Lab Activities : 1

Write program to print number from 1 to 10 using for loop, while and do while.

A) Using for loop:

```
console.log("Printing numbers from 1 to 10 using for loop:");
for (let i = 1; i <= 10; i++) {
   console.log(i);
}
```

B) Using while loop:

```
console.log("Printing numbers from 1 to 10 using while loop:");
let j = 1;
while (j <= 10) {
   console.log(j);
   j++;
}
```

C) Using do while loop:

```
console.log("Printing numbers from 1 to 10 using do while
loop:");
let k = 1;
do {
   console.log(k);
   k++;
} while (k <= 10);
```

Lab Activities : 2

Write a program to checks whether the number is positive, negative, or zero, and print the result accordingly.

```
let number = prompt("Enter a number:");

number = parseFloat(number);

if (number > 0) {
   console.log(number + " is positive.");
} else if (number < 0) {
   console.log(number + " is negative.");
} else {
   console.log(number + " is zero.");
```

Lab Activities : 3

Write a JS program that demonstrates variable scopes by declaring variables with different scopes and observing their behavior. Include variables declared within functions as well as those declared outside of any function.

```
var globalVar = "I'm a global var (using var)";
let globalLet = "Global let";
const globalConst = "Global const";

function scoping() {

var functionVar = "Function var";
let functionLet = "Function let";
const functionConst = "Function const";
```

```
    console.log("Inside the function:");
   console.log("Global Var:", globalVar);
    console.log("Global Let:", globalLet);
    console.log("Global Const:", globalConst);
    console.log("Function Var:", functionVar);
    console.log("Function Let:", functionLet);
    console.log("Function Const:", functionConst);
}

// Accessing global scope variables
console.log("Outside the function:");
console.log("Global Var:", globalVar);
console.log("Global Let:", globalLet);
console.log("Global Const:", globalConst);

// Accessing function scope variables outside the function: Result-
ReferenceError

// console.log("functionVar:", functionVar); // Error
// console.log("functionLet:", functionLet); // Error
// console.log("functionConst:", functionConst); // Error

// Reassigning global const variable: Result-TypeError
// globalConst = "Reassign global const";

scoping();
```

Lab Activities : 4

Declare an array named friends containing the names of your favourite friends and print the contents of the array to the console.

```
let friends = ["kanav", "Yash", "Ram", "Sham", "Aryan"];


console.log("My favorite friends:");
for (let i = 0; i < friends.length; i++) {
   console.log(friends[i]);
```

Lab Activities : 5

Declare an array named fruits containing the names of different fruits. Use the push, pop, slice, print the updated array to the console after each operation.

```
let fruits = ["Apple", "Banana", "Orange", "Mango", "Grapes"];
console.log("Initial array:", fruits);


// Push:
fruits.push("Pineapple");
console.log("After adding Pineapple in Array:", fruits);
// Pop:
let removeFruit = fruits.pop();
console.log("After removing last fruit from an Array:", removedFruit + ,
fruits);
// Slice:
let slicedFruits = fruits.slice(1, 4);
console.log("Sliced array (fruits[1:4]): ", slicedFruits
```

Lab Activities : 6

Declare an array named friends containing the names of your favourite friends and print the contents of the array to the console.

```
let sum = function(num1, num2) {
    return num1 + num2;
};

// Random Number between 50
let randomNum1 = Math.floor(Math.random() * 50);
let randomNum2 = Math.floor(Math.random() * 50);

console.log("Random num1:", randomNum1);
console.log("Random num2:", randomNum2);
console.log("Sum of the random numbers:", sum(randomNum1, randomNum2));
```

Lab Activities : 7

Create a JavaScript object representing a student with properties like name, age, email, and courses enrolled. Display the student object in the console.

```
let student = {
    name: "Ram",
    age: 24,
    email: "ram.dev@example.com",
    coursesEnrolled: ["FED", "Operating System", "DS"]
};
```

```
console.log("Student Information:");
console.log("Name:", student.name);
console.log("Age:", student.age);
console.log("Email:", student.email);
console.log("Courses Enrolled:", student.coursesEnrolled);
```

Lab Activities : 8

Parse the JSON string back to an object using JSON.parse() method. Display the parsed object in the console.

```
 <!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<title>JSON</title>
</head>
<body>
  <h1>JSON String- object</h1>
    <script>
          let jsonString = '{"name":"Ram","age":24,"email":"ram.dev@example.com","coursesEnrolled":["FED","OS","DS"]}';

          let parsedObject = JSON.parse(jsonString);
          console.log("Parsed Object:");
          console.log(parsedObject);
    </script>
</body>
</html>
```

Lab Activities : 9

Create an HTML document with multiple elements such as <div>, <p>, and <span>. Write JavaScript code to select and modify specific elements using document.getElementById(), document.querySelector(), and other DOM manipulation methods.

```
 <!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>
   <div>
      <div class="div_first">
         This is a first div
      </div>
      <h1 id="heading">This is a heading</h1>
      <p id="paragraph">This is a paragraph</p>

      <div class="div_second">
         This is a Second Div
         <span class="spann">inside which span is present</span>
      </div>
   </div>
   <script>
      let heading = document.getElementById('heading');
      heading.style.color = "green";

      let para = document.getElementById('paragraph');
```

```
        para.style.backgroundColor = "pink";

        let divs = document.getElementsByClassName('div_first');
        divs[0].style.backgroundColor = "blue";


        let spanElement = document.querySelector('.spann');
        spanElement.style.fontWeight = 'bold';
    </script>
</body>
</html
```

Lab Activities : 10

Write JavaScript functions to create new HTML elements such as <li>, <button>, or <input> dynamically. Demonstrate how to append these elements to existing parent elements in the DOM tree. Implement functionality to update the content or attributes of dynamically created elements based on user input or events.

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-
scale=1.0">
<title>Dynamically Create HTML Elements</title>
</head>
<body>
    <h1>Programming Language: </h1>
    <ul id="list">
        <li>Java</li>
        <li>C++</li>
    </ul>
    <button id="addButton">Add New List Item</button>

    <script>
        function createListItem() {

            let newListItem = document.createElement("li");

            let listItemContent = prompt("Enter Programming language to
enter inside a list:");

            newListItem.textContent = listItemContent;

            let listContainer = document.getElementById("list");
            listContainer.appendChild(newListItem);
        }

        let addButton = document.getElementById("addButton");
        addButton.addEventListener("click", createListItem);
    </script>
</body>
</html>
```

Lab Activities : 11

Write a JavaScript function that simulates asynchronous behavior using setTimeout(). The function should take a callback as an argument and execute it after a specified delay.

```
function maggiLaao(cb) {
  console.log("Maggi lene chale gaye");
  setTimeout(function () {
    console.log("Maggi Lekar aa gaye");
    cb(maggiKhaao);
  }, 2000);
}

function maggiBanao(cb) {
  console.log("Maggi Banana start");
  setTimeout(function () {
    console.log("Maggi Bann gai");
    cb();
  }, 2000);
}

function maggiKhaao() {
  console.log("Maggi Khana start");
  setTimeout(function () {
    console.log("Maggi Khana Samapt");
  }, 2000);
}
maggiLaao(maggiBanao);
```

Lab Activities : 12

Create a simple web page with a button. Upon clicking the button, use Ajax to fetch data from a public API (e.g., JSON-Placeholder) and display it on the page.

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-
scale=1.0">
<title>Fetch Data from JSONPlaceholder API</title>
</head>
<body>
  <h1>Fetch Data from JSONPlaceholder API</h1>
  <button id="fetchButton">Fetch Data</button>
  <div id="data"></div>

  <script>
    function fetchData() {
      let xhr = new XMLHttpRequest();
      let url = "https://jsonplaceholder.typicode.com/posts/1";
      xhr.open("GET", url, true);

      xhr.onload = function() {
        if (xhr.status === 200) {

          let responseData = JSON.parse(xhr.responseText);

          let dataContainer = document.getElementById("data");
          dataContainer.innerHTML = "<h2>Data:</h2>" +
            "<p>Title: " + responseData.title + "</p>" +
            "<p>Body: " + responseData.body + "</p>";
            } else {
                console.error("Error:", xhr.status);
            }
        };
        xhr.send();

        let fetchButton = document.getElementById("fetchButton");
        fetchButton.addEventListener("click", fetchData);
    </script>
</body>
</html>
```

Lab Activities : 13

Write a JavaScript function that attempts to parse a JSON string. Use a try-catch block to handle any errors that may occur during parsing. If successful, log the parsed data; otherwise, log an error message.

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-
scale=1.0">
<title>JSON</title>
</head>
<body>
  <h1>JSON-TRY AND CATCH</h1>
   <script>
          function parseJSON(jsonString) {
              try {
                  let Data = JSON.parse(jsonString);

                  console.log("Parsed data:", Data);
              } catch (error) {
                  console.error("Error: ", error.message);
              }}
          let jsonStringOne = '{"name": "Ram", "age": 24}';
          let jsonStringTwo = '{"name": "Sham", "age": }';

          parseJSON(jsonStringOne);
          parseJSON(jsonStringTwo);
    </script>
</body>
</html
```

Lab Activities : 14

Demonstrate the concept of closure by calling the inner function and accessing the outer variables.

```
function outer(){
    var a=10;
    function inner(){
        a++;
        function innermost(){
            a++;
            console.log(a);
        }
        return innermost;
    }
    return inner;
}
let f1=outer();
let f2=outer();

let f11=f1();
let f12=f1();

let f21=f2();
let f22=f2();
f11();
f11();
f12();
f12();
f21();
f21();
f22();
```

Lab Activities : 15

Understanding component by Create two separate JavaScript files, one containing a React component and another serving as the main entry file.

- **App.js:**

```
import React from 'react'

const Person = () => {
  return (
    <article className='person'>
        <h3>Name: Yash</h3>
        <h4>Age: 23</h4>
        <p>Crush: Piggy</p>
    </article>
  )
}

export default Person
```

- **Person.js:**

```
import React from 'react'

const Person = () => {
  return (
    <article className='person'>
        <h3>Name: Yash</h3>
        <h4>Age: 23</h4>
        <p>Crush: Piggy</p>
    </article>
  )
}

export default Person
```

Lab Activities : 16

Design and utilize template literals to dynamically generate content within a React component.

- **App.js:**

```
import LearnJSX from "./components/LearnJSX"

function App(){
  return <div>
    <LearnJSX/>
    </div>
}
export default App
```

- **LearnJSX.jsx:**

```
import React from 'react'

const LearnJSX = () => {

  const randomNum = Math.floor(Math.random()*10+1)

  const luckyNum = 7;

  return (
    <div>
        <h1>This is a heading tag:
          {Math.floor(Math.random()*6+1)}
        </h1>
        <h2>My Random Number Btw 1 to 10 is:{randomNum}
</h2>
        {randomNum === luckyNum ?
      <img src="https://media.giphy.com/me-
dia/v1.Y2lkPTc5MGI3NjExdW5naGYxcmphZW00ZHE1emJuajE
2OWt3ZnI5ZzNqMTVmcmU4Njg2biZlcD12MV9pbnRlcm5hbF9n
aWZfYnlfaWQmY3Q9Zw/2sXf9PbHcEdE1x059I/giphy.gif"/> :
<p>Not a lucky Number 😰</p>
      }
    </div>
  )
}
export default LearnJSX
```

Lab Activities : 17

Develop a simple single-page application using React.js and also Understand the differences between CSR and SSR in React.js.

```
import import React, { useState } from 'react';

// Counter component
const Counter = () => {
  const [count, setCount] = useState(0);

  const increment = () => {
    setCount(count + 1);
  };

  const decrement = () => {
    setCount(count - 1);
  };

  return (
    <div>
      <h2>Counter</h2>
      <button onClick={decrement}>-</button>
      <span>{count}</span>
      <button onClick={increment}>+</button>
    </div>
  );
};

// App component
const App = () => {
  return (
    <div>
      <h1>This is a Simple SPA with React Example</h1>
      <Counter />
    </div>
  );
};

export default App;
```

Client-Side Rendering (CSR):

A) In CSR, the entire rendering process happens in the browser.
B) When a user visits a CSR-based application, the browser fetches the HTML file, along with a bundle of JavaScript files.
C) The browser then executes the JavaScript files to render the UI dynamically.

Server-Side Rendering (SSR):

A) In SSR, the initial rendering of the UI occurs on the server.
B) When a user visits an SSR-based application, the server generates the HTML for the initial page load dynamically and sends it to the browser.
C) Once the HTML is received, the browser renders the page immediately, without waiting for JavaScript files to be down-loaded and executed.

Lab Activities : 18

Creating a simple React Page app. That, practice handle form events such as onChange and onSubmit in React.

```
import import React, { useState } from 'react'
import '../form/Form.css'

const Form = () => {

   const [name,setName] = useState('')
   const [price,setPrice] = useState(0)
   const [img,setImg] = useState('')

   const nameChangeHandler = (event)=> {
      setName(event.target.value)
   }

   const priceChangeHandler = (event)=> {
      setPrice(event.target.value)
   }
```

```
    const imgUrlChangeHandler = (event)=> {
        setImg(event.target.value)
    }

    const formSubmitHandler = (event) =>{
        event.preventDefault();
        console.log(name)
        console.log(price)
        console.log(img)
    }

  return (
    <form onSubmit={formSubmitHandler} className='form'>
        <div>
            <label htmlFor='name'>Name:
            </label>
            <input type='text' id='name' placeholder='Product Name'
onChange={nameChangeHandler}/>
            <div>
            <label htmlFor='name'>Price:
            </label>
            <input type='number' id='name' placeholder='Product
Price' onChange={priceChangeHandler}/>
        </div>
        <div>
            <label htmlFor='name'>Image Url:
            </label>
            <input type='text' id='img' placeholder='Product Img Url'
onChange={imgUrlChangeHandler}/>
        </div>
        <button>Create Product</button>
        </div>
    </form>
  )
}

export default Form
```

Lab Activities : 19

Create a react app with the conditional rendering based on component state in React.

//Toggler.jsx:

```
import import React, { useState } from 'react';

const Toggler = () => {
  // Define state variable to toggle visibility
  const [isVisible, setIsVisible] = useState(true);

  const toggleVisibility = () => {
    setIsVisible(!isVisible);
  };

  return (
    <div>
      <h1>Conditional Rendering Example</h1>
      <button onClick={toggleVisibility}>Toggle Visibility</button>
      {isVisible ? (
        <p>The component is visible!</p>
      ) : (
        <p>The component is hidden!</p>
      )}
    </div>
  );
};
export default Toggler;
```

// App component

```
import './Toggler.jsx'

const App = () => {
  return (
    <div>
      <h1>Condition Randering example:</h1>
      <Toggler/>
    </div>
  );
};

export default App
```

Lab Activities : 20

Create a simple React application using a component-driven approach. Demonstrate a application with multiple components. Components such as Header, Sidebar, Content, Footer etc. Ensure that each component is responsible for a specific UI element or functionality.

```
import React from 'react';
// Header component:
const Header = () => {
  return (
    <header>
      <h1>This is a header of our Website</h1>
    </header>
  );};
// Sidebar component:
const Sidebar = () => {
  return (
    <aside>
      <h2>Sidebar</h2>
      <ul>
        <li>Home</li>
        <li>About</li>
        <li>Contact</li>
      </ul>
    </aside>
 );};
// Content component:
const Content = () => {
  return (
    <main>
      <h2>Content</h2>
      <p>Welcome to my website!</p>
    </main>
  );};
// Footer component
const Footer = () => {
  return (
    <footer>
      <p>&copy; 2024 My Website</p>
    </footer>
  );};
```

// App component:

```
const App = () => {
  return (
    <div>
      <Header />
      <div className="container">
        <Sidebar />
        <Content />
      </div>
      <Footer />
    </div>
  );};
export default App;
```

/* App.css */

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

.header {
  background-color: #333;
  color: #fff;
  padding: 20px;
  text-align: center;
}

.sidebar {
  background-color: #f4f4f4;
  padding: 20px;
}

.content {
  flex-grow: 1;
  padding: 20px;
}

.footer {
  background-color: #333;
  color: #fff;
  padding: 20px;
  text-align: center;
}
```

Lab Activities : 21

Create a new react app. Install React Router via npm and configure basic routing for multiple pages. Create separate components for each page and link them using React Router's <Link> component.

```
//About.jsx:
import React from 'react'

const About = () => {
  return (
    <div> This is a about functional component</div>
  )
}
export default About

//Main.jsx:
import React from 'react'

const Main = () => {
  return (
    <div> This is a about Main component</div>
  )
}
export default Main

//Products.jsx:
import React from 'react'

const Products = () => {
  return (
    <div> This is a about Product component</div>
  )
}
export default Products

//App.js:
import React from 'react'
import {Routes ,Route, Link} from 'react-router-dom';
import Main from './Component/Main';
import About from './Component/About';
import Product from './Component/Product';
```

```
const App = () => {
  return (
    <div>
      <nav>
        <ul>
          <li><Link  to='/'>Main</Link> </li>
          <li><Link to='/About'>About</Link> </li>
          <li><Link to='/Product'>Product</Link> </li>
        </ul>
      </nav>

      <Routes>
        <Route path='/' element={<Main/>}></Route>
        <Route path='/About' element={<About/>}></Route>
        <Route path='/Product' element={<Product/>}></Route>
      </Routes>
    </div>
  )
}
export default App
```

Lab Activities : 22

Showcase each react hooks using react functionality and differentiate scenarious for each hooks usage.

```
// useState:
import React,{useState} from 'react'

const UseState = () => {

    const [count,setCount] = useState(0);


    const Increase = () =>{
        setCount(count + 1)
    }
    const Decrease = () =>{
        setCount(count - 1)
    }

  return (
    <div>
        <center>
        <h1>Counter:{count}</h1>
        <button onClick={Increase}>Increase Counter</button>
        <button onClick={Decrease}>Decrease Counter</button>
        </center>
    </div>
  )
}
export default UseState

// useEffect:
import React from 'react'
import { useEffect,useState } from 'react'
const UseEffect = () => {

    const [count, setCount] = useState(0)

    useEffect(()=>{

        const fetchData = async () =>{
            const api = await fetch('https://jsonplaceholder.typi-
code.com/todos')
```

```
            const result = await api.json()
            console.table(result)
        }
        fetchData()

    },[])

  return (
    <div>
        <h1>UseEffect Hook:</h1>
        <h2>{count}</h2>
        <button onClick={()=>{setCount(count + 1)}}>Count In-
crease</button>
    </div>
  )
}

export default UseEffect
```

// useRef:

```
import React, { useEffect, useRef, useState } from 'react'

const UseReff= () => {

        const[ count,setCount]=useState(0);
        const PrevCount=useRef('');

        useEffect(()=>{
            PrevCount.current=count;
        })

        function handle()
        {
            setCount(PrevCount=>PrevCount+1);
        }
  return (
      <div>
        <h1> PrevCount : {PrevCount.current}</h1>
        <h2> Count : {count}</h2>
         <button onClick={handle}>increament</button>
    </div>
  )}

export default UseReff
```