HTML5_Day5_Hands_On_Koduru_Ushasree

Problem 1: Greeting Generator (Level-1)

Scenario

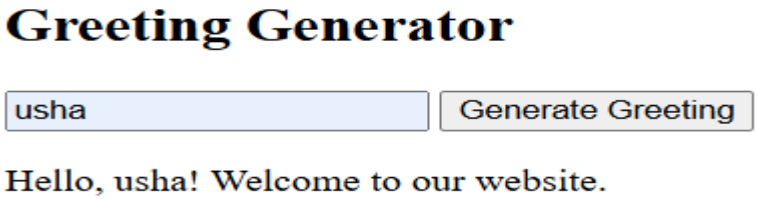
A simple webpage should greet users based on the name they enter. This helps beginners understand how JavaScript functions accept parameters and how variables behave inside functions.

📌 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."

Technical Requirements :-

This problem teaches how a function works with input from the user. The user enters their name in a text box, and when they click the button, a function runs. Inside the function, we take the input value, store it in a local variable, create a greeting message, and display it on the webpage using DOM manipulation. It helps you understand how functions receive data, how local variables work, and how JavaScript updates HTML elements dynamically.



Code :-

p1.html > html > body > p#greetingMessage

```html
<!DOCTYPE html>
<html>
<head>
    <title>Greeting Generator</title>
</head>
<body>

    <h2>Greeting Generator</h2>

    <!-- Input box to enter user's name -->
    <input type="text" id="username" placeholder="Enter your name">

    <!-- Button to generate greeting -->
    <button onclick="generateGreeting()">Generate Greeting</button>

    <!-- Paragraph where greeting will be displayed -->
    <p id="greetingMessage"></p>

<script>

    // This function runs when button is clicked
    function generateGreeting() {

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

        // Create a local variable for greeting message
        // (This variable exists only inside this function)
        var message = "Hello, " + name + "! Welcome to our website.";

        // Display greeting inside <p> element
        document.getElementById("greetingMessage").innerHTML = message;
    }

</script>

</body>
```

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

Scenario

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 Requirements :-

This problem focuses on JavaScript objects. You create a user object with properties like name, age, and city. A function accepts this object as a parameter and displays each property in separate <p> elements using dot notation (like user.name). When a button is clicked, the function runs and updates the webpage dynamically. This helps you understand how objects store related data and how objects can be passed to functions.

Output :-

Code :-

p2.html > html > body > script > user > age

```
2   <html>
6   <body>
9
10  <button onclick="showUser()">Display User Info</button>
11
12  <p id="name"></p>
13  <p id="age"></p>
14  <p id="city"></p>
15
16  <script>
17
18      // Create user object
19      var user = {
20          name: "Usha",
21          age: 21,
22          city: "Bangalore"
23      };
24
25      // Wrapper function for button click
26      function showUser() {
27          displayUserInfo(user);
28      }
29
30      // Function that accepts object as parameter
31      function displayUserInfo(userObj) {
32
33          // Access object properties using dot notation
34          document.getElementById("name").innerHTML = "Name: " + userObj.name;
35          document.getElementById("age").innerHTML = "Age: " + userObj.age;
36          document.getElementById("city").innerHTML = "City: " + userObj.city;
37      }
38
39  </script>
40
41  </body>
42  </html>
```

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

Scenario

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.

Code :-

p3.html > html > body > button#resetBtn

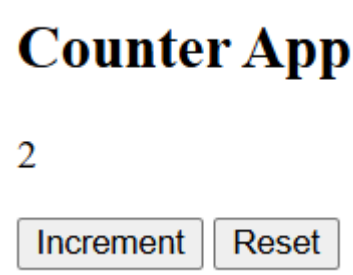
```
<!DOCTYPE html>
<html>
<head>
    <title>Counter App</title>
</head>
<body>

<h2>Counter App</h2>

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

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

<script>

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

    // Function to increment counter
    function incrementCounter(step) {

        // Update global variable
        counter = counter + step;

        // Update UI
        document.getElementById("counterDisplay").innerHTML = counter;
    }

```

```
    // Function to reset counter
    function resetCounter() {

        counter = 0;

        document.getElementById("counterDisplay").innerHTML = counter;
    }

    // Attach event listeners (NO inline JS)
    document.getElementById("incrementBtn").addEventListener("click", function() {
        incrementCounter(1);   // Step value passed
    });

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

/script>

/body>
/html>
```

Technical Requirements :-

This problem teaches the difference between global and local scope. A global variable stores the counter value so it can be accessed and modified by different functions. The incrementCounter(step) function increases the counter by the given step value, and the reset function sets it back to 0. The DOM is updated inside the functions so the user sees the changes instantly. This helps you understand real-world use of global variables and how UI updates happen dynamically.

Output :-



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

Scenario

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 .

Add another function updateMarks(newMarks):

- Updates marks and refreshes the display.

Code :-

p4.html > html > body > script > updateStudentProfile

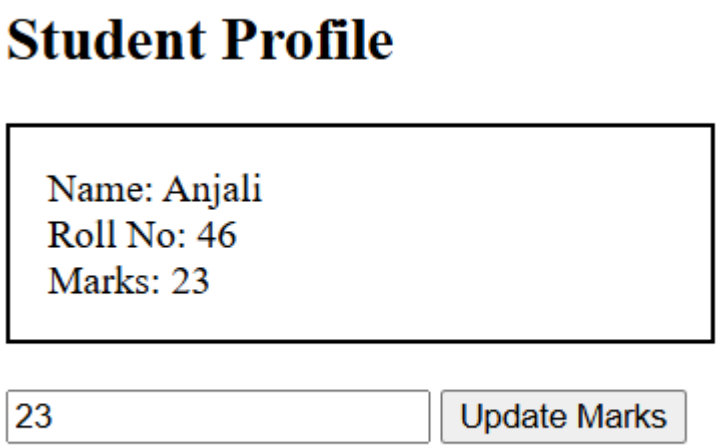
```
<!DOCTYPE html>
<html>
<head>
    <title>Student Profile Manager</title>
    <style>
        #profileBox {
            border: 2px solid black;
            padding: 15px;
            width: 250px;
        }
    </style>
</head>
<body>

<h2>Student Profile</h2>

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

<br>

<input type="number" id="newMarks" placeholder="Enter New Marks">
<button onclick="updateMarksFromInput()">Update Marks</button>
```

```
<script>
    // Function to display student profile
    function updateStudentProfile(studentObj) {
        var output =
            "Name: " + studentObj.name + "<br>" +
            "Roll No: " + studentObj.rollNo + "<br>" +
            "Marks: " + studentObj.marks;

        document.getElementById("profileBox").innerHTML = output;
    }
    // Function to update marks
    function updateMarks(newMarks) {
// Update object property
        student.marks = newMarks;
        // Refresh display
        updateStudentProfile(student);
    }
// Get input value and call updateMarks
    function updateMarksFromInput() {

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

        updateMarks(marksValue);
    }
// Initial display on page load
    updateStudentProfile(student);

</script>

</body>
</html>
```

Technical Requirements :-

This problem builds a mini system using objects and functions together. A global student object stores details like name, roll number, and marks. One function displays the student information in a styled , and another function updates the marks and refreshes the display. This teaches you how to modify object properties using functions and how shared data (global object) can be updated and reflected on the webpage without reloading.

Output :-


Student Profile
Name: Anjali
Roll No: 46
Marks: 23
23
Update Marks