**Week 7: React Hands-On:**

**9.React:**

**Code:**

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8" />

<title>ReactJS ES6 Lab Solution</title>

</head>

<body>

<h1>ES6 Hands-on Lab - Console Output Below</h1>

<p>Open your browser console (F12 → Console) to see results.</p>

<script>

// 1. Features of ES6

// let and const

let currentYear = 2025;

const PI = 3.14159;

// Arrow function

const multiply = (a, b) => a \* b;

console.log("Arrow function multiply(5,4):", multiply(5, 4));

// Template literals

const userName = "Alice";

console.log(`Welcome, ${userName}!`);

// Default parameters

function greet(name = "Guest") {

return `Hello, ${name}!`;

}

console.log(greet());

console.log(greet("Bob"));

// Spread and Rest operators

const nums = [1, 2, 3];

const extendedNums = [...nums, 4, 5];

console.log("Spread operator result:", extendedNums);

function sumAll(...args) {

return args.reduce((total, num) => total + num, 0);

}

console.log("Rest operator sumAll(1,2,3,4):", sumAll(1, 2, 3, 4));

// Destructuring

const person = { firstName: "John", lastName: "Doe" };

const { firstName, lastName } = person;

console.log("Destructured values:", firstName, lastName);

// Classes

class Vehicle {

constructor(type) {

this.type = type;

}

getType() {

return `This is a ${this.type}`;

}

}

const car = new Vehicle("car");

console.log(car.getType());

// Promise (with async/await demonstration)

const fetchData = () =>

new Promise((resolve) => setTimeout(() => resolve("Data fetched!"), 700));

fetchData().then((message) => console.log("Promise resolved:", message));

async function asyncDemo() {

const msg = await fetchData();

console.log("Async/Await result:", msg);

}

asyncDemo();

// Map and Set

const mySet = new Set([10, 20, 20, 30]);

console.log("Set contents:", [...mySet]);

const myMap = new Map();

myMap.set("name", "Alice");

console.log("Map value for name:", myMap.get("name"));

// 2. Differences between var and let

if (true) {

var varVariable = "I'm var, accessible outside block";

let letVariable = "I'm let, only inside block";

}

console.log("var variable:", varVariable);

// console.log(letVariable); // Uncommenting will cause ReferenceError

// Redeclaration

var a = 1;

var a = 2; // Allowed

console.log("Redeclared var a:", a);

let b = 1;

// let b = 2; // Uncommenting causes SyntaxError

// 3. Explanation of const

const constantValue = 42;

console.log("Constant value:", constantValue);

// constantValue = 50; // Uncommenting will cause error

const objConst = { prop: "initial" };

objConst.prop = "changed"; // This is allowed

console.log("Modified const object prop:", objConst.prop);

// objConst = {}; // Uncommenting will cause error

// 4. ES6 Class fundamentals

class Animal {

constructor(name) {

this.name = name;

}

speak() { return `${this.name} makes a sound.`;

}

}

class Dog extends Animal {

constructor(name) {

super(name);

}

speak() {

return `${this.name} barks.`;

}

}

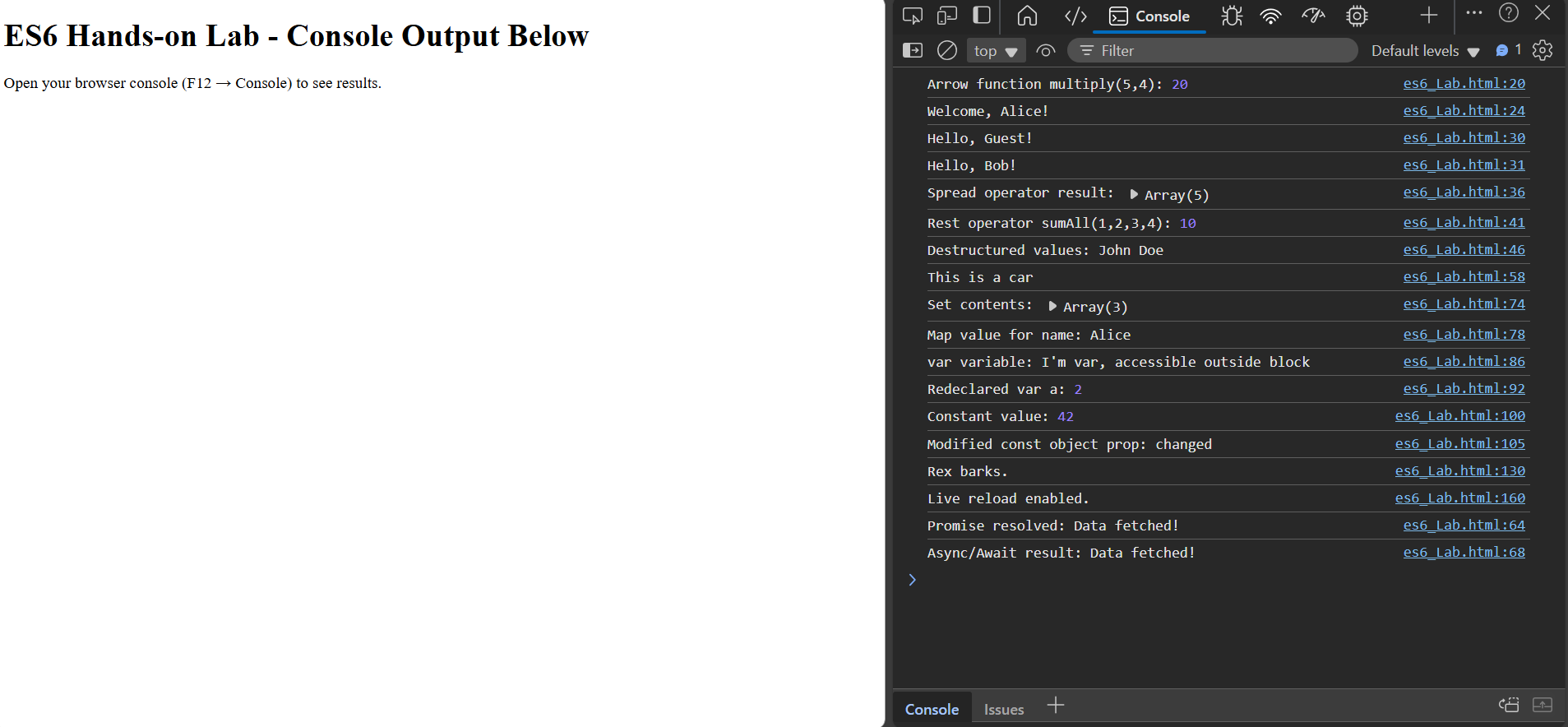
const myDog = new Dog("Rex");

console.log(myDog.speak());

</script>

</body>

</html>

**Output:**

**10.React:**

**Code:**

## Vanilla JavaScript Version

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8" />

<title>Office Details Lab</title>

</head>

<body>

<div id="root"></div>

<script src="https://unpkg.com/react@18/umd/react.development.js"></script>

<script src="https://unpkg.com/react-dom@18/umd/react-dom.development.js"></script>

<script>

const office = {

name: "Tech Hub",

rent: 50000,

address: "123 React Lane, City"

};

const element = React.createElement(

"div",

null,

React.createElement("h2", null, "Office Details"),

React.createElement("p", null, "Name: " + office.name),

React.createElement("p", null, "Rent: " + office.rent),

React.createElement("p", null, "Address: " + office.address)

);

const root = ReactDOM.createRoot(document.getElementById('root'));

root.render(element);

</script>

</body>

</html>

## [App.js](http://app.js):

function OfficeInfo() {

const office = {

name: "Tech Hub",

rent: 50000,

address: "123 React Lane, City"

};

return (

<div>

<h2>Office Details</h2>

<p><strong>Name:</strong> {office.name}</p>

<p><strong>Rent:</strong> {office.rent}</p>

<p><strong>Address:</strong> {office.address}</p>

</div>

);

}

export default OfficeInfo;

**Output:**

Office Details

Name: Tech Hub

Rent: 50000

Address: 123 React Lane, City

**11.React:**

[App.js](http://app.js):

import React, { useState } from "react";

function EventDemo() {

const [message, setMessage] = useState("");

const [color, setColor] = useState("black");

// Handler to show alert on button click

function handleClick() {

alert("Button was clicked!");

}

// Handler to display input text and demonstrate SyntheticEvent

function handleInputChange(event) {

// You can access SyntheticEvent properties like event.target.value

setMessage(event.target.value);

// To demonstrate event.persist(), uncomment the following:

// event.persist();

// setTimeout(() => {

// console.log("Event value after timeout:", event.target.value);

// }, 1000);

}

// Handler to change color on mouse enter and leave

function handleMouseEnter() {

setColor("blue");

}

function handleMouseLeave() {

setColor("black");

}

return (

<div>

<h2>React Event and SyntheticEvent Demo</h2>

{/\* Button with onClick event \*/}

<button onClick={handleClick}>Click Me</button>

<br /><br />

{/\* Input with onChange event - SyntheticEvent \*/}

<input

type="text"

placeholder="Type something..."

onChange={handleInputChange}

/>

<p>Input value: {message}</p>

<br />

{/\* Div that changes color on mouse events \*/}

<div

onMouseEnter={handleMouseEnter}

onMouseLeave={handleMouseLeave}

style={{

width: "200px",

height: "50px",

border: "1px solid #ccc",

color: color,

lineHeight: "50px",

textAlign: "center",

userSelect: "none",

}}

>

Hover over me!

</div>

</div>

);

}

export default EventDemo;

**Output:**

React Event and SyntheticEvent Demo

[Click Me]

[ Type something... ]

Input value: Hello

[ Hover over me! ] (text turns blue on hover)

React Event and SyntheticEvent Demo

[Click Me]

[ Type something... ]

Input value: Hello

[ Hover over me! ] (text turns blue on hover)

**12.React:**

[**App.js**](http://app.js)**:**

import React, { useState } from "react";

// User and Guest page components

function UserPage({ onLogout }) {

return (

<div>

<h2>Welcome, User!</h2>

<button onClick={onLogout}>Logout</button>

</div>

);

}

function GuestPage({ onLogin }) {

return (

<div>

<h2>Welcome, Guest!</h2>

<button onClick={onLogin}>Login</button>

</div>

);

}

// Main App with conditional rendering

function App() {

const [isLoggedIn, setIsLoggedIn] = useState(false);

function handleLogin() {

setIsLoggedIn(true);

}

function handleLogout() {

setIsLoggedIn(false);

}

return (

<div style={{ textAlign: "center", padding: "50px" }}>

{/\* Conditionally render pages based on login state \*/}

{isLoggedIn ? (

<UserPage onLogout={handleLogout} />

) : (

<GuestPage onLogin={handleLogin} />

)}

</div>

);

}

export default App;

function SecretMessage({ show }) {

if (!show) return null;

return <div>Secret: You are seeing this!</div>;

}

// Usage:

<SecretMessage show={isLoggedIn} />

**Output:**

Welcome, Guest!

[ Login ]

Welcome, User!

[ Logout ]

**13.React:**

**Code:**

[**App.js**](http://app.js)**:**

import React, { useState } from "react";

// Simple List Component with keys and map()

function FruitList({ fruits }) {

return (

<ul>

{fruits.map((fruit, idx) => (

<li key={fruit.id}>{fruit.name}</li>

))}

</ul>

);

}

function App() {

// Demo array for the list

const [fruits, setFruits] = useState([

{ id: 1, name: "Apple" },

{ id: 2, name: "Banana" },

{ id: 3, name: "Cherry" }

]);

// Conditional rendering: show/hide the list

const [showList, setShowList] = useState(true);

return (

<div style={{ padding: "30px" }}>

<h2>Conditional Rendering & List Lab</h2>

<button onClick={() => setShowList(!showList)}>

{showList ? "Hide" : "Show"} List

</button>

{/\* Conditional rendering using && \*/}

{showList && (

<div>

<h3>Fruit List:</h3>

<FruitList fruits={fruits} />

</div>

)}

{/\* Render different content if the list is empty \*/}

{showList && fruits.length === 0 && (

<p>No fruits to display.</p>

)}

</div>

);

}

export default App;

**Output:**

Conditional Rendering & List Lab

[ Hide List ]

Fruit List:

• Apple

• Banana

• Cherry

Conditional Rendering & List Lab

[ Show List ]

Conditional Rendering & List Lab

[ Hide List ]

No fruits to display.