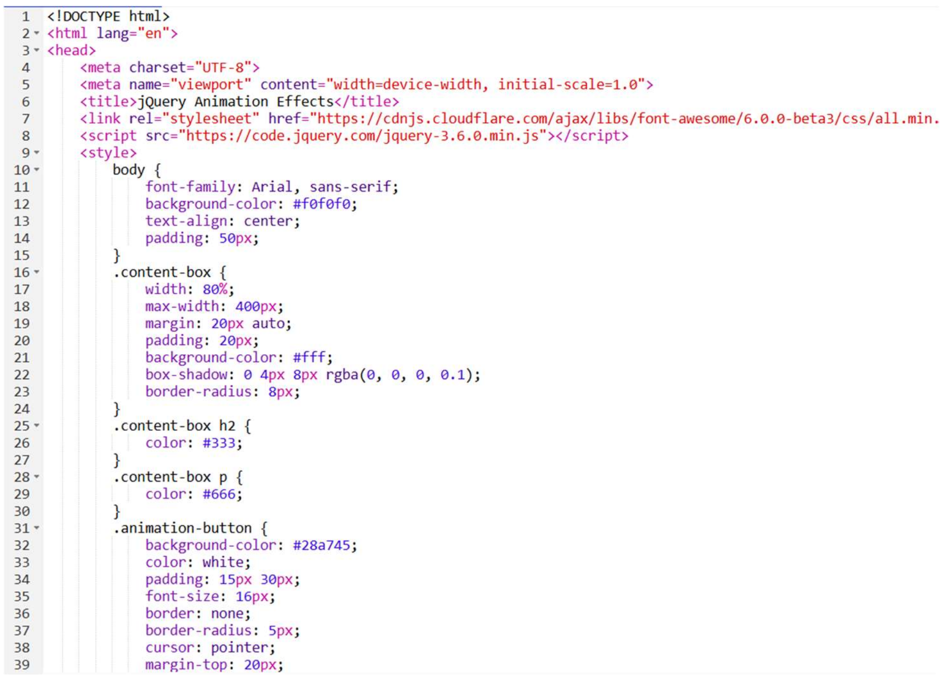
- jQuery animations can be chained together to run multiple effects sequentially.
- Example: $('#box').fadeIn().animate({width: '200px'}).fadeOut();
- This makes it easy to create complex animations without additional callbacks.

Event-based Animations:

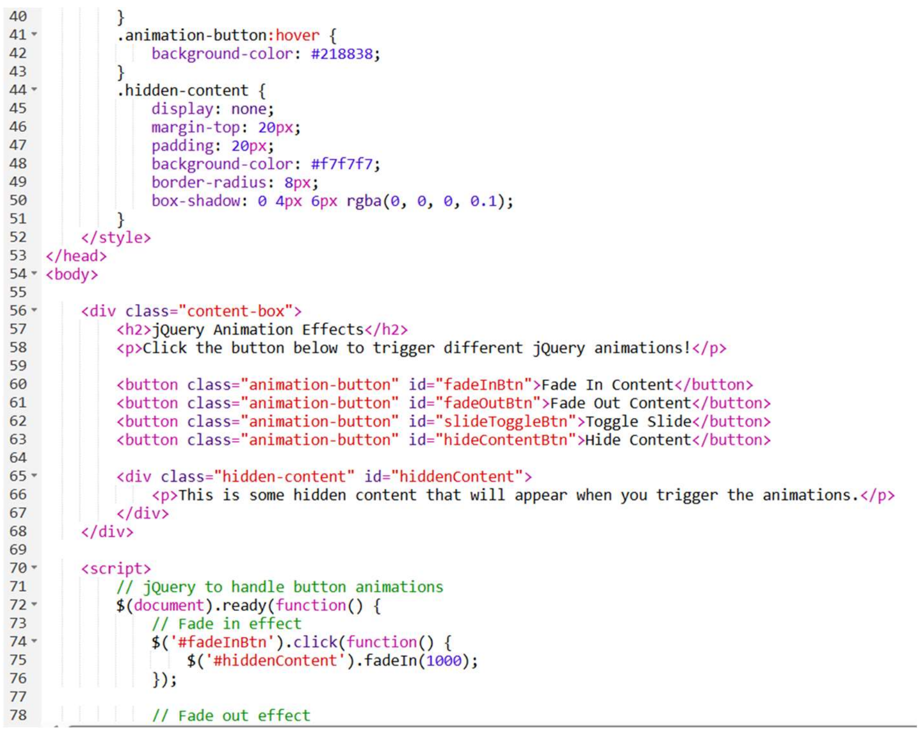
- jQuery animations are often triggered by events such as click(), hover(), or mouseover().
- Example: $('#button').click(function(){ $('#box').fadeOut(); });

- This adds interactivity, making animations responsive to user actions.

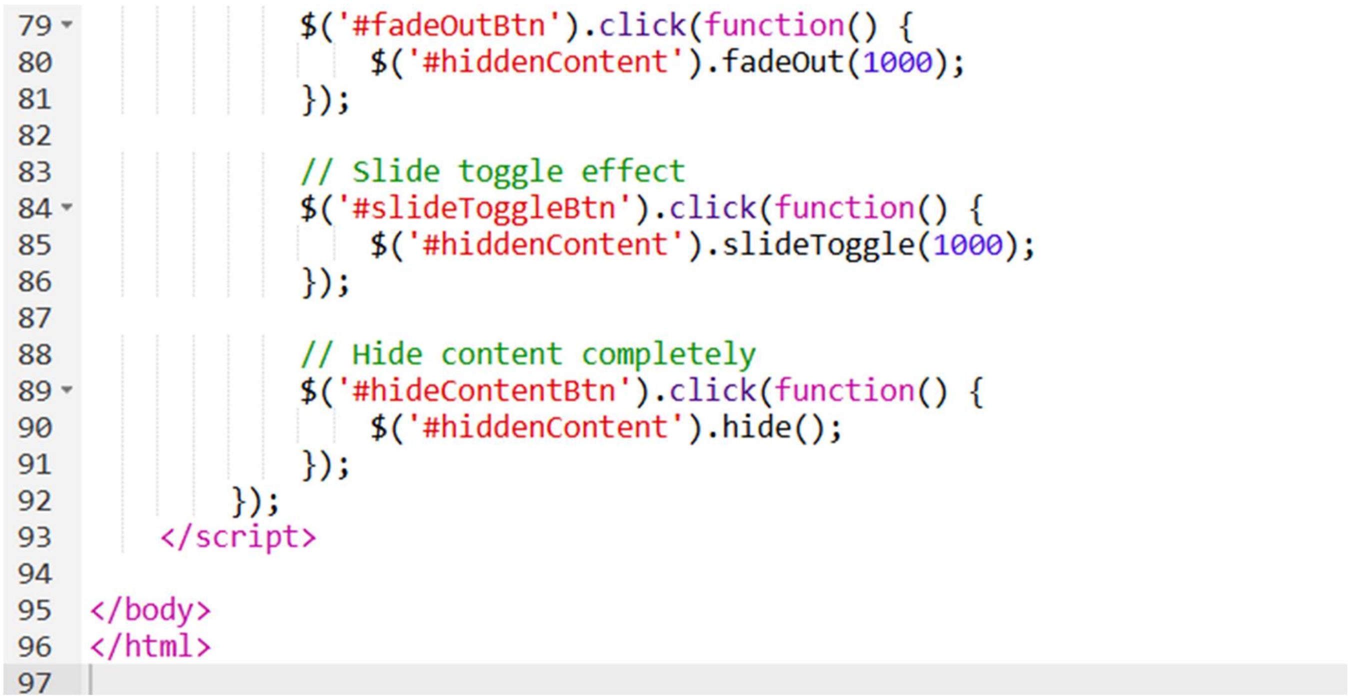
Code:



```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>jQuery Animation Effects</title>
    <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/6.0.0-beta3/css/all.min.
    <script src="https://code.jquery.com/jquery-3.6.0.min.js"></script>
    <style>
        body {
            font-family: Arial, sans-serif;
            background-color: #f0f0f0;
            text-align: center;
            padding: 50px;
        }
        .content-box {
            width: 80%;
            max-width: 400px;
            margin: 20px auto;
            padding: 20px;
            background-color: #fff;
            box-shadow: 0 4px 8px rgba(0, 0, 0, 0.1);
            border-radius: 8px;
        }
        .content-box h2 {
            color: #333;
        }
        .content-box p {
            color: #666;
        }
        .animation-button {
            background-color: #28a745;
            color: white;
            padding: 15px 30px;
            font-size: 16px;
            border: none;
            border-radius: 5px;
            cursor: pointer;
            margin-top: 20px;
```

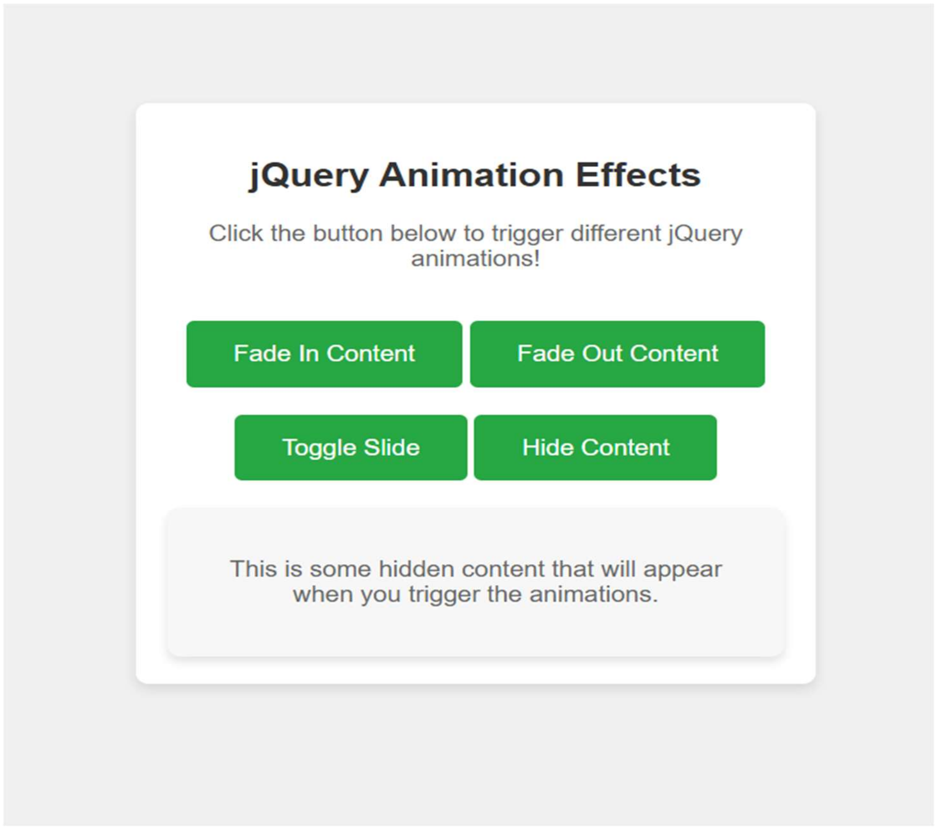


```
        }
        .animation-button:hover {
            background-color: #218838;
        }
        .hidden-content {
            display: none;
            margin-top: 20px;
            padding: 20px;
            background-color: #f7f7f7;
            border-radius: 8px;
            box-shadow: 0 4px 6px rgba(0, 0, 0, 0.1);
        }
    </style>
</head>
<body>

    <div class="content-box">
        <h2>jQuery Animation Effects</h2>
        <p>Click the button below to trigger different jQuery animations!</p>

        <button class="animation-button" id="fadeInBtn">Fade In Content</button>
        <button class="animation-button" id="fadeOutBtn">Fade Out Content</button>
        <button class="animation-button" id="slideToggleBtn">Toggle Slide</button>
        <button class="animation-button" id="hideContentBtn">Hide Content</button>

        <div class="hidden-content" id="hiddenContent">
            <p>This is some hidden content that will appear when you trigger the animations.</p>
        </div>
    </div>

    <script>
        // jQuery to handle button animations
        $(document).ready(function() {
            // Fade in effect
            $('#fadeInBtn').click(function() {
                $('#hiddenContent').fadeIn(1000);
            });

            // Fade out effect
```



```
            $('#fadeOutBtn').click(function() {
                $('#hiddenContent').fadeOut(1000);
            });

            // Slide toggle effect
            $('#slideToggleBtn').click(function() {
                $('#hiddenContent').slideToggle(1000);
            });

            // Hide content completely
            $('#hideContentBtn').click(function() {
                $('#hiddenContent').hide();
            });
        });
    </script>

</body>
</html>
```

Output:



jQuery Animation Effects

Click the button below to trigger different jQuery animations!

This is some hidden content that will appear when you trigger the animations.

Learning Outcomes:

1. Create interactive web elements using jQuery
2. Implement animation effects with jQuery

Course Outcomes:

1. Use jQuery to animate webpage elements
2. Enhance web pages with jQuery interactions

Conclusion:

Viva Questions:

1. Which method is used to hide an element in jQuery?
2. Which function is used to animate an element?
3. What is the jQuery selector symbol?

For Faculty use:

Correction Parameters	Formative Assessment[40%]	Timely Completion of Practical[40%]	Attendance Learning Attitude[20%]