# Form Validation – Case Study

## Project Overview:

In this project, you will add JavaScript to a form to ensure that users fill it out correctly. You’ll learn how to:

1. Validate that all required fields are filled.
2. Display error messages if something is wrong.
3. Clear errors when users begin typing again.

# Step 1: Understanding the Goal

Your form asks users for two pieces of information:

1. **Name**: The user’s name (required).
2. **Message**: A message they want to send (required).

We’ll add **JavaScript validation** to make sure both fields are filled before the form is submitted. If either field is left blank, the form will display an error message next to the empty field and stop the submission.

# Step 2: Setting Up JavaScript

Let’s start by linking your **JavaScript file** to the HTML file. You should already have the following at the end of your HTML file:

Now, create a file called **script.js** where we will write the validation logic.

# Step 3: Plan the Validation

We need to check two things:

* The **name** field isn’t empty.
* The **message** field isn’t empty.

If either field is empty, we will:

1. Show an error message in red text next to the field.
2. Stop the form from submitting.

We will also:

* Clear error messages when users start typing again.

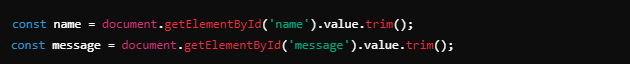
# Step 4: Writing the validateForm() Function

## What this function will do:

1. Get the values entered in the **name** and **message** fields.
2. If either field is empty, display an error message.
3. If both fields are filled, allow the form to submit.

## Step-by-Step Instructions:

1. **Get input values**:
   * Use JavaScript to retrieve the values entered in the form’s input fields.
   * You’ll need the document.getElementById() method to access the fields by their id.
   * Store the values in variables.



1. **Validate the fields**:
   * We’ll check if the fields are empty.
   * If a field is empty, display an error message and prevent the form from submitting by returning false.

A screen shot of a computer program

Description automatically generated

1. **Return a validation result**:
   * If both fields are filled, the function should return true to allow the form to submit.
   * Otherwise, it returns false to prevent the form from submitting.

# Step 5: Clearing Error Messages

When users start typing again, we should hide the error messages. Let’s create a function called clearErrors() that hides the error messages.

## What this function will do:

1. It will reset the error message areas by setting them to empty and hiding them.
2. We’ll call this function at the start of validateForm() to clear any previous errors.

A computer code on a black background

Description automatically generated

* This function will be called **before** the validation logic to make sure any old error messages are cleared when the form is validated again.

# Step 6: Connecting the JavaScript to the Form

To link the JavaScript to your form, you’ll modify the form’s onsubmit attribute in the HTML to run your validation function when the user tries to submit the form.

In your HTML, you should already have this:

This will trigger the validateForm() function whenever the user clicks the submit button. If validateForm() returns false, the form won’t submit.

# Final Code Walkthrough

Here’s what your script.js file should look like:

## Complete JavaScript Code:

// Validate the form when submitted

function validateForm() {

let isValid = true;

// Get input values

const name = document.getElementById('name').value.trim();

const message = document.getElementById('message').value.trim();

// Clear previous error messages

clearErrors();

// Validate name field

if (name === '') {

document.getElementById('nameError').innerText = "Name is required.";

document.getElementById('nameError').style.display = 'block';

isValid = false;

}

// Validate message field

if (message === '') {

document.getElementById('messageError').innerText = "Message is required.";

document.getElementById('messageError').style.display = 'block';

isValid = false;

}

// Return true if the form is valid, false otherwise

return isValid;

}

// Clear error messages when form is re-validated

function clearErrors() {

document.getElementById('nameError').style.display = 'none';

document.getElementById('messageError').style.display = 'none';

}

# Testing the Form

1. Test 1: Try submitting the form with empty fields.
   * You should see an error message next to the fields that are missing input.
2. Test 2: Type something in the **name** field, but leave the **message** field empty.
   * Only the message field should display an error message.
3. Test 3: Fill in both fields and submit the form.
   * The form should now submit without showing any errors.

# Key Takeaways

* **Form validation** is crucial to ensure that users fill out the form correctly before submitting it.
* You can use **JavaScript** to dynamically check the form’s inputs and display helpful error messages.
* **Clearing error messages** before each validation ensures that users aren’t confused by old error messages.

# Next Steps

If you finish early and want to explore more:

* Add another input field (like email) and create a similar validation for it.
* Try adding a “clear form” button that resets all the fields to empty.