DAY-5_week_1_ Hands-On<Rushabh Ramesh Bhusanur>

Problem 1:

Code:

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Problem 1 - Greeting Generator</title>
</head>
<body>

  <h2>Greeting Generator</h2>

  <!-- Requirement: Create an input box to enter user’s name -->
  <input type="text" id="nameInput" placeholder="Enter your name">

  <!-- Requirement: Create a button labeled “Generate Greeting” -->
  <button id="greetBtn">Generate Greeting</button>

  <!-- Requirement: Display greeting inside a <p> element -->
  <p id="greetingOutput"></p>

  <script>
    // Requirement: Use function keyword (not arrow function)
    // Requirement: Function should accept name as parameter
    function generateGreeting(userName) {

      // Requirement: Variable for greeting must be declared inside function (local scope)
      let greetingMessage = "Hello, " + userName + "! Welcome to our website.";
```

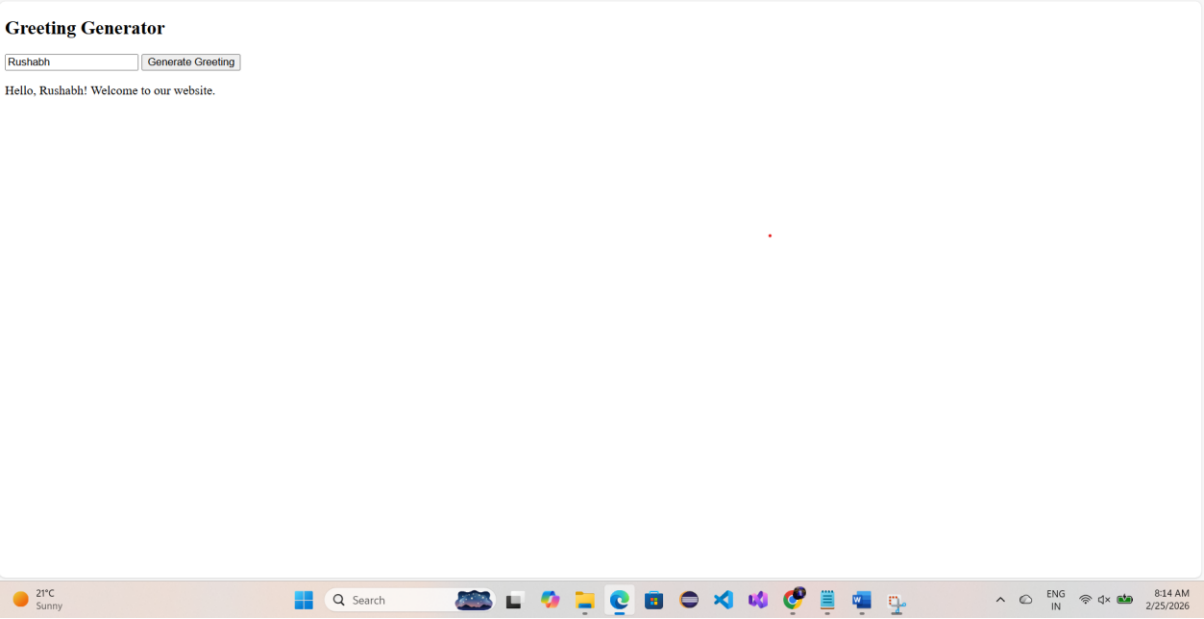
```
    // Requirement: Use document.getElementById() for DOM manipulation
    document.getElementById("greetingOutput").innerText = greetingMessage;
  }

  // Requirement: Button click should call function
  document.getElementById("greetBtn").addEventListener("click", function () {
    let name = document.getElementById("nameInput").value;
    generateGreeting(name);
  });
</script>

</body>
</html>
```

Code Snapshot:

File Edit View

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Problem 1 - Greeting Generator</title>
</head>
<body>

  <h2>Greeting Generator</h2>

  <!-- Requirement: Create an input box to enter user's name -->
  <input type="text" id="nameInput" placeholder="Enter your name">

  <!-- Requirement: Create a button labeled "Generate Greeting" -->
  <button id="greetBtn">Generate Greeting</button>

  <!-- Requirement: Display greeting inside a <p> element -->
  <p id="greetingOutput"></p>

  <script>
    // Requirement: Use function keyword (not arrow function)
    // Requirement: Function should accept name as parameter
    function generateGreeting(userName) {

      // Requirement: Variable for greeting must be declared inside function (local scope)
      let greetingMessage = "Hello, " + userName + "! Welcome to our website.";

      // Requirement: Use document.getElementById() for DOM manipulation
      document.getElementById("greetingOutput").innerText = greetingMessage;
    }

    // Requirement: Button click should call function
    document.getElementById("greetBtn").addEventListener("click", function () {
      let name = document.getElementById("nameInput").value;
      generateGreeting(name);
    });
  </script>

</body>
</html>
```

Output Snapshot:



Code Explanation:

This HTML page creates a simple “Greeting Generator” where the user enters their name in a text input box and clicks the “Generate Greeting” button to see a personalized message. When the button is clicked, an event listener runs, fetches the value entered in the input field using document.getElementById(), and passes that value to the generateGreeting() function. Inside this function, a local variable greetingMessage is created to build the greeting text using the provided name, and then the DOM is updated by setting the text of the <p> element with id greetingOutput to show the greeting on the webpage. In short, the code demonstrates how to take user input, process it using a function, and dynamically update the webpage using JavaScript DOM manipulation.

Problem 2:

Code:

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Problem 2 - User Info Display</title>
</head>
<body>

  <h2>User Information</h2>

  <!-- Display areas -->
  <p id="name"></p>
  <p id="age"></p>
  <p id="city"></p>

  <!-- Requirement: Button click should trigger function -->
  <button id="showBtn">Show User Info</button>

  <script>
    // Requirement: Create a JavaScript object user
    let user = {
      name: "Rahul",
      age: 22,
      city: "Mumbai"
    };

    // Requirement: Function accepts object as parameter
    function displayUserInfo(userObj) {
```

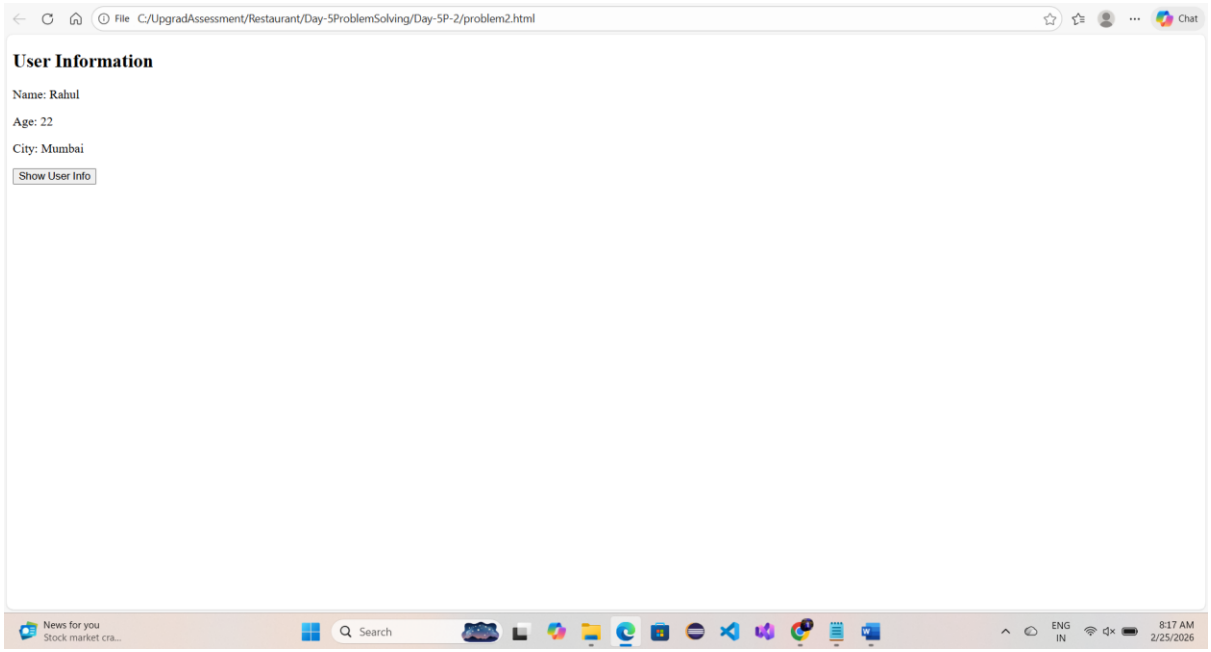
```
    // Requirement: Access object properties using dot notation
    document.getElementById("name").innerText = "Name: " + userObj.name;
    document.getElementById("age").innerText = "Age: " + userObj.age;
    document.getElementById("city").innerText = "City: " + userObj.city;
  }

  // Requirement: Function should not use global variables directly
  document.getElementById("showBtn").addEventListener("click", function () {
    displayUserInfo(user);
  });
</script>

</body>
</html>
```

Code Snapshot:

File Edit View

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Problem 2 - User Info Display</title>
</head>
<body>

  <h2>User Information</h2>

  <!-- Display areas -->
  <p id="name"></p>
  <p id="age"></p>
  <p id="city"></p>

  <!-- Requirement: Button click should trigger function -->
  <button id="showBtn">Show User Info</button>

  <script>
    // Requirement: Create a JavaScript object user
    let user = {
      name: "Rahul",
      age: 22,
      city: "Mumbai"
    };

    // Requirement: Function accepts object as parameter
    function displayUserInfo(userObj) {

      // Requirement: Access object properties using dot notation
      document.getElementById("name").innerText = "Name: " + userObj.name;
      document.getElementById("age").innerText = "Age: " + userObj.age;
      document.getElementById("city").innerText = "City: " + userObj.city;
    }

    // Requirement: Function should not use global variables directly
    document.getElementById("showBtn").addEventListener("click", function () {
      displayUserInfo(user);
    });
  </script>
```

Output Snapshot:



Code Explanation:

This code builds a simple web page that displays user details (name, age, and city) when a button is clicked. A JavaScript object called user is created to store the user's information, and the function displayUserInfo() is designed to accept this object as a parameter instead of directly using global variables, which makes the function reusable and cleaner. When the "Show User Info" button is clicked, an event listener calls the function and passes the user object to it. Inside the function, the object's properties are accessed using dot notation (userObj.name, userObj.age, userObj.city) and the values are dynamically inserted into the respective <p> elements using document.getElementById() to update the DOM, so the information appears on the webpage.

Problem 3:

Code:

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Problem 3 - Counter App</title>
</head>
<body>

  <h2>Counter App</h2>

  <!-- Requirement: Display counter starting from 0 -->
  <p id="counterDisplay">0</p>

  <!-- Requirement: Two buttons Increment and Reset -->
  <button id="incBtn">Increment</button>
  <button id="resetBtn">Reset</button>

  <script>
    // Requirement: Global variable to store counter value
    let counter = 0;

    // Requirement: Function accepts step as parameter
    function incrementCounter(step) {
      counter = counter + step;

      // Requirement: DOM update must happen inside function
      document.getElementById("counterDisplay").innerText = counter;
    }
```

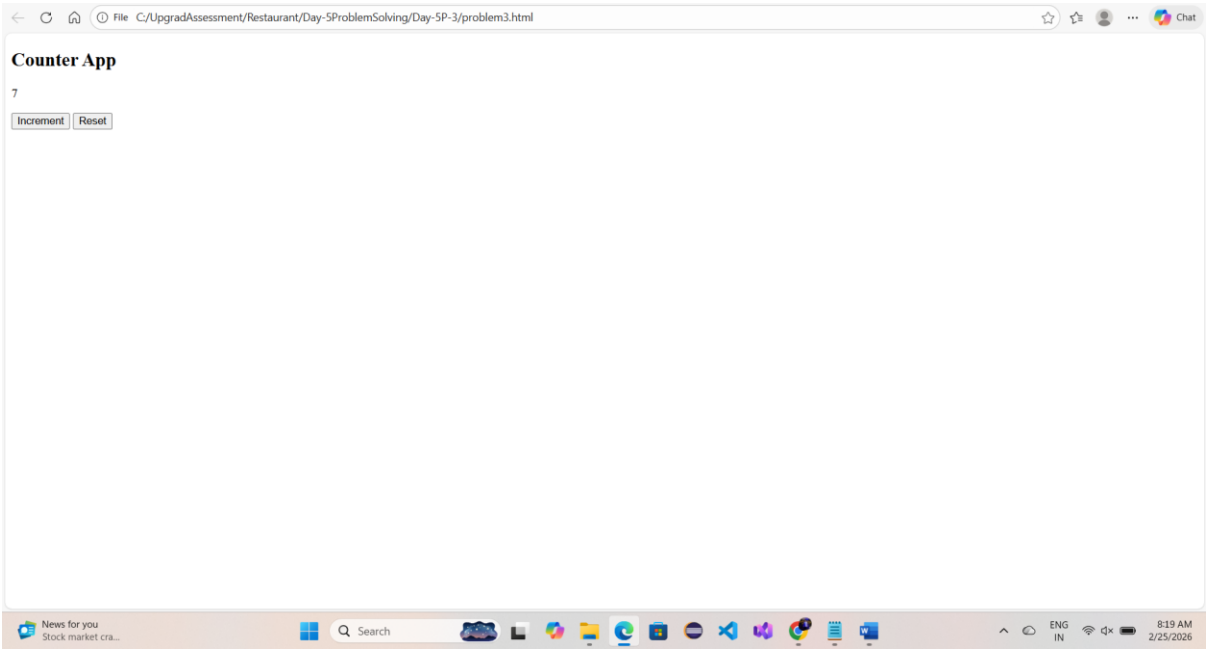
```
    function resetCounter() {
      counter = 0;
      document.getElementById("counterDisplay").innerText = counter;
    }

    // Requirement: No inline JavaScript in HTML
    document.getElementById("incBtn").addEventListener("click", function () {
      incrementCounter(1);
    });

    document.getElementById("resetBtn").addEventListener("click", function () {
      resetCounter();
    });
  </script>

</body>
</html>
```

Code Snapshot:

File Edit View

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Problem 3 - Counter App</title>
</head>
<body>

  <h2>Counter App</h2>

  <!-- Requirement: Display counter starting from 0 -->
  <p id="counterDisplay">0</p>

  <!-- Requirement: Two buttons Increment and Reset -->
  <button id="incBtn">Increment</button>
  <button id="resetBtn">Reset</button>

  <script>
    // Requirement: Global variable to store counter value
    let counter = 0;

    // Requirement: Function accepts step as parameter
    function incrementCounter(step) {
      counter = counter + step;

      // Requirement: DOM update must happen inside function
      document.getElementById("counterDisplay").innerText = counter;
    }

    function resetCounter() {
      counter = 0;
      document.getElementById("counterDisplay").innerText = counter;
    }

    // Requirement: No inline JavaScript in HTML
    document.getElementById("incBtn").addEventListener("click", function () {
      incrementCounter(1);
    });

    document.getElementById("resetBtn").addEventListener("click", function () {
```

Output Snapshot:



Code Explanation:

This code creates a simple Counter App that starts from 0 and lets the user increment the value or reset it back to zero using two buttons. A global variable counter is used to store the current count, and the function incrementCounter(step) increases this value by the given step (here, 1) whenever the Increment button is clicked. The function also updates the displayed number on the webpage by changing the text of the <p> element using document.getElementById(), so the UI always reflects the latest counter value. The resetCounter() function sets the counter back to 0 and updates the display in the same way. Event listeners are attached to both buttons (instead of using inline JavaScript in HTML), which trigger the appropriate functions when clicked, demonstrating clean separation between HTML structure and JavaScript behavior.

Problem 4:

Code:

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Problem 4 - Student Profile</title>

  <style>
    /* Requirement: Display details in a styled <div> */
    #profileBox {
      border: 2px solid #4CAF50;
      padding: 15px;
      width: 250px;
      margin-top: 10px;
    }
  </style>
</head>
<body>

  <h2>Student Profile</h2>

  <div id="profileBox"></div>

  <button id="showProfileBtn">Show Profile</button>
  <button id="updateMarksBtn">Update Marks</button>

  <script>
    // Requirement: Student object in global scope
    let student = {
      name: "Amit",
```

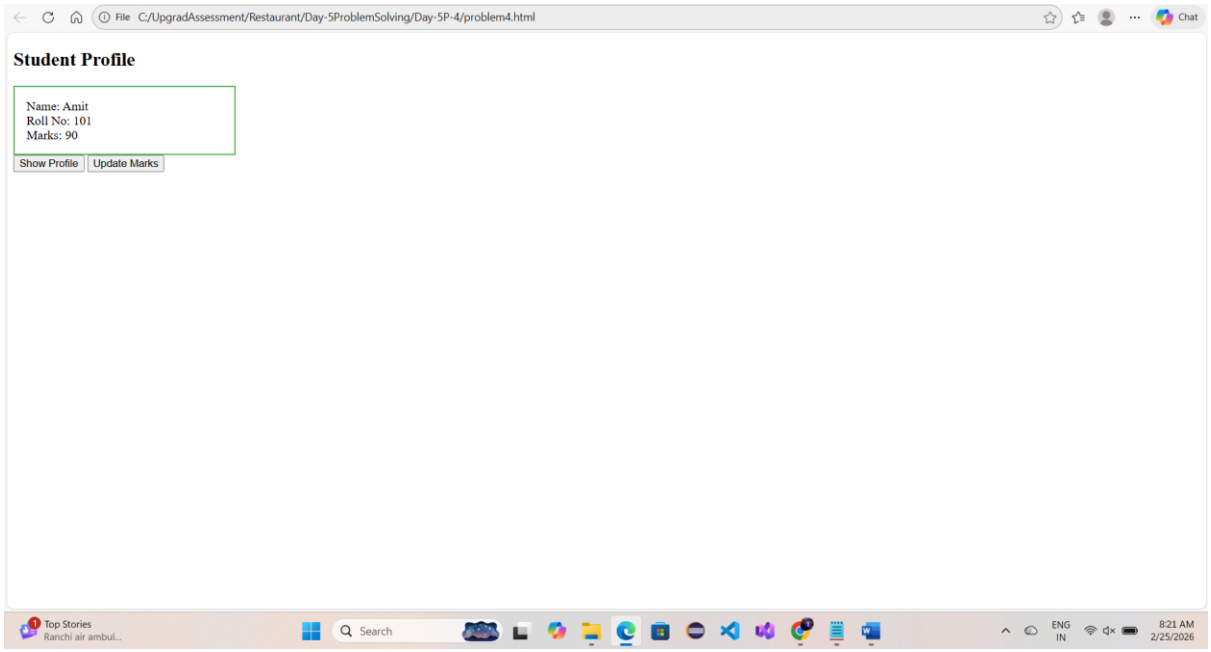
```
    rollNo: 101,
    marks: 75
  };

  // Requirement: Function accepts object as parameter
  function updateStudentProfile(studentObj) {
    document.getElementById("profileBox").innerHTML =
      "Name: " + studentObj.name + "<br>" +
      "Roll No: " + studentObj.rollNo + "<br>" +
      "Marks: " + studentObj.marks;
  }

  // Requirement: Function updates marks using parameter
  function updateMarks(newMarks) {
    student.marks = newMarks;
    updateStudentProfile(student);
  }

  document.getElementById("showProfileBtn").addEventListener("click", function () {
    updateStudentProfile(student);
  });

  document.getElementById("updateMarksBtn").addEventListener("click", function () {
    updateMarks(90);
  });
 </script>

</body>
</html>
```

Code Snapshot:

File Edit View

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Problem 4 - Student Profile</title>

  <style>
    /* Requirement: Display details in a styled <div> */
    #profileBox {
      border: 2px solid #4CAF50;
      padding: 15px;
      width: 250px;
      margin-top: 10px;
    }
  </style>
</head>
<body>

  <h2>Student Profile</h2>

  <div id="profileBox"></div>

  <button id="showProfileBtn">Show Profile</button>
  <button id="updateMarksBtn">Update Marks</button>

  <script>
    // Requirement: Student object in global scope
    let student = {
      name: "Amit",
      rollNo: 101,
      marks: 75
    };

    // Requirement: Function accepts object as parameter
    function updateStudentProfile(studentObj) {
      document.getElementById("profileBox").innerHTML =
        "Name: " + studentObj.name + "<br>" +
        "Roll No: " + studentObj.rollNo + "<br>" +
        "Marks: " + studentObj.marks;
    }
```

Output Snapshot:



Code Explanation:

This code builds a simple Student Profile page where a student's details (name, roll number, and marks) are displayed inside a styled when the "Show Profile" button is clicked, and the marks can be updated using the "Update Marks" button. A global student object holds the data, and the function updateStudentProfile() accepts this object as a parameter and updates the UI by inserting the values into the profileBox using DOM manipulation. The updateMarks(newMarks) function demonstrates how data can be modified through a function parameter by changing the marks property of the student object and then immediately refreshing the displayed profile. Event listeners are used instead of inline HTML events to trigger these functions on button clicks, showing a clean separation of structure (HTML), style (CSS), and behavior (JavaScript).