JAVASCRIPT-DAY-5-Hands-on-Srinu

Problem 1: Greeting Generator (Level-1)

1.Problem Statement

A simple webpage should greet users based on the name they enter. This helps beginners understand how JavaScript functions accept parameters and how variables behave inside functions.

Requirements Create an input box to enter a user's name. Create a button labeled "Generate Greeting". When the button is clicked: A JavaScript function should accept the name as a parameter. Display a greeting message like:
"Hello, Rahul! Welcome to our website."

Display the greeting inside a <p> element.

Technical Constraints Use only vanilla JavaScript (no libraries). Use function keyword (not arrow functions for this task). Use document.getElementById() for DOM manipulation.

Variable for the greeting message must be declared inside the function.

Learning Outcome

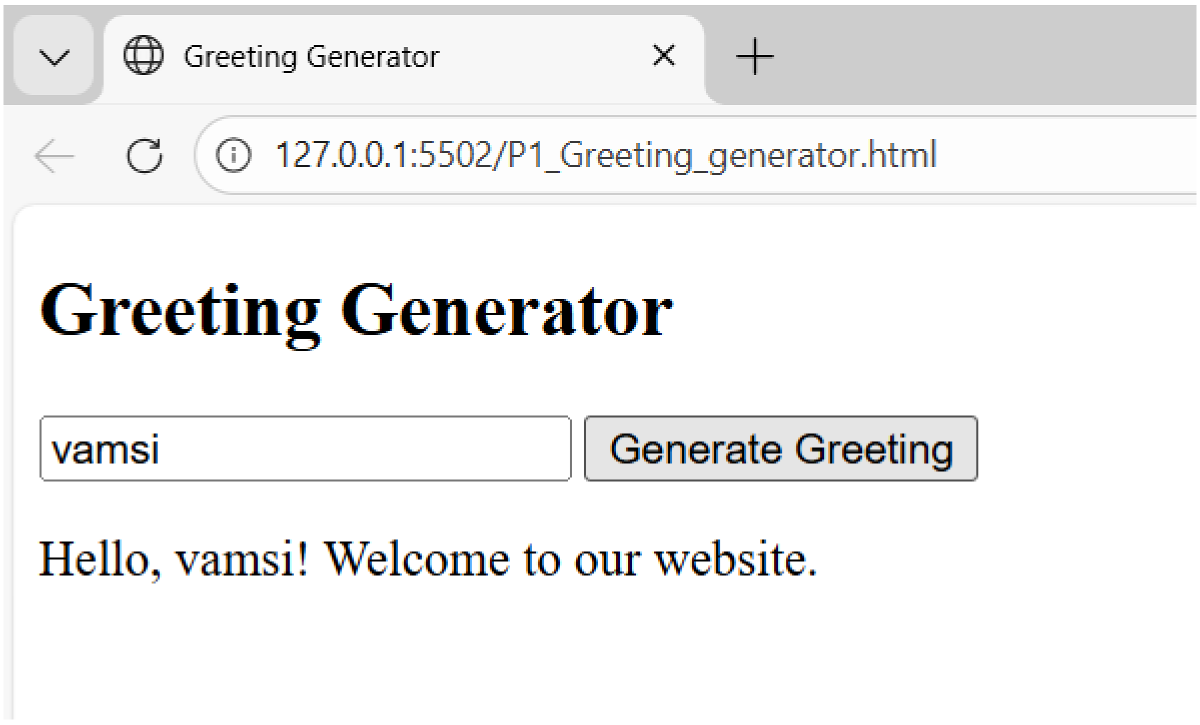
You should be able to: Understand how functions receive input using parameters. Learn local variable scope. Manipulate webpage elements using the DOM.

2. Code

```
<!DOCTYPE html>

<html>

<head>

  <title>Greeting Generator</title>

</head>

<body>


  <h2>Greeting Generator</h2>


  <!-- Input Box -->

  <input type="text" id="username" placeholder="Enter your name">


  <!-- Button -->
```

```
    <button onclick="generateGreeting()">Generate Greeting</button>

    <!-- Paragraph to display greeting -->
    <p id="greetingMessage"></p>

    <script>
        // Function using function keyword
        function generateGreeting() {

            // Get value from input box
            var name = document.getElementById("username").value;

            // Greeting variable declared INSIDE the function (local scope)
            var greeting;

            if (name.trim() === "") {
                greeting = "Please enter your name.";
            } else {
                greeting = "Hello, " + name + "! Welcome to our website.";
            }

            // Display greeting inside <p>
            document.getElementById("greetingMessage").innerText = greeting;
        }
    </script>

</body>
</html>
```

3.Output Screenshot



Greeting Generator

vamsi
Generate Greeting

Hello, vamsi! Welcome to our website.

4.Code Explanation

<!DOCTYPE html> → Declares the document as HTML5.

<html> → Root element of the webpage.

<head> → Contains metadata and title of the webpage.

<title>Greeting Generator</title> → Sets the title shown in the browser tab.

<body> → Contains visible content of the webpage.

<h2>Greeting Generator</h2> → Displays the heading on the webpage.

<input type="text" id="username" placeholder="Enter your name"> → Creates a text input box to enter the user's name.

type="text" → Specifies that the input field accepts text.

id="username" → Assigns a unique ID to access this element using JavaScript.

placeholder="Enter your name" → Displays hint text inside the input box.

<button onclick="generateGreeting()"> → Creates a button that calls the function when clicked.

onclick="generateGreeting()" → Triggers the generateGreeting function on button click.

<p id="greetingMessage"></p> → Paragraph element used to display the greeting message.

id="greetingMessage" → Allows JavaScript to access and modify this paragraph.

<script> → Starts the JavaScript section.

function generateGreeting() → Declares a function using the function keyword.

var name = document.getElementById("username").value; → Gets the value entered in the input box.

document.getElementById("username") → Selects the input element by ID.

.value → Retrieves the entered text from the input field.

var greeting; → Declares a local variable to store the greeting message.

if (name.trim() === "") → Checks if the input is empty after removing extra spaces.

name.trim() → Removes spaces from the beginning and end of the string.

greeting = "Please enter your name."; → Assigns message if input is empty.

else → Executes when input is not empty.

greeting = "Hello, " + name + "! Welcome to our website."; → Creates a personalized greeting message.

document.getElementById("greetingMessage").innerText = greeting; → Displays the greeting message inside the paragraph.

innerText → Updates only the text content of the element.

</script> → Ends the JavaScript section.

</body> → Ends the body section.

</html> → Ends the HTML document.

Problem 2: Simple Object-Based User Info Display (Level-1)

1.Problem Statement

A webpage needs to display basic user information stored inside a JavaScript object.

Requirements

Create a JavaScript object user with properties: name age city

Create a function displayUserInfo(userObj): Accepts the object as a parameter. Displays user details in separate <p> elements.

A button click should trigger the function.

Technical Constraints

Object properties must be accessed using dot notation.Function should not use global variables. DOM elements should be updated dynamically.

Learning Outcome

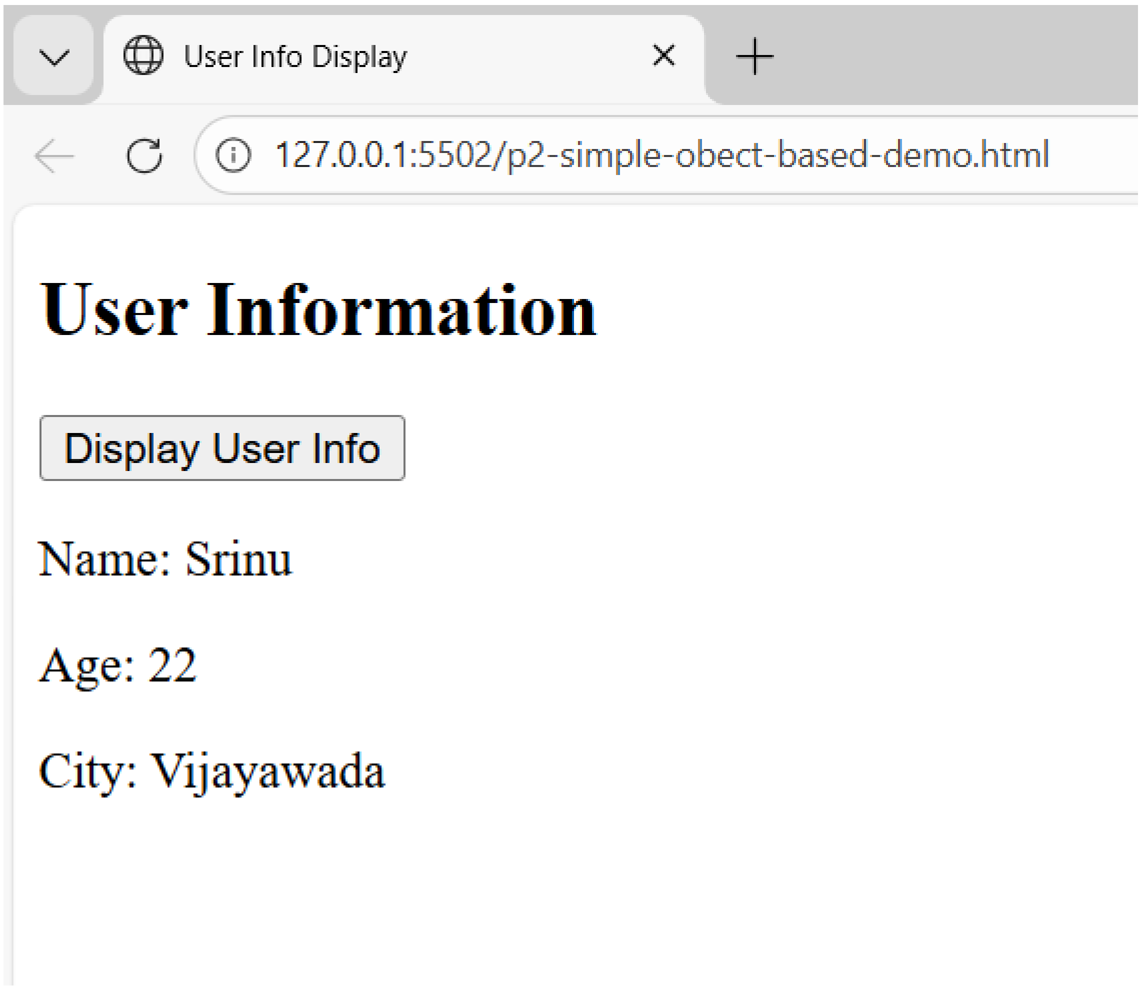
You will be able to: Understand JavaScript objects. Pass objects to functions. Display object data dynamically on a webpage.

2. Code

```
<!DOCTYPE html>

<html>

<head>

  <title>User Info Display</title>

</head>

<body>


  <h2>User Information</h2>


  <!-- Button to trigger function -->

  <button onclick="displayUserInfo(user)">Display User Info</button>


  <!-- Paragraph elements to display data -->

  <p id="name"></p>

  <p id="age"></p>
```

```
    <p id="city"></p>

    <script>

        // Requirement: Create user object
        var user = {
            name: "Srinu",
            age: 22,
            city: "Vijayawada"
        };

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

            // Access 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;
        }

    </script>

</body>
</html>
```

3. Output Screenshot



User Information

Name: Srinu

Age: 22

City: Vijayawada

4. Code Explanation

<!DOCTYPE html> → Declares the document as HTML5.

<html> → Root element of the webpage.

<head> → Contains title and metadata.

<title>User Info Display</title> → Sets the title shown in the browser tab.

<body> → Contains visible content of the webpage.

<h2>User Information</h2> → Displays the heading on the webpage.

<button onclick="displayUserInfo(user)"> → Creates a button that calls the function when clicked.

onclick="displayUserInfo(user)" → Passes the user object as an argument to the function.

<p id="name"></p> → Paragraph element to display the user's name.

<p id="age"></p> → Paragraph element to display the user's age.

<p id="city"></p> → Paragraph element to display the user's city.

<script> → Starts the JavaScript section.

var user = { ... } → Creates an object named user.

name: "Srinu" → Defines the name property inside the object.

age: 22 → Defines the age property inside the object.

city: "Vijayawada" → Defines the city property inside the object.

function displayUserInfo(userObj) → Declares a function that accepts an object as a parameter.

userObj → Parameter that receives the passed object.

document.getElementById("name").innerText = "Name: " + userObj.name; → Displays the name property using dot notation.

userObj.name → Accesses the name property from the object.

document.getElementById("age").innerText = "Age: " + userObj.age; → Displays the age property.

userObj.age → Accesses the age property.

document.getElementById("city").innerText = "City: " + userObj.city; → Displays the city property.

userObj.city → Accesses the city property.

innerText → Updates only the text content inside the paragraph element.

</script> → Ends the JavaScript section.

</body> → Ends the body section.

</html> → Ends the HTML document.

Problem 3: Counter App with Scope Control (Level-2)

1.Problem Statement

Build a counter application where users can increment and reset a value. This problem focuses on variable scope and DOM manipulation.

Requirements Display a counter value starting from 0.

Create two buttons: Increment Reset

Use: A global variable to store counter value. A function incrementCounter(step) that:

Accepts step value as a parameter. Updates the counter.

Reset button should reset the counter to 0.

Technical Constraints Counter value must be maintained outside the function (global scope).

DOM updates must happen inside functions only. No inline JavaScript in HTML.

Learning Outcome Learners should be able to: Understand variable scope (global vs local).

Apply function parameters in real scenarios. Control UI behavior using JavaScript functions.

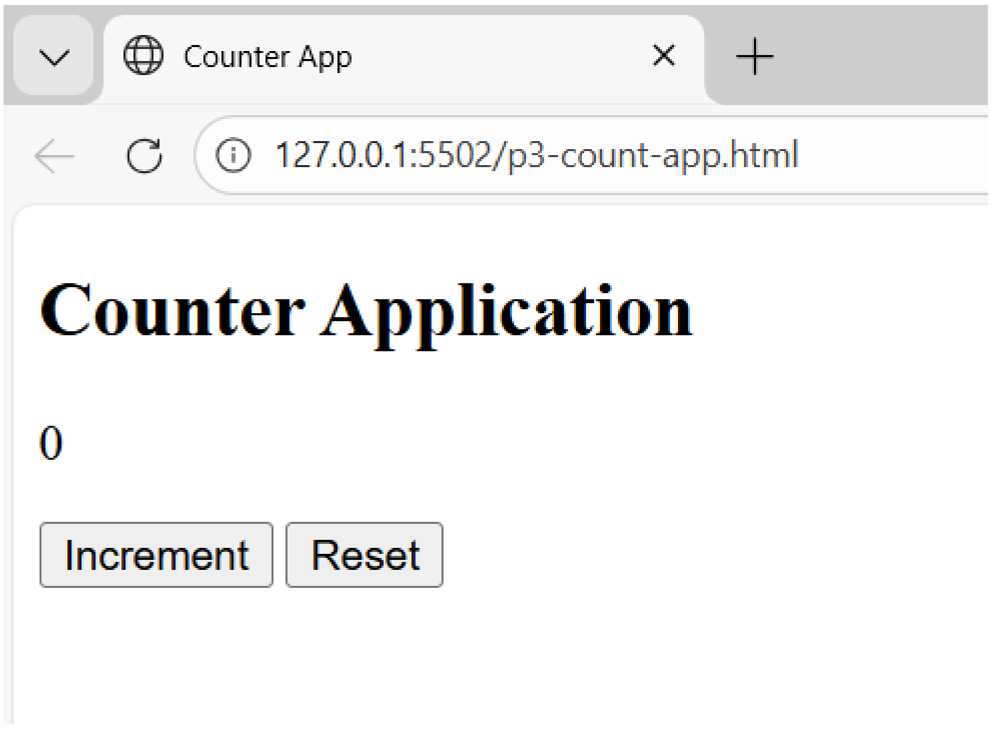
2.Code

```
<!DOCTYPE html>

<html>

<head>

  <title>Counter App</title>

</head>

<body>


  <h2>Counter Application</h2>


  <!-- Display Counter Value -->

  <p id="counterDisplay">0</p>


  <!-- Buttons (No inline JS used) -->

  <button id="incrementBtn">Increment</button>
```

```
<button id="resetBtn">Reset</button>

<script>

   //  Global variable (maintained outside functions)
   let counter = 0;

   // Get DOM elements
   const display = document.getElementById("counterDisplay");
   const incrementBtn = document.getElementById("incrementBtn");
   const resetBtn = document.getElementById("resetBtn");

   //  Function that accepts step parameter
   function incrementCounter(step) {
      counter += step;   // Update global counter
      display.innerText = counter;  // DOM update inside function
   }

   // Reset function
   function resetCounter() {
      counter = 0;
      display.innerText = counter;  // DOM update inside function
   }

   // Event Listeners (No inline JS)
   incrementBtn.addEventListener("click", function () {
      incrementCounter(1);   // Passing step value as parameter
   });
```

```
        resetBtn.addEventListener("click", function () {
          resetCounter();
        });

    </script>

</body>
</html>
```

3. Output Screenshot



Counter Application

0

4. Code Explanation

<!DOCTYPE html> → Declares the document as HTML5.

<html> → Root element of the webpage.

<head> → Contains metadata and title.

<title>Counter App</title> → Sets the title shown in the browser tab.

<body> → Contains visible content of the webpage.

<h2>Counter Application</h2> → Displays the heading on the webpage.

<p id="counterDisplay">0</p> → Paragraph element used to display the counter value, initially set to 0.

id="counterDisplay" → Unique identifier used to access this element in JavaScript.

<button id="incrementBtn">Increment</button> → Button used to increase the counter value.

<button id="resetBtn">Reset</button> → Button used to reset the counter value to 0.

<script> → Starts the JavaScript section.

let counter = 0; → Declares a global variable to store the counter value.

const display = document.getElementById("counterDisplay"); → Selects the paragraph element and stores it in a variable.

const incrementBtn = document.getElementById("incrementBtn"); → Selects the increment button.

const resetBtn = document.getElementById("resetBtn"); → Selects the reset button.

function incrementCounter(step) → Declares a function that accepts a parameter named step.

counter += step; → Increases the global counter value by the step amount.

display.innerText = counter; → Updates the displayed counter value in the DOM.

function resetCounter() → Declares a function to reset the counter.

counter = 0; → Sets the global counter value back to 0.

display.innerText = counter; → Updates the displayed value after reset.

incrementBtn.addEventListener("click", function () { ... }); → Adds a click event listener to the increment button.

incrementCounter(1); → Calls the increment function and passes 1 as the step value.

resetBtn.addEventListener("click", function () { ... }); → Adds a click event listener to the reset button.

resetCounter(); → Calls the reset function when reset button is clicked.

addEventListener() → Attaches an event to an element without using inline JavaScript.

</script> → Ends the JavaScript section.

</body> → Ends the body section.

</html> → Ends the HTML document.

Problem 4: Dynamic Student Profile Manager (Level-2)

1.Problem Statement

Create a mini student profile system where details are stored in an object and displayed dynamically using DOM manipulation.

Requirements Create a student object with: name rollNo marks

Create a function updateStudentProfile(studentObj): Accepts the object as a parameter.

Displays details in a styled <div>. Add another function updateMarks(newMarks):

Updates marks and refreshes the display.

Technical Constraints Student object must be declared in global scope.

Functions should update object values using parameters. DOM should update without page refresh.

Learning Outcome

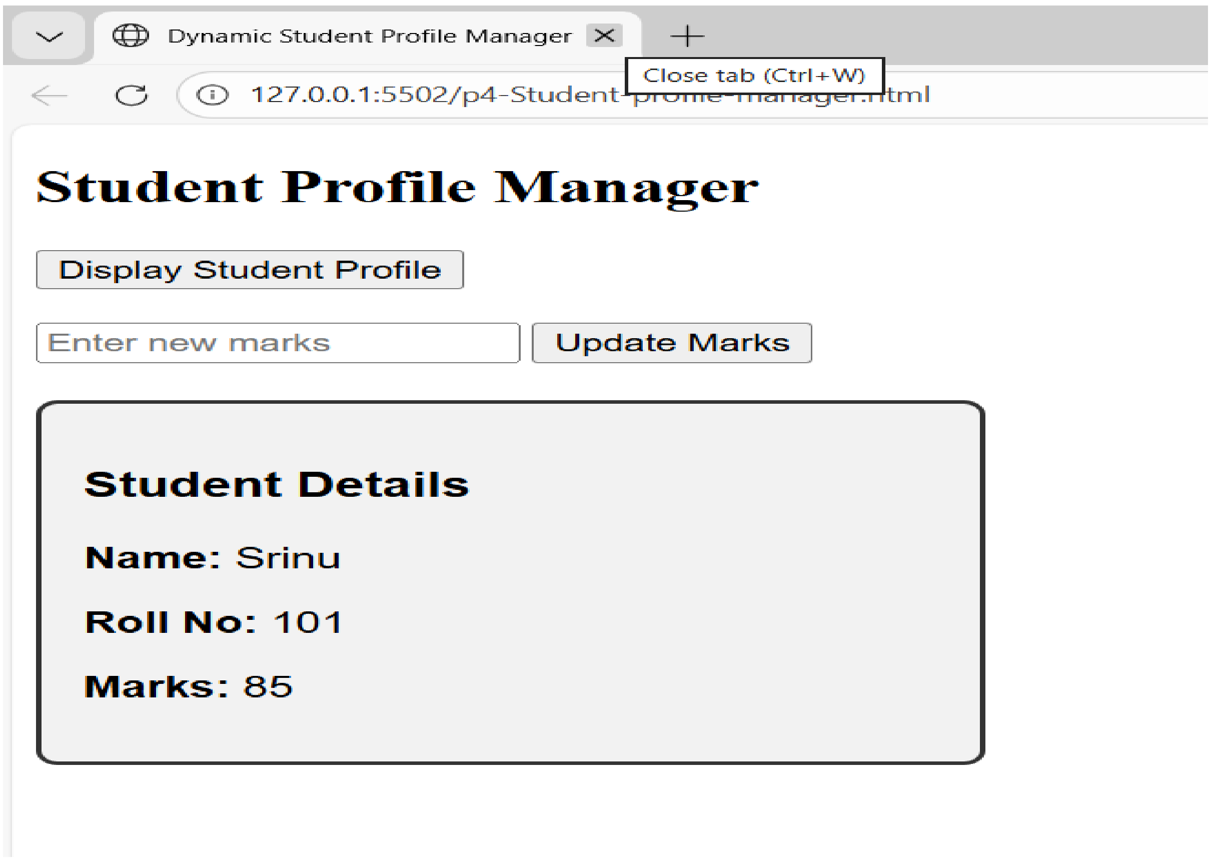
You will be able to: Deep understanding of object manipulation. Function interaction with shared data. Real-world use of DOM updates and scope handling.

2. Code

```
<!DOCTYPE html>

<html>

<head>

  <title>Dynamic Student Profile Manager</title>

  <style>

    /* Styled div for displaying profile */

    #profileContainer {

      border: 2px solid #333;

      padding: 15px;

      width: 300px;

      margin-top: 20px;

      background-color: #f2f2f2;

      border-radius: 8px;

      font-family: Arial, sans-serif;

    }
  </style>
</head>
<body>

  <h2>Student Profile Manager</h2>

  <!-- Button to display student profile -->
  <button onclick="updateStudentProfile(student)">
    Display Student Profile
  </button>

  <br><br>

  <!-- Input to update marks -->
  <input type="number" id="newMarksInput" placeholder="Enter new marks">
  <button onclick="updateMarksFromInput()">
    Update Marks
  </button>

  <!-- Styled div where student details will appear -->
  <div id="profileContainer"></div>

  <script>

    // ================================
    // Requirement: Student object in GLOBAL scope
    // ================================
    var student = {
      name: "Srinu",
      rollNo: "101",
      marks: 85
    };

    // ================================================
    // Requirement: Function accepts object parameter
    // Displays details in styled <div>
    // ================================================
    function updateStudentProfile(studentObj) {

      document.getElementById("profileContainer").innerHTML =
        "<h3>Student Details</h3>" +
        "<p><strong>Name:</strong> " + studentObj.name + "</p>" +
        "<p><strong>Roll No:</strong> " + studentObj.rollNo + "</p>" +
        "<p><strong>Marks:</strong> " + studentObj.marks + "</p>";
    }

    // ================================================
    // Requirement: Update marks using parameter
    // Update object value and refresh display
    // ================================================
    function updateMarks(newMarks) {

      student.marks = newMarks;  // Update object property

      // Refresh display without page reload
      updateStudentProfile(student);
    }
```

```
        // Helper function to get input value
        function updateMarksFromInput() {

            var newMarks = document.getElementById("newMarksInput").value;

            if (newMarks === "") {
                alert("Please enter marks");
                return;
            }

            updateMarks(Number(newMarks)); // Pass parameter
        }

    </script>

</body>
</html>
```

3. Output Screenshot


Dynamic Student Profile Manager
Close tab (Ctrl+W)
127.0.0.1:5502/p4-Student-profile-manager.html
Display Student Profile
Enter new marks
Update Marks

Student Profile Manager

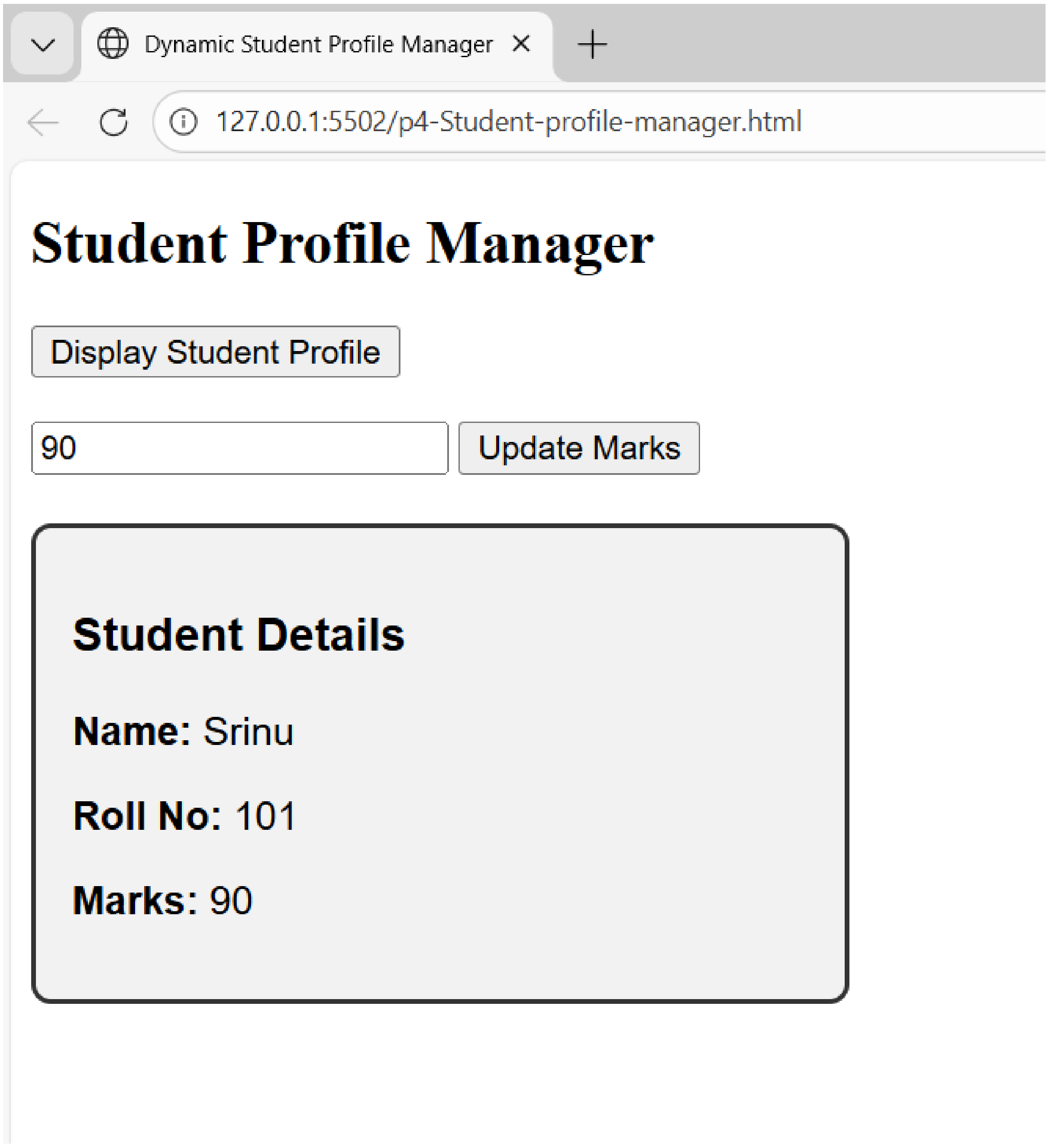
Student Details

Name: Srinu

Roll No: 101

Marks: 85

After updating the marks


Dynamic Student Profile Manager
127.0.0.1:5502/p4-Student-profile-manager.html
Display Student Profile
90
Update Marks

Student Profile Manager

Student Details

Name: Srinu

Roll No: 101

Marks: 90

4. Code Explanation

<!DOCTYPE html> → Declares the document as HTML5.

<html> → Root element of the webpage.

<head> → Contains title and internal CSS.

<title>Dynamic Student Profile Manager</title> → Sets the browser tab title.

<style> → Starts internal CSS styling.

#profileContainer → ID selector used to style the profile display div.

border: 2px solid #333; → Adds a solid border around the div.

padding: 15px; → Adds space inside the div between content and border.

width: 300px; → Sets fixed width for the div.

margin-top: 20px; → Adds space above the div.

background-color: #f2f2f2; → Sets light background color.

border-radius: 8px; → Creates rounded corners.

font-family: Arial, sans-serif; → Sets font style inside the div.

</style> → Ends CSS section.

<body> → Contains visible webpage content.

<h2>Student Profile Manager</h2> → Displays heading on the webpage.

<button onclick="updateStudentProfile(student)"> → Button that calls function and passes student object.

onclick="updateStudentProfile(student)" → Executes function when button is clicked.

<input type="number" id="newMarksInput"> → Input field to enter new marks.

type="number" → Allows only numeric input.

id="newMarksInput" → Used to access input value in JavaScript.

<button onclick="updateMarksFromInput()"> → Button to update marks using input value.

<div id="profileContainer"></div> → Empty div where student details will be displayed dynamically.

<script> → Starts JavaScript section.

var student = { name: "Srinu", rollNo: "101", marks: 85 }; → Creates a student object in global scope.

name: "Srinu" → Defines name property.

rollNo: "101" → Defines roll number property.

marks: 85 → Defines marks property.

function updateStudentProfile(studentObj) → Function that accepts an object as parameter.

studentObj → Parameter that receives the passed object.

document.getElementById("profileContainer").innerHTML = → Updates the styled div content dynamically.

innerHTML → Inserts HTML content inside the div.

"<h3>Student Details</h3>" → Adds subheading inside div.

studentObj.name → Accesses name property using dot notation.

studentObj.rollNo → Accesses roll number property.

studentObj.marks → Accesses marks property.

function updateMarks(newMarks) → Function that accepts new marks as parameter.

student.marks = newMarks; → Updates marks property of the global student object.

updateStudentProfile(student); → Refreshes the display without reloading the page.

function updateMarksFromInput() → Helper function to get input value from textbox.

document.getElementById("newMarksInput").value → Retrieves value entered in input field.

if (newMarks === "") → Checks if input is empty.

alert("Please enter marks"); → Displays alert message if no input is given.

return; → Stops function execution if input is empty.

updateMarks(Number(newMarks)); → Converts input to number and passes it as argument.

Number(newMarks) → Converts string input into numeric value.

</script> → Ends JavaScript section.

</body> → Ends body section.

</html> → Ends HTML document.