Theory-1

Use of Basic Tags

❖ **Document Structure Tags:**	1. `<!DOCTYPE html>`: Defines the document type and version of HTML. 2. `<html>`: The root element that contains all other HTML elements. 3. `<head>`: Contains meta-information about the document, like `<title>` and links to stylesheets. 4. `<body>`: Contains the content of the document that is visible to users.
❖ **Content Sectioning Tags:**	1. `<header>`: Represents introductory content or a set of navigational links. 2. `<nav>`: Defines a set of navigation links. 3. `<section>`: Defines a section in a document. 4. `<article>`: Represents independent, self-contained content. 5. `<aside>`: Contains content loosely related to the page content. 6. `<footer>`: Defines the footer for a document or section.
❖ **Text Content Tags:**	1. `<h1>` to `<h6>`: Represent headings, `<h1>` being the highest level. 2. `<p>`: Defines a paragraph. 3. ` `: Inserts a single line break. 4. `<hr>`: Represents a thematic break between paragraph-level elements.
❖ **Inline Text Semantics:**	1. `<strong>`: Indicates strong importance, seriousness, or urgency. 2. `<mark>`: Highlights text. 3. `<small>`: Indicates small print. 4. `<sub>`: Defines subscripted text. 5. `<sup>`: Defines superscripted text.
❖ **Links and Resources:**	1. `<a>`: Defines a hyperlink. 2. `<img>`: Embeds an image into the document.
❖ **Lists:**	1. `<ul>`: Defines an unordered list. 2. `<ol>`: Defines an ordered list. 3. `<li>`: Defines a list item.
❖ **Tables:**	1. `<table>`: Defines a table. 2. `<tr>`: Defines a row in a table. 3. `<th>`: Defines a header cell in a table. 4. `<td>`: Defines a standard cell in a table.
❖ **Forms**	1. `<form>`: Defines an HTML form for user input. 2. `<input>`: Defines an input control. 3. `<label>`: Defines a label for an `<input>` element. 4. `<button>`: Defines a clickable button.

Practical–1: Design a web page using different text formatted text.

Source code: -

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Document</title>
</head>
<body>
  <p>This text is normal</p>
  <p><i>This text is italic</i></p>
  <p><small>This text is smaller text.</small></p>
  <p>Do not forget to buy  <mark>milk</mark></p>
  <p>My favorite color id <del>blue</del>red</p>
  <p>This is <sub>subsripted</sub>text</p>
  <p>This  is <sup>superipted</sup>text</p>
  <p><em>This is em tag</em></p>
  <p><strong>This is strong text.</strong></p>
  <p><big>This is big text</big></p>
  <p><s>This is line-throug</s></p>
  <p><u>This is underline tag</u></p>
</body>
</html>
```

Output: -

This text is normal

This text is italic

This text is smaller text.

Do not forget to buy milk

My favorite color id ~~blue~~red

This is $_{\text{subsripted}}$text

This is $^{\text{superipted}}$text

This is em tag

This is strong text.

This is big text

~~This is line-throug~~

This is underline tag

Example -1: Design a web page using Paragraph. (IT lab)

Source code: -

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Document</title>
</head>
<body>
  <header>
    <p>Karan Guatam</p>
  </header>
  <section>
    <p>Welcome to your web page!</p>
    <br>
    <p>
      Lorem ipsum dolor sit amet consectetur adipisicing elit. Odit consequatur nisi
      voluptate consectetur dignissimos veniam minima autem corporis quia recusandae.
    </p>
  </section>
  <footer>
    <p>&copy; 2024 karan gautam</p>
  </footer>
</body>
</html>
```

Output: -

Karan Guatam

Welcome to your web page!

Lorem ipsum dolor sit amet consectetur adipisicing elit. Odit consequatur nisi voluptate consectetur dignissimos veniam minima autem corporis quia recusandae.

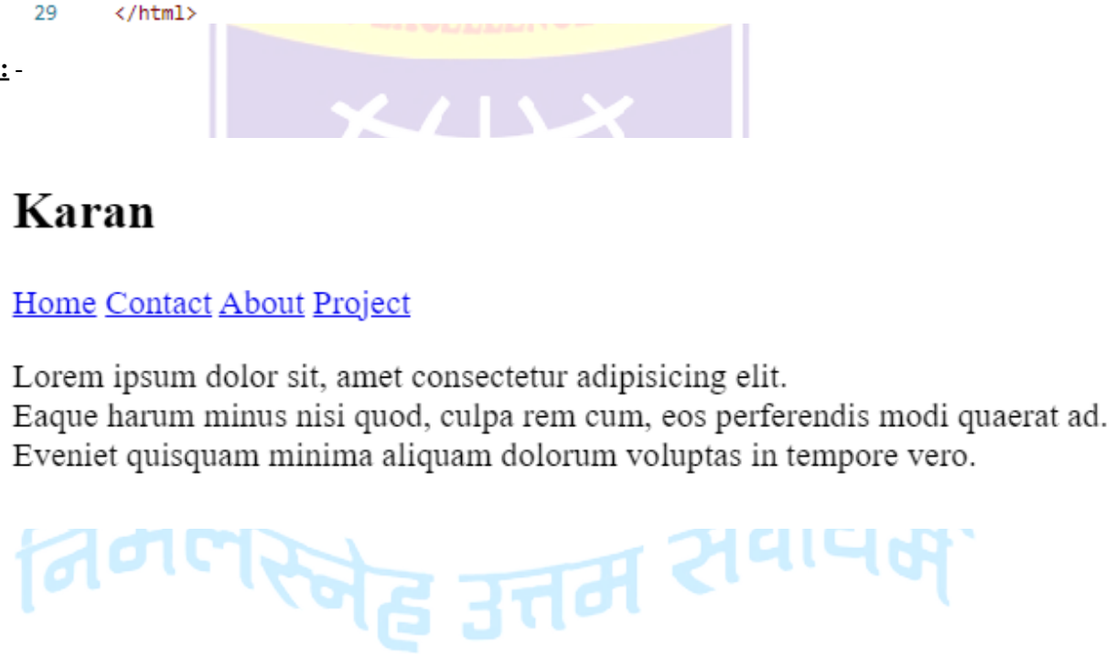
© 2024 karan gautam

Practical – 1: Design a web page with links to different pages. (IT Lab)

Source code: -

```
<!DOCTYPE html>
<html lang="en">

<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Document</title>
</head>

<body>
  <header>
    <h2>Karan</h2>
    <nav>
      <a href="#">Home</a>
      <a href="#">Contact</a>
      <a href="#">About</a>
      <a href="#">Project</a>
    </nav>
  </header>
  <p>
    Lorem ipsum dolor sit, amet consectetur adipisicing elit.
    <br>
    Eaque harum minus nisi quod, culpa rem cum, eos perferendis modi quaerat ad.
    <br>
    Eveniet quisquam minima aliquam dolorum voluptas in tempore vero.
  </p>
</body>

</html>
```

Output: -



Example - 2: Design a web page with links to different pages allow navigation between web pages

Source code: -

```
<!-- Practical - 1: Design a web page with links to different pages. (IT Lab)  -->
<!DOCTYPE html>
<html lang="en">

<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Document</title>
</head>

<body>
  <table border="1">
    <thead>
      <th>sr.</th>
      <th>Subject</th>
      <th>Link</th>
    </thead>
    <tbody>
      <tr>
        <td>1</td>
        <td>Practiacl 1</td>
        <td><a href="./p1.html">link</a></td>
      </tr>
      <tr>
        <td>2</td>
        <td>Practical 2</td>
        <td><a href="./p2.html">link</a></td>
      </tr>
    </tbody>
  </table>
</body>

</html>
```

Output: -

sr.	Subject	Link
1	Practiacl 1	link
2	Practical 2	link

Learning Outcomes: -

Course Outcomes: -

Conclusion: -

Viva Question: -

1. Define HTML tags and explain their significance in web development.
2. Describe the role of the <head> and <body> tags in an HTML document.
3. How does the <title> tag contribute to the structure and functionality of a web page?
4. Explain the purpose of the <h1> to <h6> tags in organizing content hierarchy within a webpage

.

For Faculty Use

Correction Parameters	Formative Assessment []	Timely completion of practical []	Attendance Learning Attitude []

Theory-2

Image maps, Tables, Forms and Media

In web development, various elements contribute to creating dynamic and interactive web pages. Image maps, tables, forms, and media are essential components that enhance user experience, facilitate information presentation, and enable interaction. Here's a comprehensive overview of each:

1. Image Maps:

Image maps allow developers to define clickable areas within an image, turning static visuals into interactive elements. This functionality is achieved by associating different regions of an image with specific URLs or actions. Image maps are commonly used for navigation, allowing users to click on specific parts of an image to access different sections of a website.

2. Tables:

Tables are fundamental structures for organizing and presenting data in a tabular format on web pages. HTML tables consist of rows and columns, where data is organized logically and visually. Tables are versatile and widely used for displaying various types of information, such as pricing lists, schedules, and comparison charts. With the advent of CSS, tables are often combined with styling techniques to create visually appealing layouts.

3. Forms:

Forms are interactive elements that enable users to input data and interact with web applications. HTML forms consist of input fields, checkboxes, radio buttons, dropdown menus, and buttons, among others. Forms facilitate various user interactions, including user authentication, data submission, and user feedback. Form data can be processed on the server-side using scripting languages like PHP, Python, or JavaScript.

4. Media:

Media elements, including images, audio, video, and animations, enrich the content of web pages and enhance user engagement. HTML provides tags such as <img> for images, <audio> for audio files, <video> for video content, and <iframe> for embedding external content like maps or social media feeds. Media elements are integrated into web pages to convey information, evoke emotions, and provide interactive experiences.

Conclusion:

In conclusion, image maps, tables, forms, and media are integral components of web development, each serving specific purposes in creating dynamic and interactive web experiences. Understanding how to effectively utilize these elements empowers developers to create engaging and user-friendly websites that cater to diverse user needs and preferences. As web technologies continue to evolve, leveraging these elements in innovative ways contributes to the evolution of web design and user experience standards.

Practical – 2: Design a web page with Image maps. (IT Lab)

Source code: -

```
<!DOCTYPE html>
<html lang="en">

<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Image Maps</title>
</head>

<body>
  <h1>Image Maps</h1>
  <img src="./Img.jpeg" usemap="#image">
  <map name="image">
    <area target="_blank" alt="monitor" title="monitor" href="https://en.wikipedia.org/wiki/Computer_monitor"
      coords="2,6,124,78" shape="rect">
    <area target="" alt="cpu" title="cpu" href="https://en.wikipedia.org/wiki/Central_processing_unit"
      coords="162,8,190,0,243,53,242,145,196,141,158,144" shape="poly">
    <area target="" alt="keyboard" title="keyboard" href="https://en.wikipedia.org/wiki/Computer_keyboard"
      coords="6,121,115,156" shape="rect">
    <area target="" alt="mouse" title="mouse" href="https://en.wikipedia.org/wiki/Computer_mouse"
      coords="119,132,130,127,138,152,126,158,122,145" shape="poly">
  </map>
</body>

</html>
```

Output: -

Example – 1: Design a web page with server-side image maps. (Homework)

Source code: -

```
<!DOCTYPE html>
<html>
<head>
<title>Client-Side Image Map Example</title>
</head>
<body>

<h2>Interactive Image Map</h2>
<p>Click on the computer, phone, or the cup of coffee in the image:</p>

<!-- Replace 'workplace.jpg' with the actual image URL -->
<img src="https://www.bing.com/th?id=OIP.irv-95KJakV30GK-6e8_0gHaE6" alt="Workplace"
usemap="#workmap" width="400" height="379">

<map name="workmap">
  <!-- Define the clickable areas with their coordinates and links -->
  <area shape="rect" coords="34,44,270,350" alt="Computer" href="computer.htm">
  <area shape="rect" coords="290,172,333,250" alt="Phone" href="phone.htm">
  <area shape="circle" coords="337,300,44" alt="Coffee" href="coffee.htm">
</map>

</body>
</html>
```

Output: -

Practical- 2: Design a web page with different tables. (IT Lab)

Source code: -

```html
<!DOCTYPE html>
<html>
<head>
<title>Web Page with Different Tables</title>
<style>
  /* Add some basic styling to your tables */
  table {
    width: 100%;
    border-collapse: collapse;
    margin-bottom: 20px;
  }
  th, td {
    border: 1px solid #ddd;
    padding: 8px;
    text-align: left;
  }
  th {
    background-color: #f2f2f2;
  }
</style>
</head>
<body>

<h2>Product Table</h2>
<table>
  <tr>
    <th>Product ID</th>
    <th>Name</th>
    <th>Price</th>
  </tr>
  <tr>
    <td>1</td>
    <td>Laptop</td>
    <td>$999</td>
  </tr>
  <tr>
    <td>2</td>
    <td>Smartphone</td>
    <td>$499</td>
  </tr>
</table>

<h2>Employee Table</h2>
<table>
  <tr>
    <th>Employee ID</th>
    <th>Name</th>
    <th>Department</th>
  </tr>
  <tr>
    <td>A001</td>
    <td>John Doe</td>
    <td>Marketing</td>
  </tr>
  <tr>
    <td>A002</td>
    <td>Jane Smith</td>
    <td>Development</td>
  </tr>
</table>

</body>
</html>
```

Output: -

Product Table

Product ID	Name	Price
1	Laptop	$999
2	Smartphone	$499

Employee Table

Employee ID	Name	Department
A001	John Doe	Marketing
A002	Jane Smith	Development

Learning Outcomes: -

Course Outcomes: -

Conclusion: -

Viva Question: -

1. What are image maps, and how are they used in web development?

2. How do tables help organize information on webpages? Can you give an example?

3. What is the purpose of web forms, and why are they important for user interaction?

5. How does media contribute to improving the user experience on a webpage, and what are some examples of media types used in web design?

For Faculty Use

Correction Parameters	Formative Assessment []	Timely completion of practical []	Attendance Learning Attitude []

Theory-3

Java Script

JavaScript, the omnipresent language of the web, embodies the essence of modern interactivity, seamlessly bridging the gap between static content and dynamic user experiences. Its concise syntax and versatile capabilities empower developers to craft immersive applications that resonate with users across diverse platforms and devices.

At its core, JavaScript is a high-level, interpreted programming language renowned for its lightweight footprint and ubiquitous support across web browsers. Its dynamic nature enables developers to manipulate HTML and CSS, dynamically alter page content, and respond to user interactions in real-time, fostering a dynamic and engaging browsing experience.

One of JavaScript's fundamental paradigms lies in its event-driven architecture, where actions such as clicks, scrolls, and keyboard inputs trigger corresponding event handlers. This paradigm fosters responsive and interactive web applications by decoupling user actions from application logic, enabling developers to create intuitive interfaces that react seamlessly to user input.

Moreover, JavaScript's asynchronous nature empowers developers to execute non-blocking operations, ensuring smooth and uninterrupted user experiences. Techniques such as callbacks, promises, and async/await enable developers to handle tasks such as fetching data from remote servers, processing large datasets, and executing animations without freezing the user interface.

The versatility of JavaScript extends beyond the confines of web browsers, as evidenced by its adoption in server-side development with platforms like Node.js. By leveraging JavaScript on both the client and server sides, developers can achieve unparalleled synergy, sharing code between frontend and backend components and streamlining development workflows.

Furthermore, the vibrant ecosystem surrounding JavaScript, encompassing frameworks like React, Angular, and Vue.js, empowers developers to build scalable, modular, and maintainable applications with ease. These frameworks provide robust abstractions, reusable components, and efficient state management mechanisms, accelerating development and fostering code reusability.

However, with great power comes great responsibility. JavaScript's flexibility can also pose challenges, such as browser compatibility issues, performance bottlenecks, and security vulnerabilities. It is imperative for developers to adhere to best practices, adopt modern development tools, and stay abreast of emerging trends to mitigate these challenges effectively.

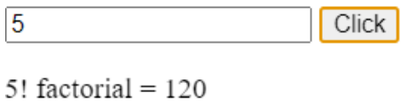
In conclusion, JavaScript epitomizes the cornerstone of modern web development, enabling developers to craft dynamic, interactive, and responsive applications that transcend traditional boundaries. With its rich ecosystem, vibrant community, and relentless innovation, JavaScript continues to shape the digital landscape, empowering developers to transform ideas into reality and revolutionize the way we interact with the web.

Practical – 7: Using JavaScript design, a web page that prints factorial of given number. (IT Lab)

Source code: -

```
<!DOCTYPE html>
<html lang="en">

<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>factorial</title>
</head>

<body>
  <input type="text" class="input" placeholder="Enter your number">
  <button class="btn">Click</button>
  <p class="Result"></p>

  <script>
    const input = document.querySelector(".input");
    const btn = document.querySelector(".btn");
    const Result = document.querySelector(".Result");

    btn.addEventListener("click", () => {
      const inputValue = input.value;
      let fact = 1;
      for (let i = 1; i <= inputValue; i++) {
        fact *= i;
      }

      Result.textContent = `${inputValue}! factorial = ` + fact;
    })

  </script>
</body>

</html>
```

Output: -



5! factorial = 120

Example -1: Write a JavaScript program to display all the prime numbers between 1 and 100. (IT Lab)

Source code: -

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

  <script>
      // Function to check if a number is prime
      function isPrime(num) {
          if (num <= 1) {
              return false;
          }
          for (let i = 2; i <= Math.sqrt(num); i++) {
              if (num % i === 0) {
                  return false;
              }
          }
          return true;
      }

      // Function to display prime numbers between 1 and 100
      function displayPrimes() {
          let primes = [];
          for (let i = 2; i <= 100; i++) {
              if (isPrime(i)) {
                  primes.push(i);
              }
          }
          document.write("<h2>Prime Numbers between 1 and 100:</h2>");
          document.write(primes.join(", "));
      }

      // Call the function to display prime numbers
      displayPrimes();
  </script>

  </body>
  </html>

</body>
</html>
```

Output: -

Prime Numbers between 1 and 100:

2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47,
53, 59, 61, 67, 71, 73, 79, 83, 89, 97

Example – 2: Using JavaScript design, a web page that prints Fibonacci series. (Homework)

Source code: -

```
<!DOCTYPE html>
<html>

<head>
    <title>Fibonacci Series</title>
</head>

<body>

    <h2>Fibonacci Series:</h2>
    <p id="fibonacciSeries"></p>

    <script>
        // Function to generate Fibonacci series
        function generateFibonacciSeries(limit) {
            let fibonacciSeries = [0, 1];

            for (let i = 2; i < limit; i++) {
                let nextNumber = fibonacciSeries[i - 1] + fibonacciSeries[i - 2];
                fibonacciSeries.push(nextNumber);
            }

            return fibonacciSeries;
        }

        // Function to display Fibonacci series on the web page
        function displayFibonacciSeries() {
            const limit = 10; // Change this to generate Fibonacci series up to a different limit
            const fibonacciSeries = generateFibonacciSeries(limit);
            const fibonacciSeriesElement = document.getElementById("fibonacciSeries");

            fibonacciSeriesElement.textContent = fibonacciSeries.join(", ");
        }

        // Call the function to display Fibonacci series
        displayFibonacciSeries();
    </script>

</body>

</html>
```

Output: -

Fibonacci Series:

0, 1, 1, 2, 3, 5, 8, 13, 21, 34

Example – 3: Design a form and validate username, password and contact number on the form using JavaScript. (Homework)

Source code: -

```
<!DOCTYPE html>
<html>

<head>
    <title>Form Validation</title>
    <style>
        .error {
            color: red;
        }
    </style>
</head>

<body>

    <h2>Registration Form</h2>
    <form id="registrationForm" onsubmit="return validateForm()">
        <label for="username">Username:</label><br>
        <input type="text" id="username" name="username"><span id="usernameError" class="error"></span><br>

        <label for="password">Password:</label><br>
        <input type="password" id="password" name="password"><span id="passwordError" class="error"></span><br>

        <label for="contact">Contact Number:</label><br>
        <input type="text" id="contact" name="contact"><span id="contactError" class="error"></span><br>

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

    <script>
        function validateForm() {
            // Reset error messages
            document.getElementById("usernameError").textContent = "";
            document.getElementById("passwordError").textContent = "";
            document.getElementById("contactError").textContent = "";

            let isValid = true;

            // Username validation
            let username = document.getElementById("username").value;
            if (username.trim() === "") {
                document.getElementById("usernameError").textContent = "Username is required";
                isValid = false;
            }

            // Password validation
            let password = document.getElementById("password").value;
            if (password.trim() === "") {
                document.getElementById("passwordError").textContent = "Password is required";
                isValid = false;
            }

            // Contact number validation
            let contact = document.getElementById("contact").value;
            if (contact.trim() === "") {
                document.getElementById("contactError").textContent = "Contact number is required";
                isValid = false;
            } else if (!/^\d{10}$/.test(contact)) {
                document.getElementById("contactError").textContent = "Contact number must be 10 digits";
                isValid = false;
            }

            return isValid;
        }
    </script>

</body>

</html>
```

Output: -

Registration Form

Username:
ghfgh
Password:
•••
Contact Number:
kjl Contact number must be 10 digits
Submit

Learning Outcomes: -

Course Outcomes: -

Conclusion: -

Viva Question: -

1. What is JavaScript's event-driven architecture, and how does it contribute to creating interactive web applications?

2. How does JavaScript handle asynchronous tasks, and why is this capability crucial for building responsive user interfaces?

3. Can you explain the concept of code reusability in JavaScript frameworks, and how it simplifies the development process?

4. What are some common challenges associated with JavaScript development, and how can developers address them effectively to ensure robust and secure applications?

For Faculty Use

Correction Parameters	Formative Assessment []	Timely completion of practical []	Attendance Learning Attitude []

Theory-4

Control and looping statements and Java Script references

Control and looping statements are pivotal components of JavaScript, offering developers powerful tools to manage program execution and iterate through data structures. These constructs enable developers to craft dynamic, responsive, and efficient code, shaping the behaviour of applications in diverse scenarios.

Conditional Statements:

JavaScript features conditional statements like `if`, `else if`, and `else`, facilitating decision-making within code blocks. Conditional statements evaluate expressions and execute specific branches of code based on the result, enabling developers to handle varying conditions and respond accordingly.

Example:

```
let num = 10;
if (num > 0) {
console.log("Positive number");
} else if (num < 0) {
console.log("Negative number");
} else {
console.log("Zero");
}
```

Looping Statements:

JavaScript offers versatile looping constructs such as `for`, `while`, and `do-while`, enabling developers to iterate over arrays, collections, or perform repetitive tasks. These looping statements facilitate efficient data traversal and manipulation, contributing to streamlined algorithmic logic.

Example:

```
// Using for loop to iterate over an array
let fruits = ["apple", "banana", "orange"];
for (let i = 0; i < fruits.length; i++) {
   console.log(fruits[i]);
}

// Using while loop to print numbers from 1 to 5
let num = 1;
while (num <= 5) {
   console.log(num);
   num++;
}
```

Control Flow Statements: JavaScript includes control flow statements like `break` and `continue` to influence loop behaviour. The `break` statement terminates loop execution prematurely, while `continue` skips the current iteration and proceeds to the next iteration, offering finer control over loop execution.

Example:

```
// Using break statement to exit loop early
for (let i = 0; i < 10; i++) {
   if (i === 5) {
      break;
   }
   console.log(i);
}

// Using continue statement to skip even numbers
for (let i = 0; i < 10; i++) {
   if (i % 2 === 0) {
      continue;
   }
   console.log(i);
}
```

JavaScript References: To delve deeper into control and looping statements in JavaScript, developers can turn to authoritative references such as the Mozilla Developer Network (MDN) documentation. These resources provide comprehensive explanations, syntax details, and illustrative examples, empowering developers to grasp concepts effectively and apply them proficiently in real-world scenarios.

By mastering control and looping statements and leveraging reliable references, developers can enhance their coding proficiency, optimize program efficiency, and create dynamic applications that deliver exceptional user experiences. With JavaScript's versatile capabilities and robust reference materials, the possibilities for innovation and creativity in web development are boundless.

Practical – 8: Design a web page demonstrating different conditional statements. (IT Lab)

Source code: -

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Conditional Statements Demo</title>
    <style>
        body {
            font-family: Arial, sans-serif;
            margin: 20px;
        }
        .container {
            max-width: 800px;
            margin: 0 auto;
        }
        .example {
            margin-bottom: 20px;
            border: 1px solid #ccc;
            padding: 10px;
        }
        h2 {
            margin-top: 0;
        }
        p {
            margin-top: 5px;
        }
    </style>
</head>
<body>
    <div class="container">
        <h1>Conditional Statements Demo</h1>

        <div class="example">
            <h2>Example 1: Using if Statement</h2>
            <p id="ifStatementOutput"></p>
        </div>

        <div class="example">
            <h2>Example 2: Using if...else Statement</h2>
            <p id="ifElseStatementOutput"></p>
        </div>

        <div class="example">
            <h2>Example 3: Using if...else if...else Statement</h2>
            <p id="ifElseIfStatementOutput"></p>
        </div>

        <script>
            // Example 1: Using if Statement
            let x = 10;
            let ifOutput = "";
            if (x > 0) {
                ifOutput = "x is a positive number.";
            }
            document.getElementById("ifStatementOutput").innerHTML = ifOutput;

            // Example 2: Using if...else Statement
            let y = -5;
            let ifElseOutput = "";
            if (y > 0) {
                ifElseOutput = "y is a positive number.";
            } else {
                ifElseOutput = "y is a non-positive number.";
            }
            document.getElementById("ifElseStatementOutput").innerHTML = ifElseOutput;

            // Example 3: Using if...else if...else Statement
            let z = 0;
            let ifElseIfOutput = "";
            if (z > 0) {
                ifElseIfOutput = "z is a positive number.";
            } else if (z < 0) {
                ifElseIfOutput = "z is a negative number.";
            } else {
                ifElseIfOutput = "z is zero.";
            }
            document.getElementById("ifElseIfStatementOutput").innerHTML = ifElseIfOutput;
        </script>
    </div>
</body>
</html>
```

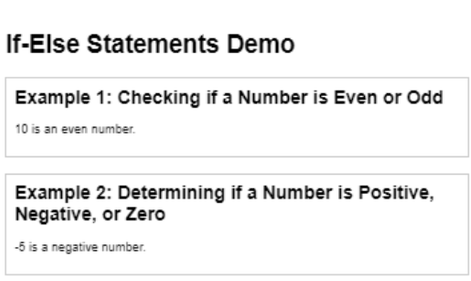
Output: -

Example – 1: Design a web page demonstrating if-else statements. (IT Lab)

Source code: -

```html
<!DOCTYPE html>
<html lang="en">

<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>If-Else Statements Demo</title>
  <style>
    body {
      font-family: Arial, sans-serif;
      margin: 20px;
    }

    .container {
      max-width: 600px;
      margin: 0 auto;
    }

    .example {
      margin-bottom: 20px;
      border: 1px solid #ccc;
      padding: 10px;
    }

    h2 {
      margin-top: 0;
    }

    p {
      margin-top: 5px;
    }
  </style>
</head>

<body>
  <div class="container">
    <h1>If-Else Statements Demo</h1>

    <div class="example">
      <h2>Example 1: Checking if a Number is Even or Odd</h2>
      <p id="evenOddOutput"></p>
    </div>

    <div class="example">
      <h2>Example 2: Determining if a Number is Positive, Negative, or Zero</h2>
      <p id="posNegZeroOutput"></p>
    </div>

    <script>
      // Example 1: Checking if a Number is Even or Odd
      let num1 = 10;
      let evenOddOutput = "";
      if (num1 % 2 === 0) {
        evenOddOutput = num1 + " is an even number.";
      } else {
        evenOddOutput = num1 + " is an odd number.";
      }
      document.getElementById("evenOddOutput").innerHTML = evenOddOutput;

      // Example 2: Determining if a Number is Positive, Negative, or Zero
      let num2 = -5;
      let posNegZeroOutput = "";
      if (num2 > 0) {
        posNegZeroOutput = num2 + " is a positive number.";
      } else if (num2 < 0) {
        posNegZeroOutput = num2 + " is a negative number.";
      } else {
        posNegZeroOutput = num2 + " is zero.";
      }
      document.getElementById("posNegZeroOutput").innerHTML = posNegZeroOutput;
    </script>
  </div>
</body>

</html>
```

Output: -

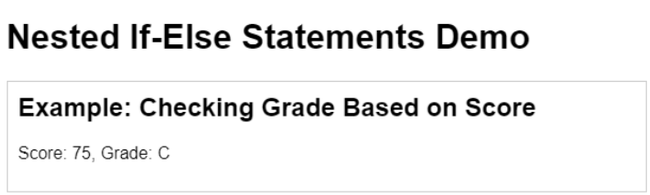


Example – 2: Design a web page demonstrating nested if-else conditional statements. (Homework)

Source code: -

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Nested If-Else Statements Demo</title>
    <style>
        body {
            font-family: Arial, sans-serif;
            margin: 20px;
        }
        .container {
            max-width: 600px;
            margin: 0 auto;
        }
        .example {
            margin-bottom: 20px;
            border: 1px solid #ccc;
            padding: 10px;
        }
        h2 {
            margin-top: 0;
        }
        p {
            margin-top: 5px;
        }
    </style>
</head>
<body>
    <div class="container">
        <h1>Nested If-Else Statements Demo</h1>

        <div class="example">
            <h2>Example: Checking Grade Based on Score</h2>
            <p id="gradeOutput"></p>
        </div>

        <script>
            // Example: Checking Grade Based on Score
            let score = 75;
            let gradeOutput = "";

            if (score >= 0 && score <= 100) {
                if (score >= 90) {
                    gradeOutput = "A";
                } else if (score >= 80) {
                    gradeOutput = "B";
                } else if (score >= 70) {
                    gradeOutput = "C";
                } else if (score >= 60) {
                    gradeOutput = "D";
                } else {
                    gradeOutput = "F";
                }
            } else {
                gradeOutput = "Invalid score!";
            }

            document.getElementById("gradeOutput").innerHTML = "Score: " + score + ", Grade: " + gradeOutput;
        </script>
    </div>
</body>
</html>
```

Output: -



Example – 3: Using JavaScript design, a web page that prints factorial of given number. (Homework)

Source code: -

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Factorial Calculator</title>
    <style>
        body {
            font-family: Arial, sans-serif;
            margin: 20px;
        }
        .container {
            max-width: 600px;
            margin: 0 auto;
        }
        .form-group {
            margin-bottom: 10px;
        }
        label {
            font-weight: bold;
        }
        button {
            padding: 5px 10px;
            cursor: pointer;
        }
        #result {
            margin-top: 20px;
        }
    </style>
</head>
<body>
    <div class="container">
        <h1>Factorial Calculator</h1>

        <div class="form-group">
            <label for="number">Enter a number:</label>
            <input type="number" id="number" min="0" step="1" required>
        </div>

        <button onclick="calculateFactorial()">Calculate Factorial</button>

        <div id="result"></div>

        <script>
            function calculateFactorial() {
                let number = parseInt(document.getElementById("number").value);
                let result = 1;

                if (number < 0) {
                    document.getElementById("result").innerText = "Factorial is not defined for negative numbers.";
                } else if (number === 0) {
                    document.getElementById("result").innerText = "Factorial of 0 is 1.";
                } else {
                    for (let i = 1; i <= number; i++) {
                        result *= i;
                    }
                    document.getElementById("result").innerText = "Factorial of " + number + " is " + result + ".";
                }
            }
        </script>
    </div>
</body>
</html>
```

Output: -

Example -4: Write a JavaScript program to display all the prime numbers between 1 and 100. (Homework)

Source code: -

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Prime Number Finder</title>
    <style>
        body {
            font-family: Arial, sans-serif;
            margin: 20px;
        }
        .container {
            max-width: 600px;
            margin: 0 auto;
        }
        #primeNumbers {
            margin-top: 10px;
        }
    </style>
</head>
<body>
    <div class="container">
        <h1>Prime Number Finder</h1>

        <p>Prime numbers between 1 and 100:</p>

        <ul id="primeNumbers"></ul>

        <script>
            function isPrime(num) {
                for (let i = 2; i <= Math.sqrt(num); i++) {
                    if (num % i === 0) {
                        return false;
                    }
                }
                return num > 1;
            }

            function displayPrimeNumbers() {
                let primeNumbersList = document.getElementById("primeNumbers");
                primeNumbersList.innerHTML = ""; // Clear previous list

                for (let i = 1; i <= 100; i++) {
                    if (isPrime(i)) {
                        let listItem = document.createElement("li");
                        listItem.textContent = i;
                        primeNumbersList.appendChild(listItem);
                    }
                }
            }

            displayPrimeNumbers(); // Call the function to display prime numbers on page load
        </script>
    </div>
</body>
</html>
```

Output:

Learning Outcomes: -

Course Outcomes: -

Conclusion: -

Viva Question: -

1. How would you describe the purpose of control and looping statements in JavaScript?

2. Can you provide an example of how you would use a conditional statement like 'if-else' in JavaScript to control program flow?

3. What is the significance of loop constructs like 'for', 'while', and 'do-while' in JavaScript, and how do they differ in their usage?

4. Why is it essential for developers to refer to JavaScript documentation and reliable resources when working with control and looping statements??

For Faculty Use

Correction Parameters	Formative Assessment []	Timely completion of practical []	Attendance Learning Attitude []

Theory-5

Basic PHP I

PHP, which stands for "Hypertext Preprocessor," is a widely-used server-side scripting language primarily designed for web development. It seamlessly integrates with HTML, allowing developers to create dynamic and interactive web pages with ease. In this one-page theory, we'll delve into the fundamentals of PHP, its key features, and its role in modern web development.

1. Syntax and Structure: PHP code is embedded within HTML documents and typically enclosed within `<?php` and `?>` tags. This allows PHP to execute server-side, generating dynamic content before the page is sent to the client's browser. PHP statements end with a semicolon (;), and comments can be added using `//` for single-line comments and `/* */` for multi-line comments.

2. Variables and Data Types: PHP supports various data types, including integers, floats, strings, booleans, arrays, and objects. Variables in PHP start with a dollar sign ($) followed by the variable name. They are loosely typed, meaning you don't need to declare the data type explicitly. Variable scope can be global or local, depending on where they are defined.

3. Control Structures: PHP offers a range of control structures for decision-making and looping, similar to other programming languages. These include `if`, `else`, `elseif` for conditional statements, `for`, `while`, `do-while` for loops, and `switch` for multi-way branching. These structures allow developers to control the flow of execution based on conditions and iterate over data efficiently.

4. Functions: Functions in PHP allow developers to encapsulate reusable pieces of code. They are defined using the `function` keyword followed by the function name and parameters. Functions can return values using the `return` statement and can accept any number of arguments. PHP also provides built-in functions for common tasks, such as string manipulation, mathematical operations, and database interactions.

5. File Handling and I/O Operations: PHP enables file handling operations, allowing developers to read from and write to files on the server's file system. This includes functions like `fopen()`, `fwrite()`, `fread()`, and `fclose()` for basic file manipulation. Additionally, PHP supports various input/output operations, such as reading form data, processing user input, and sending HTTP headers.

6. Database Integration:PHP is commonly used in conjunction with database management systems like MySQL, PostgreSQL, and SQLite for building dynamic web applications. It provides built-in functions and extensions for connecting to databases, executing SQL queries, and fetching results. This enables developers to create interactive websites that store and retrieve data from backend databases efficiently.

7. Security Considerations: When developing web applications with PHP, it's crucial to consider security best practices to protect against common vulnerabilities such as SQL injection, cross-site scripting (XSS), and cross-site request forgery (CSRF). Techniques like input validation, parameterized queries, and sanitization help mitigate these risks and ensure the integrity and confidentiality of user data.

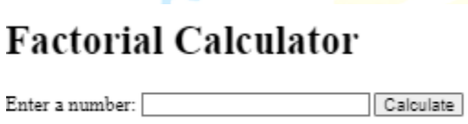
Conclusion: In summary, PHP serves as a powerful tool for server-side web development, offering a versatile and feature-rich environment for building dynamic and interactive websites. Its seamless integration with HTML, extensive library of functions, and robust database support make it a popular choice among developers for creating a wide range of web applications. By mastering the basics of PHP, developers can unlock endless possibilities for crafting innovative and engaging online experiences.

Practical –10: Write a PHP Program to accept a number from the user and print it factorial. (IT Lab)

Source code: -

```
<!DOCTYPE html>
<html>

<head>
  <title>Factorial Calculator</title>
</head>

<body>
  <h1>Factorial Calculator</h1>
  <form method="post">
    Enter a number: <input type="number" name="number" required>
    <input type="submit" name="submit" value="Calculate">
  </form>

  <?php
    function calculateFactorial($num) {
        if ($num < 0) {
            return "Factorial is not defined for negative numbers.";
        } elseif ($num == 0) {
            return "Factorial of 0 is 1.";
        } else {
            $factorial = 1;
            for ($i = 1; $i <= $num; $i++) {
                $factorial *= $i;
            }
            return "Factorial of $num is $factorial.";
        }
    }

    if (isset($_POST['submit'])) {
        $number = $_POST['number'];
        echo calculateFactorial($number);
    }
    ?>
</body>

</html>
```

Output: -



Example – 1: Write a PHP program to accept a number from the user and print whether it is prime or not. (IT Lab)

Source code: -

```php
<?php
// Function to check if a number is prime
function isPrime($num) {
    if ($num <= 1) {
        return false;
    }
    for ($i = 2; $i <= sqrt($num); $i++) {
        if ($num % $i == 0) {
            return false;
        }
    }
    return true;
}

// Accepting input from user
if(isset($_POST['number'])) {
    $number = intval($_POST['number']);
    if(isPrime($number)) {
        echo "$number is a prime number.";
    } else {
        echo "$number is not a prime number.";
    }
}
?>

<!-- HTML form to accept user input -->
<form method="post">
    Enter a number: <input type="text" name="number">
    <input type="submit" value="Check">
</form>
```

Output: -

Enter a number: [ ] Check

Example -2: Design a Web Page demonstrating all types of Style Sheets. (Homework)

Source code: -

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Style Sheets Example</title>
    <!-- Internal CSS -->
    <style>
        /* Internal CSS */
        h1 {
            color: blue;
        }
        p {
            font-style: italic;
        }
    </style>
    <!-- External CSS -->
    <link rel="stylesheet" type="text/css" href="styles.css">
</head>
<body>
    <!-- Inline CSS -->
    <h1 style="color: red;">This is a heading with inline CSS</h1>
    <p style="font-weight: bold;">This is a paragraph with inline CSS</p>
    <!-- Internal CSS -->
    <h1>This is a heading with internal CSS</h1>
    <p>This is a paragraph with internal CSS</p>
    <!-- External CSS -->
    <h1>This is a heading with external CSS</h1>
    <p>This is a paragraph with external CSS</p>
</body>
</html>
```

Output: -

This is a heading with inline CSS

This is a paragraph with inline CSS

This is a heading with internal CSS

This is a paragraph with internal CSS

This is a heading with external CSS

This is a paragraph with external CSS

Example -3: Write a JavaScript program to display all the prime numbers between 1 and 100.

(Homework)

Source code: -

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Prime Numbers</title>
    <script>
        function isPrime(num) {
            if (num <= 1) {
                return false;
            }
            for (let i = 2; i <= Math.sqrt(num); i++) {
                if (num % i === 0) {
                    return false;
                }
            }
            return true;
        }

        function displayPrimeNumbers() {
            let primeNumbers = [];
            for (let i = 2; i <= 100; i++) {
                if (isPrime(i)) {
                    primeNumbers.push(i);
                }
            }
            document.getElementById('primeNumbers').textContent = primeNumbers.join(', ');
        }
    </script>
</head>
<body>
    <h2>Prime Numbers between 1 and 100:</h2>
    <button onclick="displayPrimeNumbers()">Show Prime Numbers</button>
    <p id="primeNumbers"></p>
</body>
</html>
```

Output: -

Example -4: Write a JavaScript program to design a simple calculator. (Homework)

Source code: -

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Simple Calculator</title>
    <style>
        input[type="text"] {
            width: 100%;
            margin-bottom: 10px;
            padding: 5px;
            font-size: 16px;
        }
        input[type="button"] {
            width: 50px;
            height: 50px;
            font-size: 16px;
            margin-right: 5px;
        }
        .container {
            display: flex;
            flex-wrap: wrap;
        }
    </style>
</head>
<body>
    <h2>Simple Calculator</h2>
    <div class="container">
        <input type="text" id="result" disabled>
        <input type="button" value="1" onclick="addToInput('1')">
        <input type="button" value="2" onclick="addToInput('2')">
        <input type="button" value="3" onclick="addToInput('3')">
        <input type="button" value="+" onclick="addToInput('+')">
        <input type="button" value="4" onclick="addToInput('4')">
        <input type="button" value="5" onclick="addToInput('5')">
        <input type="button" value="6" onclick="addToInput('6')">
        <input type="button" value="-" onclick="addToInput('-')">
        <input type="button" value="7" onclick="addToInput('7')">
        <input type="button" value="8" onclick="addToInput('8')">
        <input type="button" value="9" onclick="addToInput('9')">
        <input type="button" value="*" onclick="addToInput('*')">
        <input type="button" value="C" onclick="clearInput()">
        <input type="button" value="0" onclick="addToInput('0')">
        <input type="button" value="=" onclick="calculate()">
        <input type="button" value="/" onclick="addToInput('/')">
    </div>

    <script>
        function addToInput(value) {
            document.getElementById('result').value += value;
        }

        function clearInput() {
            document.getElementById('result').value = '';
        }

        function calculate() {
            let input = document.getElementById('result').value;
            let result;
            try {
                result = eval(input);
            } catch (error) {
                result = 'Error';
            }
            document.getElementById('result').value = result;
        }
    </script>
</body>
</html>
```

Output: -

Learning Outcomes: -

Course Outcomes: -

Conclusion: -

Viva Question: -

1. What is PHP, and what does it stand for?

2. Explain the difference between echo and print statements in PHP.

3. How do you declare variables in PHP? Provide an example

4. Describe the usage of the foreach loop in PHP with a practical example.

For Faculty Use

Correction Parameters	Formative Assessment []	Timely completion of practical []	Attendance Learning Attitude []

Theory-6

Basic PHP II

PHP, which stands for "Hypertext Preprocessor," is a widely-used server-side scripting language primarily designed for web development. It seamlessly integrates with HTML, allowing developers to create dynamic and interactive web pages with ease. In this one-page theory, we'll delve into the fundamentals of PHP, its key features, and its role in modern web development.

1. Syntax and Structure: PHP code is embedded within HTML documents and typically enclosed within `<?php` and `?>` tags. This allows PHP to execute server-side, generating dynamic content before the page is sent to the client's browser. PHP statements end with a semicolon (;), and comments can be added using `//` for single-line comments and `/* */` for multi-line comments.

2. Variables and Data Types: PHP supports various data types, including integers, floats, strings, Booleans, arrays, and objects. Variables in PHP start with a dollar sign ($) followed by the variable name. They are loosely typed, meaning you don't need to declare the data type explicitly. Variable scope can be global or local, depending on where they are defined.

3. Control Structures: PHP offers a range of control structures for decision-making and looping, similar to other programming languages. These include `if`, `else`, `elseif` for conditional statements, `for`, `while`, `do-while` for loops, and `switch` for multi-way branching. These structures allow developers to control the flow of execution based on conditions and iterate over data efficiently.

4. Functions: Functions in PHP allow developers to encapsulate reusable pieces of code. They are defined using the `function` keyword followed by the function name and parameters. Functions can return values using the `return` statement and can accept any number of arguments. PHP also provides built-in functions for common tasks, such as string manipulation, mathematical operations, and database interactions.

5. File Handling and I/O Operations: PHP enables file handling operations, allowing developers to read from and write to files on the server's file system. This includes functions like `fopen()`, `fwrite()`, `fread()`, and `fclose()` for basic file manipulation. Additionally, PHP supports various input/output operations, such as reading form data, processing user input, and sending HTTP headers.

6. Database Integration is commonly used in conjunction with database management systems like MySQL, PostgreSQL, and SQLite for building dynamic web applications. It provides built-in functions and extensions for connecting to databases, executing SQL queries, and fetching results. This enables developers to create interactive websites that store and retrieve data from backend databases efficiently.

7. Security Considerations: When developing web applications with PHP, it's crucial to consider security best practices to protect against common vulnerabilities such as SQL injection, cross-site scripting (XSS), and cross-site request forgery (CSRF). Techniques like input validation, parameterized queries, and sanitization help mitigate these risks and ensure the integrity and confidentiality of user data.

Conclusion: In summary, PHP serves as a powerful tool for server-side web development, offering a versatile and feature-rich environment for building dynamic and interactive websites. Its seamless integration with HTML, extensive library of functions, and robust database support make it a popular choice among developers for creating a wide range of web applications. By mastering the basics of PHP, developers can unlock endless possibilities for crafting innovative and engaging online experiences.

Practical – 11: Write a PHP code to find the greater of 2 numbers Accept the no. from the user. (IT Lab)

Source code: -

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Find the Greater of Two Numbers</title>
</head>
<body>
    <h2>Find the Greater of Two Numbers</h2>
    <form method="post">
        Enter the first number: <input type="text" name="num1"><br>
        Enter the second number: <input type="text" name="num2"><br>
        <input type="submit" value="Find Greater">
    </form>


    <?php
    // Check if form is submitted
    if ($_SERVER["REQUEST_METHOD"] == "POST") {
      // Get the input values
      $num1 = $_POST['num1'];
      $num2 = $_POST['num2'];


      // Check which number is greater
      if ($num1 > $num2) {
        echo "<p>$num1 is greater than $num2</p>";
      } elseif ($num1 < $num2) {
        echo "<p>$num2 is greater than $num1</p>";
      } else {
        echo "<p>Both numbers are equal.</p>";
      }
    }
    ?>
</body>
</html>
```

Output: -

Find the Greater of Two Numbers

Enter the first number:
Enter the second number:
Find Greater

3 is greater than 2

Example– 1: Write a PHP program to display the following Binary Pyramid.

1

0 1

1 0 1

0 1 0 1

1 0 1 0 1. **(IT Lab)**

Source code: -

```php
<?php
$rows = 5; // Number of rows in the pyramid


// Loop through each row
for ($i = 1; $i <= $rows; $i++) {
    $num = $i % 2; // Initialize with the starting number for each row


    // Loop to print the numbers in each row
    for ($j = 1; $j <= $i; $j++) {
        echo $num . " "; // Print the current number

        // Toggle between 0 and 1 for the next number
        $num = ($num == 0) ? 1 : 0;
    }

    echo "\n"; // Move to the next line after printing each row
}
?>
```

Output: -

```
1
0 1
1 0 1
0 1 0 1
1 0 1 0 1
```

Example -3: Design a web page demonstrating different semantics. (Homework)

Source code: -

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Semantic Elements Example</title>
</head>
<body>
    <header>
        <h1>Welcome to our website</h1>
        <nav>
            <ul>
                <li><a href="#">Home</a></li>
                <li><a href="#">About</a></li>
                <li><a href="#">Services</a></li>
                <li><a href="#">Contact</a></li>
            </ul>
        </nav>
    </header>

    <main>
        <article>
            <header>
                <h2>Article Title</h2>
                <p>Published on <time datetime="2024-03-25">March 25, 2024</time></p>
            </header>
            <p>This is the content of the article...</p>
        </article>

        <section>
            <h2>Section Title</h2>
            <p>This is the content of the section...</p>
        </section>
    </main>

    <footer>
        <p>&copy; 2024 Example Website. All rights reserved.</p>
    </footer>
</body>
</html>
```

Output: -

Welcome to our website

- Home
- About
- Services
- Contact

Article Title

Published on March 25, 2024

This is the content of the article...

Section Title

This is the content of the section...

© 2024 Example Website. All rights reserved.

Example -4: Design a web page with a form that uses all types of controls. (Homework)

Source code: -

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Form with Different Controls</title>
</head>
<body>
    <h2>Form with Different Controls</h2>
    <form action="#" method="post">
        <!-- Text Input -->
        <label for="text_input">Text Input:</label>
        <input type="text" id="text_input" name="text_input"><br>

        <!-- Password Input -->
        <label for="password_input">Password Input:</label>
        <input type="password" id="password_input" name="password_input"><br>

        <!-- Textarea -->
        <label for="textarea">Textarea:</label><br>
        <textarea id="textarea" name="textarea" rows="4" cols="50"></textarea><br>

        <!-- Radio Buttons -->
        <label for="radio_male">Male:</label>
        <input type="radio" id="radio_male" name="gender" value="male">
        <label for="radio_female">Female:</label>
        <input type="radio" id="radio_female" name="gender" value="female"><br>

        <!-- Checkbox -->
        <label for="checkbox">Checkbox:</label>
        <input type="checkbox" id="checkbox" name="checkbox"><br>

        <!-- Select Dropdown -->
        <label for="select">Select Dropdown:</label>
        <select id="select" name="select">
            <option value="option1">Option 1</option>
            <option value="option2">Option 2</option>
            <option value="option3">Option 3</option>
        </select><br>

        <!-- File Input -->
        <label for="file_input">File Input:</label>
        <input type="file" id="file_input" name="file_input"><br>

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

Output: -

Form with Different Controls

Text Input:
Password Input:
Textarea:

Male: Female:
Checkbox:
Select Dropdown: Option 1
File Input: Choose File No file chosen
Submit

Learning Outcomes: -

Course Outcomes: -

Conclusion: -

Viva Question: -

1. What is PHP, and what does it stand for?

2. Explain the difference between echo and print statements in PHP.

3. How do you declare variables in PHP? Provide an example

4. Describe the usage of the foreach loop in PHP with a practical example.

For Faculty Use

Correction Parameters	Formative Assessment []	Timely completion of practical []	Attendance Learning Attitude []

Theory-7

String Functions and arrays

PHP provides a wide range of built-in functions for working with strings and arrays. Understanding these functions is crucial for efficient manipulation and processing of data in PHP applications. Below is an overview of string functions and arrays in PHP:

String Functions:

1. strlen($string): Returns the length of a string.

2. strtolower($string): Converts a string to lowercase.

3. strtoupper($string): Converts a string to uppercase.

4. substr($string, $start, $length): Extracts a substring from a string.

5. strpos($haystack, $needle): Finds the position of the first occurrence of a substring within a string.

6. str_replace($search, $replace, $string): Replaces all occurrences of a substring within a string with another substring.

7. trim($string): Removes whitespace or other predefined characters from the beginning and end of a string.

8. explode($delimiter, $string): Splits a string into an array of substrings based on a delimiter.

9. implode($glue, $array): Joins the elements of an array into a string using a specified glue.

10. htmlspecialchars($string): Converts special characters to HTML entities.

Arrays:

1. count($array): Returns the number of elements in an array.

2. array_push($array, $value): Adds one or more elements to the end of an array.

3. array_pop($array): Removes and returns the last element of an array.

4. array_shift($array): Removes and returns the first element of an array.

5. array_unshift($array, $value): Adds one or more elements to the beginning of an array.

6. sort($array): Sorts an array in ascending order.

7. rsort($array): Sorts an array in descending order.

8. array_merge($array1, $array2): Merges one or more arrays into a single array.

9. array_slice($array, $offset, $length): Extracts a slice of an array.

10. array_search($value, $array): Searches an array for a given value and returns the corresponding key if successful.

Conclusion:

Understanding string functions and arrays in PHP is essential for effective development. By leveraging these functions, developers can manipulate strings and arrays efficiently, enabling the creation of dynamic and feature-rich PHP applications. Experimenting with these functions and exploring their capabilities is key to mastering PHP programming.

Practical – 12: Write a PHP program to demonstrate string length, reverse functions**. (IT Lab)**

Source code: -

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>String Length and Reverse Functions</title>
</head>
<body>
    <h2>String Length and Reverse Functions</h2>
    <form method="post">
        Enter a string: <input type="text" name="input_string"><br>
        <input type="submit" value="Submit">
    </form>


    <?php
    // Check if form is submitted
    if ($_SERVER["REQUEST_METHOD"] == "POST") {
        // Get the input string
        $input_string = $_POST['input_string'];


        // Calculate string length
        $length = strlen($input_string);
        echo "<p>Length of the string: $length</p>";


        // Reverse the string
        $reversed_string = strrev($input_string);
        echo "<p>Reversed string: $reversed_string</p>";
    }
    ?>
</body>
</html>
```

Output: -

String Length and Reverse Functions

Enter a string: [ ]
Submit

Length of the string: 5

Reversed string: narak

Example –1: Write a PHP program to demonstrate count, concat string functions. (IT Lab)

Source code: -

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Count and Concatenate String Functions</title>
</head>
<body>
    <h2>Count and Concatenate String Functions</h2>
    <form method="post">
        Enter a string: <input type="text" name="input_string"><br>
        <input type="submit" value="Submit">
    </form>


    <?php
    // Check if form is submitted
    if ($_SERVER["REQUEST_METHOD"] == "POST") {
        // Get the input string
        $input_string = $_POST['input_string'];


        // Count the number of characters in the string
        $length = strlen($input_string);
        echo "<p>Number of characters in the string: $length</p>";


        // Concatenate the string with itself
        $concatenated_string = $input_string . $input_string;
        echo "<p>Concatenated string: $concatenated_string</p>";
    }
    ?>
</body>
</html>
```

Output: -

Count and Concatenate String Functions

Enter a string: [ ]
Submit

Number of characters in the string: 5

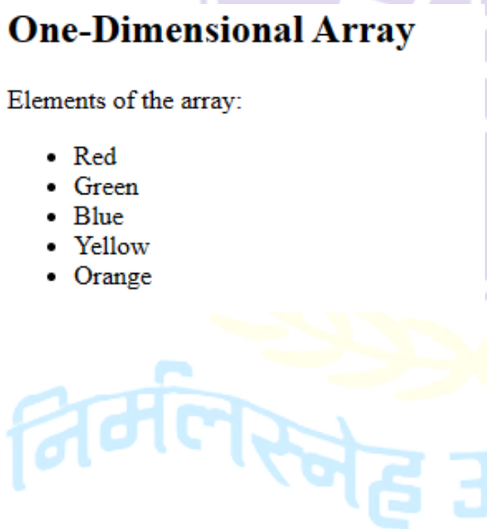
Concatenated string: karankaran

Example–2: Write a PHP program to create one dimensional array. (Homework)

Source code: -

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>One-Dimensional Array</title>
</head>
<body>
    <h2>One-Dimensional Array</h2>


    <?php
    // Define a one-dimensional array
    $colors = array("Red", "Green", "Blue", "Yellow", "Orange");


    // Display the elements of the array
    echo "<p>Elements of the array:</p>";
    echo "<ul>";
    foreach ($colors as $color) {
        echo "<li>$color</li>";
    }
    echo "</ul>";
    ?>


</body>
</html>
```

Output: -



Example -3: Write a JavaScript program to demonstrate removing and inserting elements at the

Source code: -

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Removing and Inserting Elements in JavaScript</title>
</head>
<body>
    <h2>Removing and Inserting Elements in JavaScript</h2>

    <p>Original Array:</p>
    <p id="originalArray"></p>


    <button onclick="removeAndInsert()">Remove and Insert</button>


    <script>
        // Initial array
        let numbers = [1, 2, 3, 4, 5];
        document.getElementById("originalArray").textContent = numbers.join(", ");


        function removeAndInsert() {
            // Remove element at index 2
            numbers.splice(2, 1);


            // Insert element 10 at index 2
            numbers.splice(2, 0, 10);


            // Display the updated array
            alert("Updated Array: " + numbers.join(", "));
        }
    </script>
</body>
</html>
```

Output: -

Removing and Inserting Elements in JavaScript

Original Array:

1, 2, 3, 4, 5

Remove and Insert

Learning Outcomes: -

Course Outcomes: -

Conclusion: -

Viva Question: -

1. What is the purpose of the `strlen()` function in PHP?
2. How do you concatenate two strings in PHP?
3. Explain the difference between `array_push()` and `array_pop()` functions in PHP.
4. How can you determine the length of an array in PHP without using the `count()` function?

For Faculty Use

Correction Parameters	Formative Assessment []	Timely completion of practical []	Attendance Learning Attitude []

Theory-8

PHP and Database

PHP is a server-side scripting language commonly used for web development. One of its key strengths lies in its ability to interact with databases seamlessly. This integration enables dynamic web applications to retrieve, manipulate, and store data efficiently. Below is an overview of PHP's interaction with databases:

1. Database Connectivity:

- PHP provides various extensions and libraries to connect to different types of databases, such as MySQL, PostgreSQL, SQLite, and more.

- The most commonly used extension for MySQL databases is MySQLi (MySQL Improved) and PDO (PHP Data Objects), which offer secure and efficient ways to interact with databases.

2. Connecting to a Database:

- PHP scripts establish a connection to the database server using functions like `mysqli_connect()` or `PDO::__construct()`.

- These functions require credentials such as hostname, username, password, and database name to establish a connection.

3. Executing SQL Queries:

- After establishing a connection, PHP can execute SQL queries to perform various operations like SELECT, INSERT, UPDATE, and DELETE.

- Queries are executed using functions like `mysqli_query()` or `PDO::query()`.

- Prepared statements can be used to prevent SQL injection attacks and improve performance.

4. Retrieving Data:

- SELECT queries retrieve data from the database. PHP fetches the results using functions like `mysqli_fetch_assoc()`, `mysqli_fetch_array()`, or `PDOStatement::fetch()`.

- Data is typically fetched row by row and processed as needed.

5. Manipulating Data:

- INSERT, UPDATE, and DELETE queries are used to manipulate data in the database.

- PHP executes these queries using functions like `mysqli_query()` or `PDO::exec()`.

6. Error Handling:

- PHP provides mechanisms to handle errors that may occur during database operations.

- Functions like `mysqli_error()` or `PDO::errorInfo()` can be used to retrieve error messages.

7. Closing Database Connection:

- It is essential to close the database connection once the operations are completed to free up resources.

- PHP scripts use functions like `mysqli_close()` or `PDO::close()` to close the connection.

8. Best Practices:

- Use prepared statements or parameterized queries to prevent SQL injection attacks.

- Validate and sanitize user input before executing queries to enhance security.

- Close database connections after use to conserve server resources.

- Implement error handling to gracefully handle database errors and exceptions.

Practical – 13: Write a PHP code to create: Create a database College Create a table Department (D name, Dno, Number Of faculty) **(IT Lab)**

Source code: -

```php
<?php
// Database connection parameters
$servername = "localhost";
$username = "root"; // Replace with your MySQL username
$password = ""; // Replace with your MySQL password


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


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


// Create database
$sql_create_db = "CREATE DATABASE IF NOT EXISTS College";
if ($conn->query($sql_create_db) === TRUE) {
  echo "Database 'College' created successfully<br>";
} else {
  echo "Error creating database: " . $conn->error;
}


// Select the created database
$conn->select_db("College");


// Create table
$sql_create_table = "CREATE TABLE IF NOT EXISTS Department (
    D_name VARCHAR(50) NOT NULL,
    Dno INT(11) AUTO_INCREMENT PRIMARY KEY,
    Number_Of_faculty INT(11)
)";
if ($conn->query($sql_create_table) === TRUE) {
  echo "Table 'Department' created successfully";
} else {
  echo "Error creating table: " . $conn->error;
}


// Close connection
$conn->close();

?>
```

Output: -

```
Database 'College' created successfully
Table 'Department' created successfully
```

Example – 1: Write a PHP program to create a database named "College". Create a table named "Student" with following fields (sno, name, percentage). Insert 3 records of your choice. Display the names of the students whose percentage is between 35 to 75 in a tabular format. (IT Lab)

Source code: -

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Display Students with Percentage Between 35 to 75</title>
    <style>
        table {
            border-collapse: collapse;
            width: 50%;
        }
        th, td {
            border: 1px solid black;
            padding: 8px;
            text-align: left;
        }
    </style>
</head>
<body>
    <h2>Students with Percentage Between 35 to 75</h2>
    <table>
        <thead>
            <tr>
                <th>Serial No</th>
                <th>Name</th>
                <th>Percentage</th>
            </tr>
        </thead>
        <tbody>
            <?php
            // Database connection parameters
            $servername = "localhost";
            $username = "root"; // Replace with your MySQL username
            $password = ""; // Replace with your MySQL password
            $dbname = "College";


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


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

            // SQL query to select students with percentage between 35 to 75
            $sql_select = "SELECT sno, sname, percentage FROM Student WHERE percentage BETWEEN 35 AND 75";
            $result = $conn->query($sql_select);


            if ($result->num_rows > 0) {
                // Output data of each row
                while($row = $result->fetch_assoc()) {
                    echo "<tr><td>" . $row["sno"] . "</td><td>" . $row["sname"] . "</td><td>" . $row
                    ["percentage"] . "</td></tr>";
                }
            } else {
                echo "<tr><td colspan='3'>No records found</td></tr>";
            }
            $conn->close();
            ?>
        </tbody>
    </table>
</body>
```

Output: -

Students with Percentage Between 35 to 75

Records inserted successfully

Serial No	Name	Percentage
2	Alice	50
3	Bob	60

Example – 2: Design a PHP page for authenticating a user. (Homework)

Source code: -

```
<!DOCTYPE html>
<html lang="en">

<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>User Authentication</title>
</head>

<body>
  <h2>User Authentication</h2>
  <form method="post" action="authenticate.php">
    <label for="username">Username:</label>
    <input type="text" id="username" name="username" required><br><br>

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

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

</html>
```

Output: -

User Authentication

Username:

Password:

Login

Example - 3: Write a PHP program to create a database named "College". Create a table named "Student" with following fields (sno, sname, percentage). Insert 3 records of your choice. Display the names of the students whose percentage is between 35 to 75 in a tabular format. (Homework)

Source code: -

```php
<?php
// Database connection parameters
$servername = "localhost";
$username = "root"; // Replace with your MySQL username
$password = ""; // Replace with your MySQL password
$dbname = "College"; // Corrected database name

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

// Check connection
if ($conn->connect_error) {
    die("Connection failed: " . $conn->connect_error);
}
// Define SQL query to insert records into the Student table
$sql_insert = "INSERT INTO Student (sno, sname, percentage)
               VALUES
               (1, 'John', 80),
               (2, 'Alice', 50),
               (3, 'Bob', 60)";

// Attempt to execute the insert query
try {
    $conn->query($sql_insert);
    echo "Records inserted successfully<br>";
} catch (mysqli_sql_exception $e) {
    if ($e->getCode() == 1062) {
        echo "Error: Duplicate entry. Please ensure unique primary key values.";
    } else {
        echo "Error inserting records: " . $e->getMessage();
    }
}

// SQL query to select students with percentage between 35 to 75
$sql_select = "SELECT sno, sname, percentage FROM Student WHERE percentage BETWEEN 35 AND 75";
$result = $conn->query($sql_select);

// Display student records in a table
if ($result->num_rows > 0) {
    // Output data of each row
    while ($row = $result->fetch_assoc()) {
        echo "<tr><td>" . $row["sno"] . "</td><td>" . $row["sname"] . "</td><td>" . $row["percentage"] . "</td></tr>";
    }
} else {
    echo "<tr><td colspan='3'>No records found</td></tr>";
}

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

Output: -

Error: Duplicate entry. Please ensure unique primary key values.2Alice503Bob60

Learning Outcomes: -

Course Outcomes: -

Conclusion: -

Viva Question: -

1. What is the role of PHP in database connectivity?
2. How can you establish a connection between PHP and a MySQL database?
3. Explain the purpose of the `mysqli_query()` function in PHP.
4. How do you handle errors in PHP database operations?

For Faculty Use

Correction Parameters	Formative Assessment []	Timely completion of practical []	Attendance Learning Attitude []

Theory-9

Email

Sending email is a common requirement in web applications, and PHP provides built-in functions to facilitate this task. Below is an overview of sending emails in PHP:

1. Configuring SMTP:

- Before sending emails via PHP, ensure that your server is configured to send emails. Typically, this involves configuring Simple Mail Transfer Protocol (SMTP) settings in your PHP configuration file (`php.ini`). You need to specify the SMTP server address, port, authentication credentials, etc.

2. Using the mail() Function:

- PHP provides the `mail()` function to send emails. It accepts multiple parameters, including the recipient's email address, subject, message body, additional headers, etc.
- Example: `mail($to, $subject, $message, $headers)`

3. Building Email Headers:

- Headers contain additional information about the email, such as sender, recipient, subject, and content type. You can specify headers like `From`, `To`, `Cc`, `Bcc`, `Reply-To`, etc., using the `$headers` parameter in the `mail()` function.
- Example: `$headers = "From: sender@example.com\r\n";`

4. Content Type and Encoding:

- Specify the content type and encoding in email headers to ensure proper rendering of email content. Common content types include text/plain, text/html, multipart/mixed (for attachments), etc. Encoding methods like base64 or quoted-printable may be used for attachments or non-ASCII characters.
- Example: `$headers .= "Content-Type: text/html; charset=UTF-8\r\n";`

5. Handling Attachments:

- ❖ You can attach files to emails using MIME encoding. Read the file contents, encode it (usually in base64), and include it as part of the email body with appropriate MIME headers.
- ❖ Example: `base64_encode(file_get_contents('attachment.pdf'))`

6. SMTP Libraries:

- ❖ While PHP's `mail()` function is suitable for simple email sending, for more advanced features like SMTP authentication, HTML email formatting, and handling attachments, you may consider using third-party libraries like PHPMailer or Swift Mailer.

7. Error Handling:

- Check the return value of the `mail()` function to determine if the email was sent successfully. If an error occurs, the function may return `false`, indicating failure. Proper error handling ensures graceful handling of email sending failures.

8. Testing and Debugging:

- During development, it's essential to test email functionality thoroughly. Use debugging techniques like logging or echoing to troubleshoot issues with email sending. Additionally, tools like Mailtrap or SMTP debugging tools can help simulate email sending without actually delivering emails to recipients.

Practical – 14: Write a program to send e-mail with attachment. (IT Lab)

Source code: -

```php
<?php
// Recipient Email Address
$to_email = "recipient@example.com";

// Sender Email Address
$from_email = "sender@example.com";

// Email Subject
$subject = "Email with Attachment";

// Email Body
$message = "This email contains an attachment.";

// Attachment file path
$file_path = "karangautamIBM.pdf";
 // Replace with the actual file path

// Get file name from file path
$file_name = basename($file_path);

// Generate a random boundary
$random_hash = md5(date('r', time()));

// Email headers
$headers = "From: $from_email\r\n";
$headers .= "Reply-To: $from_email\r\n";
$headers .= "MIME-Version: 1.0\r\n";
$headers .= "Content-Type: multipart/mixed; boundary=\"PHP-mixed-$random_hash\"\r\n";

// Read the attachment file content
$file_content = file_get_contents($file_path);

if ($file_content === false) {
    die("Failed to open attachment file.");
}

// Encode the attachment content
$attachment = chunk_split(base64_encode($file_content));

// Email body with attachment
$body = "--PHP-mixed-$random_hash\r\n";
$body .= "Content-Type: multipart/alternative; boundary=\"PHP-alt-$random_hash\"\r\n\r\n";
$body .= "--PHP-alt-$random_hash\r\n";
$body .= "Content-Type: text/plain; charset=\"iso-8859-1\"\r\n";
$body .= "Content-Transfer-Encoding: 7bit\r\n\r\n";
$body .= $message."\r\n\r\n";
$body .= "--PHP-alt-$random_hash\r\n";
$body .= "Content-Type: application/pdf; name=\"$file_name\"\r\n";
$body .= "Content-Disposition: attachment; filename=\"$file_name\"\r\n";
$body .= "Content-Transfer-Encoding: base64\r\n";
$body .= $attachment."\r\n\r\n";
$body .= "--PHP-mixed-$random_hash--";

// Send email
if (mail($to_email, $subject, $body, $headers)) {
    echo "Email sent successfully.";
} else {
    echo "Email sending failed.";
}
?>
```

Output: -

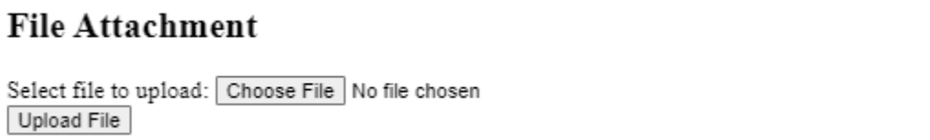
Warning: mail(): Failed to connect to mailserver at "localhost" port 25, verify your "SMTP" and "smtp_port" setting in php.ini or use ini_set() in **C:\Users\PC`\Desktop\php\index.php** on line **55**
Email sending failed.

Example – 1: Write a program for attachment of the files. (IT Lab)

Source code: -

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>File Attachment</title>
</head>
<body>
    <h2>File Attachment</h2>
    <form action="upload.php" method="post" enctype="multipart/form-data">
        <label for="file">Select file to upload:</label>
        <input type="file" name="file" id="file"><br>
        <input type="submit" value="Upload File" name="submit">
    </form>
</body>
</html>


<?php
// Check if the form is submitted
if(isset($_POST["submit"])) {
    // File upload directory
    $target_dir = "uploads/";


    // Get the filename
    $filename = basename($_FILES["file"]["name"]);


    // Concatenate target directory with the filename
    $target_file = $target_dir . $filename;


    // Check if file already exists
    if (file_exists($target_file)) {
        echo "Sorry, the file already exists.";
    } else {
        // Upload the file
        if (move_uploaded_file($_FILES["file"]["tmp_name"], $target_file)) {
            echo "The file " . $filename . " has been uploaded successfully.";
        } else {
            echo "Sorry, there was an error uploading your file.";
        }
    }
}
?>
```

Output: -



Example – 2: Write a program to send e-mail. (Homework)

Source code: -

```php
<?php
// Sender's email address
$from_email = "karangautam2023@gmail.com";


// Recipient's email address
$to_email = "ankitsharma11052005@gmail.com";


// Subject of the email
$subject = "Test Email";


// Message body
$message = "This is a test email sent from PHP.";


// Additional headers
$headers = "From: $from_email\r\n";
$headers .= "Reply-To: $from_email\r\n";
$headers .= "MIME-Version: 1.0\r\n";
$headers .= "Content-Type: text/plain; charset=UTF-8\r\n";
$headers .= "X-Mailer: PHP/" . phpversion();


// Send the email
if (mail($to_email, $subject, $message, $headers)) {
    echo "Email sent successfully.";
} else {
    echo "Failed to send email.";
}
?>
```

Output: -

```
Failed to send email.
```

Learning Outcomes: -

Course Outcomes: -

Conclusion: -

Viva Question: -

1. What PHP function is commonly used to send emails?
2. How can you specify additional headers, such as sender and subject, when sending an email in PHP?
3. What is the purpose of the `mail()` function's return value in PHP email sending?

4. Explain the significance of the `enctype="multipart/form-data"` attribute in HTML forms when uploading files for email attachments via PHP.

For Faculty Use

Correction Parameters	Formative Assessment []	Timely completion of practical []	Attendance Learning Attitude []

Theory-10

Sessions and Cookies

Introduction:

Sessions and cookies are important mechanisms in web development for maintaining user state and storing information across multiple requests. In PHP, sessions and cookies are commonly used to personalize user experiences, track user activity, and implement various security features.

Sessions:

- Sessions are a server-side mechanism used to store user data during a user's interaction with a website.
- PHP sessions work by generating a unique session ID for each user. This ID is stored as a cookie on the user's browser or passed through URLs.
- Session data is stored on the server and associated with the user's session ID. It can include user-specific information like login status, preferences, shopping cart contents, etc.
- To start a session in PHP, you typically use the `session_start()` function. This function initializes or resumes a session and allows you to access session data stored in the `$_SESSION` superglobal array.
- Session data remains available to the user as long as the session is active or until the user closes their browser. Sessions automatically expire after a specified period of inactivity or when the browser is closed, depending on server configuration.

Cookies:

- Cookies are small pieces of data stored on the user's browser by websites. They are commonly used for tracking user activity, storing user preferences, and maintaining user sessions.
- Cookies can be set, retrieved, and manipulated using PHP's `setcookie()` and `$_COOKIE` functions.
- Cookies have attributes such as name, value, expiration time, domain, and path. These attributes determine when and where the cookie can be accessed.
- Persistent cookies have an expiration time set in the future and remain stored on the user's browser even after they close it. Session cookies, on the other hand, expire when the browser session ends.
- Cookies can be used to implement features like remembering user login credentials, tracking user behavior for analytics, and personalizing website content.

Security Considerations:

- While sessions and cookies are essential for web development, they also pose security risks if not implemented properly.
- Session hijacking and cookie theft are common attacks that exploit vulnerabilities in session and cookie management.
- To mitigate these risks, developers should implement security measures such as using HTTPS to encrypt data transmission, setting secure and HTTP-only flags for cookies, and validating user input to prevent injection attacks.
- Additionally, sensitive information should not be stored directly in cookies or sessions without proper encryption or hashing.

Practical – 15: Write a program to demonstrate use of sessions and cookies. **(IT Lab)**

Source code: -

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


// Check if the user is logged in
if (isset($_SESSION['username'])) {
    // If logged in, welcome the user
    $message = "Welcome back, " . $_SESSION['username'] . "!";
} else {
    // If not logged in, redirect to the login page
    header("Location: login.php");
    exit;
}


// Set a cookie to track the number of visits
$visits = isset($_COOKIE['visits']) ? $_COOKIE['visits'] + 1 : 1;
setcookie('visits', $visits, time() + (86400 * 30), "/"); // Cookie valid for 30 days


// Display the number of visits
$visit_message = "You have visited this page $visits time(s).";
?>


<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Session and Cookie Demo</title>
</head>
<body>
    <h2>Session and Cookie Demo</h2>
    <p><?php echo $message; ?></p>
    <p><?php echo $visit_message; ?></p>
</body>
</html>
```

Output: -

Session and Cookie Demo

Example – 1: Write a program in php for create/retrieve a cookie. **(IT Lab)**

Source code: -

```php
<?php
// Create a cookie
$cookie_name = "user";
$cookie_value = "John Doe";
$cookie_expiration = time() + (86400 * 30); // Cookie valid for 30 days
$cookie_path = "/"; // Cookie available to entire domain


setcookie($cookie_name, $cookie_value, $cookie_expiration, $cookie_path);


// Retrieve the cookie
if(isset($_COOKIE[$cookie_name])) {
    $message = "Cookie '" . $cookie_name . "' is set with value '" . $_COOKIE[$cookie_name] . "'";
} else {
    $message = "Cookie '" . $cookie_name . "' is not set";
}
?>


<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Cookie Demo</title>
</head>
<body>
    <h2>Cookie Demo</h2>
    <p><?php echo $message; ?></p>
</body>
</html>
```

Output: -

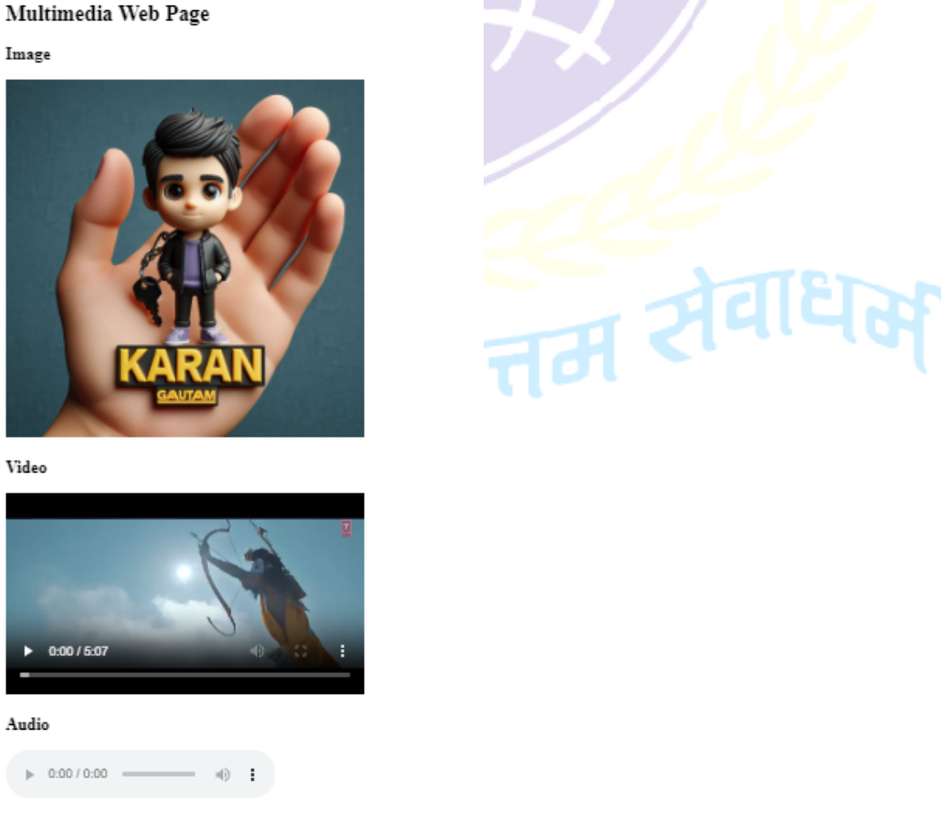
Cookie Demo

Example - 2: Design a web page embedding with multimedia features**. (Homework)**

Source code: -

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Multimedia Web Page</title>
</head>
<body>
    <h2>Multimedia Web Page</h2>

    <!-- Image -->
    <h3>Image</h3>
    <img src="./img.jpeg" alt="Image" width="400">


    <!-- Video -->
    <h3>Video</h3>
    <video width="400" controls>
        <source src="./video.mp4" type="video/mp4">
        Your browser does not support the video tag.
    </video>


    <!-- Audio -->
    <h3>Audio</h3>
    <audio controls>
        <source src="audio.mp3" type="audio/mpeg">
        Your browser does not support the audio element.
    </audio>

</body>
</html>
```

Output: -



Example - 3: Design a web page to demonstrate grouping selectors (CSS). **(Homework**

Source code: -

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Grouping Selectors Demo</title>
    <style>
        /* Grouping selectors */
        h1, h2, p {
            color: #333; /* Common color */
            font-family: Arial, sans-serif; /* Common font-family */
        }


        .box1, .box2, .box3 {
            width: 100px; /* Common width */
            height: 100px; /* Common height */
            background-color: lightblue; /* Common background color */
            margin-bottom: 20px; /* Common margin */
        }
    </style>
</head>
<body>
    <h1>Grouping Selectors Demo</h1>

    <h2>Heading 2</h2>

    <div class="box1"></div>
    <div class="box2"></div>
    <div class="box3"></div>

    <p>Lorem ipsum dolor sit amet, consectetur adipiscing elit. Sed ac justo magna.</p>
</body>
</html>
```

Output: -

Learning Outcomes: -

Course Outcomes: -

Conclusion: -

Viva Question: -

1. What is PHP, and what does it stand for?

2. Explain the difference between echo and print statements in PHP.

3. How do you declare variables in PHP? Provide an example

4. Describe the usage of the foreach loop in PHP with a practical example.

For Faculty Use

Correction Parameters	Formative Assessment []	Timely completion of practical []	Attendance Learning Attitude []