Javascript _Day - 6 Hands On _Sana Naveen

Problem 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.

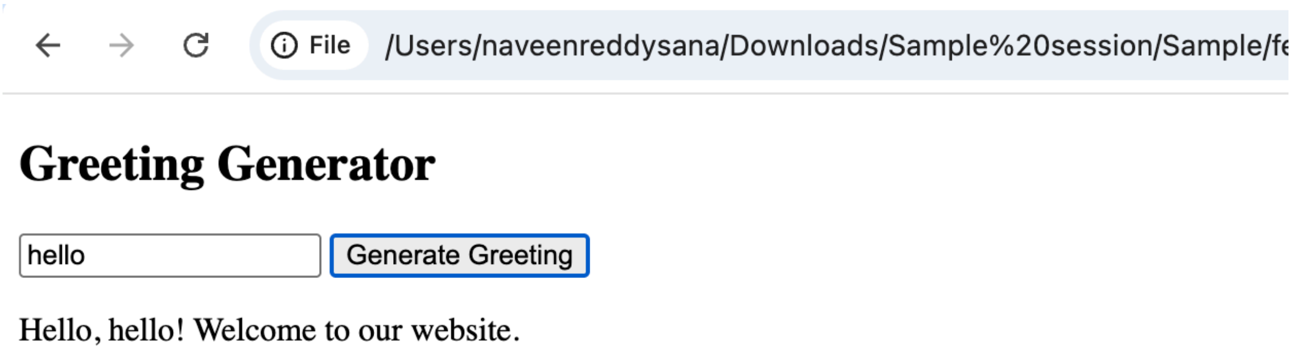
Technical Constraints

- 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.

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(document.getElementById('username').value)">
    Generate Greeting
  </button>

  <!-- Output -->
  <p id="greeting"></p>

  <script>

    function generateGreeting(name) {

      // Variable declared inside function
      var message = "Hello, " + name + "! Welcome to our website.";

      document.getElementById("greeting").innerHTML = message;
    }
    </script>

</body>
</html>
```

Output Screenshot:


File /Users/naveenreddysana/Downloads/Sample%20session/Sample/fe

Greeting Generator

hello

```
Hello, hello! Welcome to our website.
```

Explanation :

This webpage uses an input field to capture the user's name and a button with an inline onclick event to trigger a function. The generateGreeting() function accepts the name as a parameter and declares a local variable (message) inside the function scope. The greeting message is then dynamically inserted into a <p> element using document.getElementById() for DOM manipulation. The implementation uses only vanilla JavaScript and the function keyword as required.

Problem 2:

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

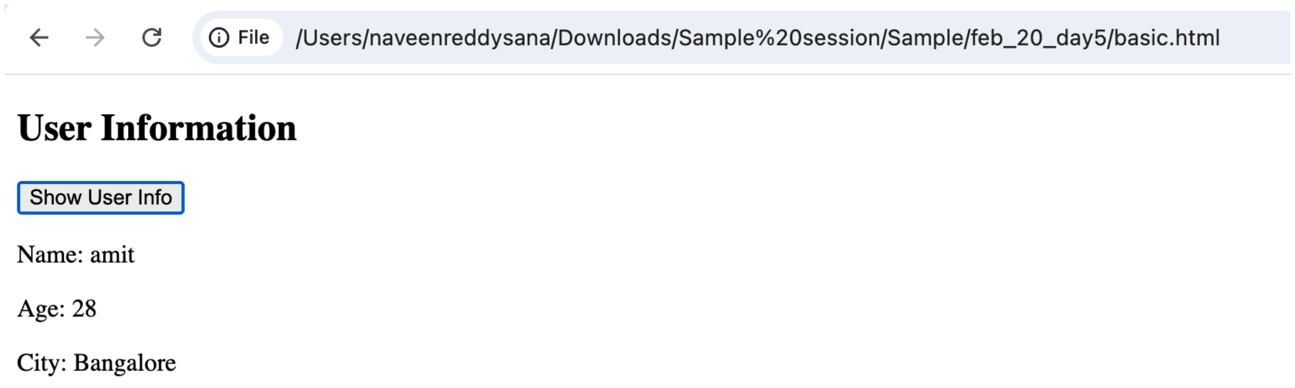
- Object properties must be accessed using dot notation.
- Function should not use global variables.
- DOM elements should be updated dynamically.

Code :

```
<!DOCTYPE html>
<html>
<head>
  <title>User Info Display</title>
</head>
<body>

  <h2>User Information</h2>

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

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

  <script>

    // JavaScript object
    var user = {
      name: "Amit",
      age: 28,
      city: "Bangalore"
    };

    function displayUserInfo(userObj) {

      document.getElementById("name").innerHTML =
        "Name: " + userObj.name;

      document.getElementById("age").innerHTML =
        "Age: " + userObj.age;

      document.getElementById("city").innerHTML =
        "City: " + userObj.city;
    }

  </script>
```

```
</body>
</html>
```

Output Screenshot :



/Users/naveenreddysana/Downloads/Sample%20session/Sample/feb_20_day5/basic.html

```
User Information

Show User Info

Name: amit

Age: 28

City: Bangalore
```

Explanation :

This webpage defines a JavaScript object user containing name, age, and city properties. A button triggers the displayUserInfo(userObj) function, which accepts the object as a parameter. Inside the function, object properties are accessed using dot notation (e.g., userObj.name) and dynamically displayed in separate <p> elements using document.getElementById(). The function operates only on the passed parameter and does not rely on global variables.

Problem 3:

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

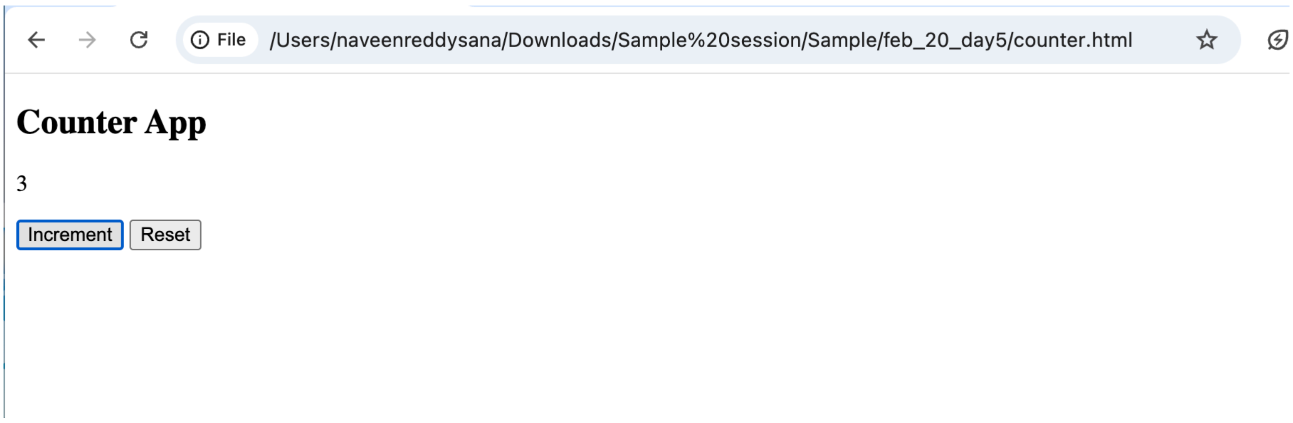
- Counter value must be maintained outside the function (global scope).
- DOM updates must happen inside functions only.
- No inline JavaScript in HTML.

Code :

```
<!DOCTYPE html>
<html>
<head>
   <title>Counter Application</title>
</head>
<body>

   <h2>Counter App</h2>

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

   <button id="incrementBtn">Increment</button>
   <button id="resetBtn">Reset</button>

   <script>

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

      // Function to increment counter
      function incrementCounter(step) {
         counter = counter + step;
         document.getElementById("counterValue").innerHTML = counter;
      }

      // Function to reset counter
      function resetCounter() {
         counter = 0;
         document.getElementById("counterValue").innerHTML = counter;
```

```
        }

        // Event listeners (no inline JS)
        document.getElementById("incrementBtn")
                .addEventListener("click", function() {
                    incrementCounter(1);
                });

        document.getElementById("resetBtn")
                .addEventListener("click", resetCounter);

    </script>

</body>
</html>
```

Output Screenshots :


File
/Users/naveenreddysana/Downloads/Sample%20session/Sample/feb_20_day5/counter.html

Counter App

3

Explanation :
This counter application uses a global variable (counter) to maintain the counter value outside the functions. The incrementCounter(step) function accepts a parameter (step) and updates the counter accordingly, while the resetCounter() function resets it to zero. DOM updates are performed inside these functions using document.getElementById(). Event handling is implemented using addEventListener() instead of inline JavaScript, ensuring separation of structure and behavior.

Problem 4:

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.

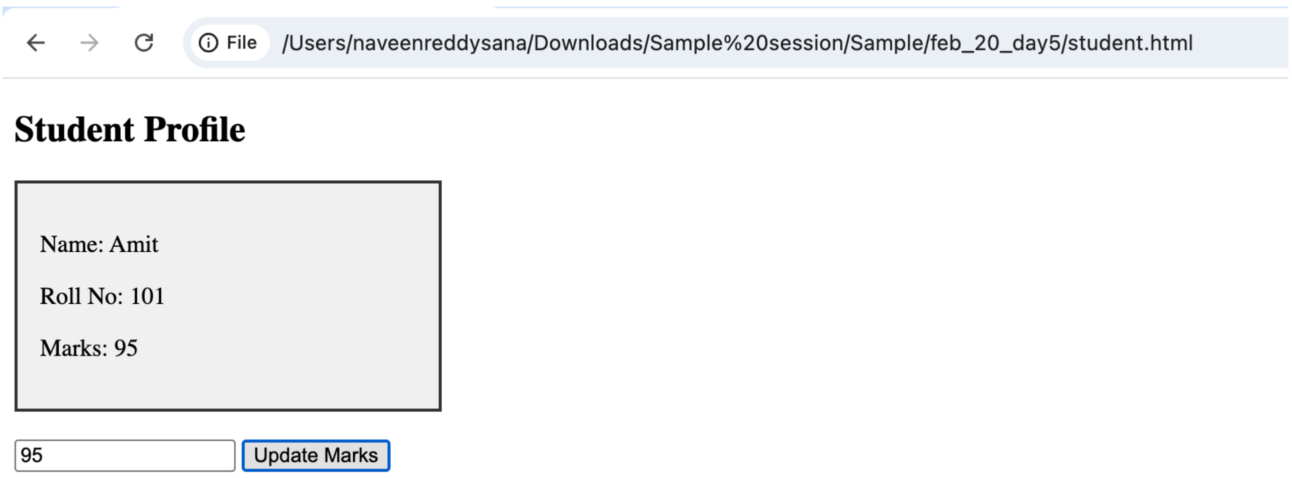
Technical Constraints

- Student object must be declared in global scope.
- Functions should update object values using parameters.
- DOM should update without page refresh.

Code :

```
<!DOCTYPE html>
<html>
<head>
    <title>Student Profile System</title>
    <style>
        #profileBox {
            border: 2px solid #333;
            padding: 15px;
            width: 250px;
            background-color: #f4f4f4;
            margin-top: 10px;
        }
    </style>
</head>
<body>

    <h2>Student Profile</h2>

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

    <br>
    <input type="number" id="newMarks" placeholder="Enter new marks">
    <button id="updateBtn">Update Marks</button>

    <script>

        // Global student object
        let student = {
            name: "Amit",
            rollNo: 101,
            marks: 95
        };

        // Function to display student profile
        function updateStudentProfile(studentObj) {

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

        // Function to update marks and refresh display
        function updateMarks(newMarks) {

            student.marks = newMarks;  // Update object using parameter
            updateStudentProfile(student);  // Refresh display
        }

        // Initial display
        updateStudentProfile(student);

        // Button event
        document.getElementById("updateBtn")
                .addEventListener("click", function() {

                    let marksValue = document.getElementById("newMarks").value;
                    updateMarks(marksValue);
                });

    </script>

</body>
</html>
```

Output Screenshot :



/Users/naveenreddysana/Downloads/Sample%20session/Sample/feb_20_day5/student.html

Student Profile

Name: Amit

Roll No: 101

Marks: 95

Explanation :
This application defines a global student object containing name, roll number, and marks. The updateStudentProfile(studentObj) function dynamically updates the DOM by displaying the object properties inside a styled <div>. The updateMarks(newMarks) function modifies the global object's marks property using the parameter and refreshes the display without reloading the page. DOM manipulation is handled dynamically using innerHTML.