MIT WORLD PEACE UNIVERSITY

Full Stack Development
Third Year B.Tech, Semester 1

Form Validation Using JavaScript

ASSIGNMENT 2

Prepared By A1−02. Mayur Behere
Batch A

Aim:

Write a program to perform following form validations using JavaScript.

Objectives:

1. To understand what is form validation.
2. To learn basic functioning of DOM objects.
3. To learn how to apply various techniques to implement it.

Theory:

1] Different types of form validations.
=>
Form validations are crucial in web development to ensure that user-submitted data is accurate, properly formatted, and meets specific criteria. Here are different types of form validations commonly used:
1) Required Field Validation: Checks if a particular field is filled out before submitting the form. It ensures that essential fields are not left blank.
2) Length Validation: Verifies if the input length meets specific criteria, such as minimum and maximum character limits. For example, a password field may require a minimum length for security purposes.
3) Numeric Validation: Ensures that the input contains only numeric characters. It can be used for fields that expect numbers, such as age or phone number.
4) Email Validation: Validates input to match the email format. It checks for the presence of '@' and '.' characters in the appropriate positions.
5) Pattern Matching Validation: Uses regular expressions to define specific patterns that the input must match. It allows validation for custom formats, such as postal codes, phone numbers, or special character requirements.
6) Date Validation: Verifies if the input date is in a specific format (e.g., MM/DD/YYYY or YYYY-MM-DD) and falls within a valid range.
7) Checkbox and Radio Button Validation: Ensures that at least one checkbox is checked or one radio button is selected from a group of options.

8) Confirmation Validation: Compares two input fields, like password and confirm password, to ensure they match before form submission.
9) Custom Validation Messages: Providing descriptive error messages to users when their input does not meet validation requirements. This assists users in understanding what corrections are needed.
10) Real-time Validation: Conducts validation while the user is filling out the form, providing immediate feedback on errors without waiting for submission.

2]HTML Document Object Model
=>
The HTML Document Object Model (DOM) is a programming interface for web documents. It represents the structure of an HTML document as a tree-like structure, where each node in the tree corresponds to a part of the document, such as elements (tags), attributes, and text. The DOM provides a way for programs or scripts to dynamically access and manipulate the content, structure, and style of a web page.
Key aspects of the HTML DOM:
1)Tree Structure: The DOM represents the HTML document as a hierarchical tree structure called nodes. Each HTML element, attribute, text, comment, etc., is a node in this tree.
2)Nodes: Nodes are the fundamental building blocks of the DOM. There are various types of nodes:

- Element Nodes: Represent HTML elements, like <div>, <p>, <span>, etc.
- Attribute Nodes: Represent attributes of elements.
- Text Nodes: Contain the text within elements.
- Comment Nodes: Represent comments within HTML.

3) Accessing and Manipulating Elements: Through scripting languages like JavaScript, developers can access, modify, add, or delete elements and their attributes using DOM methods and properties.
4) Traversal and Manipulation: Developers can traverse the DOM tree by moving from one node to another (parent, child, or sibling nodes) and manipulate elements or their attributes accordingly.
5) Event Handling: The DOM allows the attachment of event handlers to elements, enabling the execution of specific actions or functions in response to user interactions (like clicks, mouse movements, keyboard inputs, etc.).

6) Dynamic Content and Styling: Using the DOM, developers can dynamically create, change, or remove elements, as well as modify their styles and classes in real-time based on user interactions or other events.
7) Cross-Browser Consistency: The DOM provides a standardized way to interact with web documents, ensuring consistency in accessing and manipulating document content across different browsers.

3] What is JQuery? Write various JQuery Selectors
=>
jQuery is a fast, lightweight, and feature-rich JavaScript library that simplifies HTML document traversal and manipulation, event handling, animation, and AJAX interactions. It provides an easy-to-use API that abstracts away the complexities of JavaScript, making it easier for developers to write concise and efficient code.
Various jQuery Selectors:

- Element Selector: Selects all elements with a specified element name.

Example: $('p') selects all <p> elements.

- ID Selector: Selects a single element with a specific ID attribute.

Example: $('#myElement') selects the element with id="myElement".

- Class Selector: Selects all elements with a specific class name.

Example: $('.myClass') selects all elements with class="myClass".

- Attribute Selector: Selects elements based on specific attribute values.

Example: $('[data-type="example"]') selects elements with data-type="example" attribute.

- Multiple Selectors: Selects multiple elements using multiple selectors separated by commas.

Example: $('h1, h2, h3') selects all <h1>, <h2>, and <h3> elements.

- Descendant Selector: Selects all descendants of a specified ancestor element.

Example: $('div p') selects all <p> elements that are descendants of <div> elements.

- Child Selector: Selects direct children of a specified parent element.

Example: $('ul > li') selects all <li> elements that are direct children of <ul>.

- Sibling Selector: Selects elements that are siblings of another element.

Example: $('h2 + p') selects all <p> elements that are immediately preceded by <h2> elements.

- Contains Selector: Selects elements that contain specific text.

Example: $('p:contains("example")') selects all <p> elements containing the text "example".

- Not Selector: Excludes elements that match a specific selector.

Example: $('div:not(.special)') selects all <div> elements that do not have the class special.

FAQ :

1] Write 3 reasons why Form validations are important.
=>
Reasons why Form Validations are Important:

- Data Accuracy and Integrity: Form validations ensure that the data submitted by users meets specific criteria (such as correct format, length, or type). This helps maintain the accuracy and integrity of the data stored in the system/database.
- Enhanced User Experience: Validations provide immediate feedback to users, preventing them from submitting incorrect or incomplete information. This improves the overall user experience by guiding users to fill out forms correctly without frustration.
- Security Measures: Validations can also serve as a security measure against potential vulnerabilities, such as preventing malicious input (e.g., SQL injection, cross-site scripting) by validating and sanitizing user-submitted data before processing it.

2] Give an example of how to modify an attribute value using DOM.
=>
Example of Modifying an Attribute Value using DOM:
Let's say we have an HTML element with an id attribute and we want to change its src attribute using JavaScript:
HTML:

```
<img id="myImage" src="old_image.jpg" alt="Old Image">
```

JavaScript to modify the src attribute using DOM:

```
// Access the element by its ID
const imageElement = document.getElementById('myImage');
// Modify the src attribute
imageElement.src = 'new_image.jpg';
```

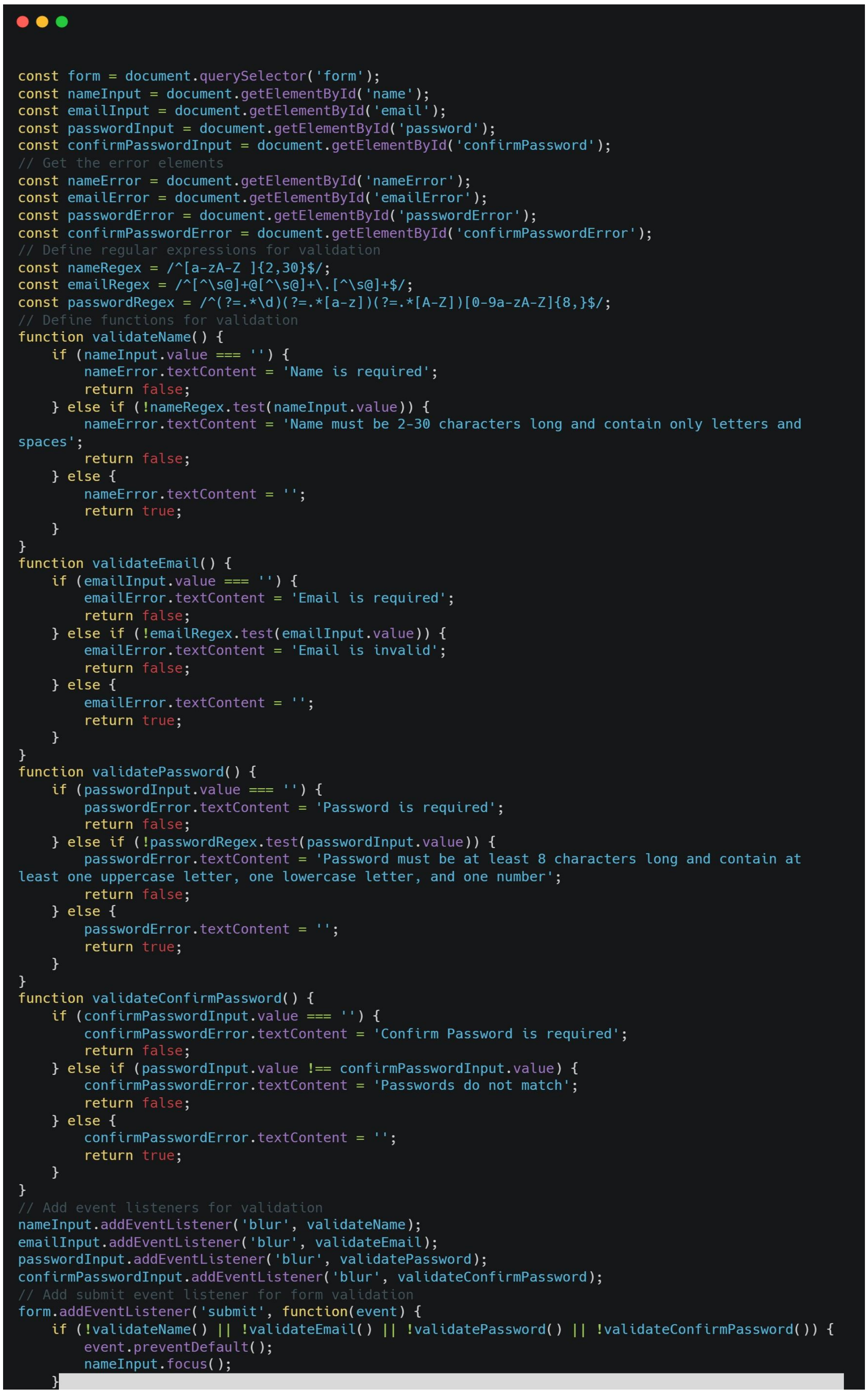
This JavaScript code accesses the <img> element by its ID (myImage) using getElementById and then changes its src attribute to 'new_image.jpg'.

3] What is jQuery Ajax?
=>
jQuery Ajax refers to the usage of the jQuery library's Ajax-related methods to perform asynchronous HTTP requests to the server. AJAX (Asynchronous JavaScript and XML) allows web pages to send and receive data from a server asynchronously without reloading the entire page.
jQuery simplifies AJAX requests and responses by providing methods like $.ajax(), $.get(), $.post(), etc., which handle the communication with the server and manage the callbacks for success, failure, or completion of the request.
For example, using jQuery's $.ajax() method to make an AJAX GET request:
Code:

```
$.ajax({
 url: 'https://api.example.com/data',
 method: 'GET',
success: function(response) {
          // Handle successful response
          console.log('Data received:', response);
 },
error: function(xhr, status, error) {
// Handle error
console.error('Error:', error);
 }
 });
```

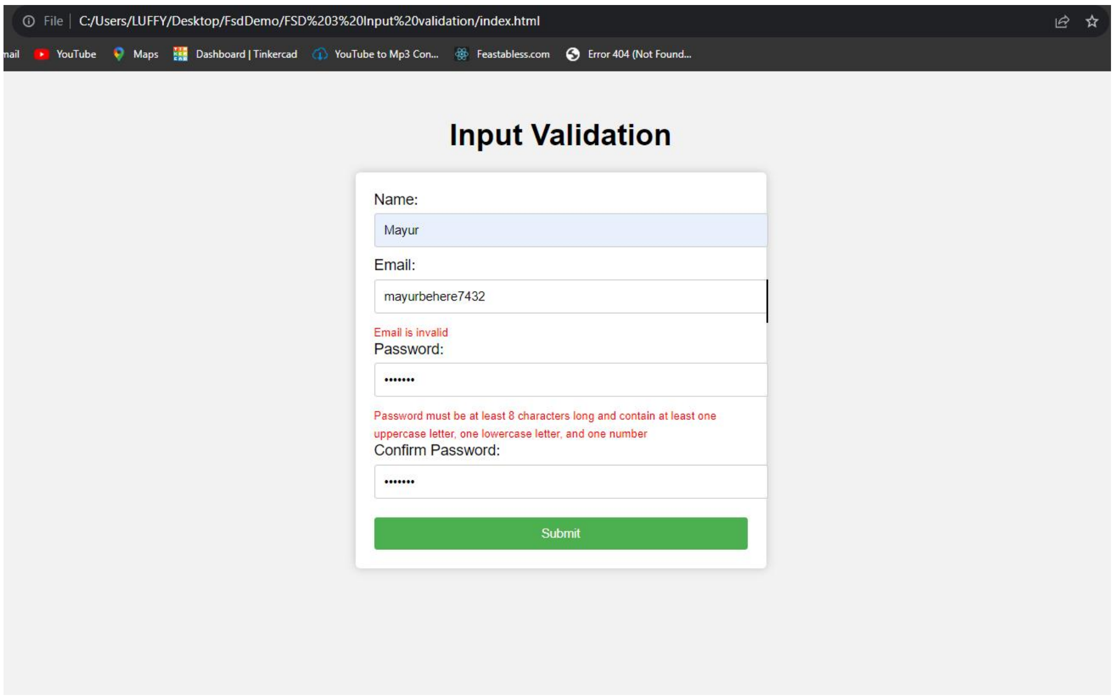
jQuery's AJAX capabilities enable developers to create more dynamic and interactive web applications by fetching data from servers, submitting form data, and updating parts of web pages without requiring a full page reload.

Code:



```
const form = document.querySelector('form');
const nameInput = document.getElementById('name');
const emailInput = document.getElementById('email');
const passwordInput = document.getElementById('password');
const confirmPasswordInput = document.getElementById('confirmPassword');
// Get the error elements
const nameError = document.getElementById('nameError');
const emailError = document.getElementById('emailError');
const passwordError = document.getElementById('passwordError');
const confirmPasswordError = document.getElementById('confirmPasswordError');
// Define regular expressions for validation
const nameRegex = /^[a-zA-Z ]{2,30}$/;
const emailRegex = /^[^\s@]+@[^\s@]+\.[^\s@]+$/;
const passwordRegex = /^(?=.*\d)(?=.*[a-z])(?=.*[A-Z])[0-9a-zA-Z]{8,}$/;
// Define functions for validation
function validateName() {
    if (nameInput.value === '') {
        nameError.textContent = 'Name is required';
        return false;
    } else if (!nameRegex.test(nameInput.value)) {
        nameError.textContent = 'Name must be 2-30 characters long and contain only letters and
spaces';
        return false;
    } else {
        nameError.textContent = '';
        return true;
    }
}
function validateEmail() {
    if (emailInput.value === '') {
        emailError.textContent = 'Email is required';
        return false;
    } else if (!emailRegex.test(emailInput.value)) {
        emailError.textContent = 'Email is invalid';
        return false;
    } else {
        emailError.textContent = '';
        return true;
    }
}
function validatePassword() {
    if (passwordInput.value === '') {
        passwordError.textContent = 'Password is required';
        return false;
    } else if (!passwordRegex.test(passwordInput.value)) {
        passwordError.textContent = 'Password must be at least 8 characters long and contain at
least one uppercase letter, one lowercase letter, and one number';
        return false;
    } else {
        passwordError.textContent = '';
        return true;
    }
}
function validateConfirmPassword() {
    if (confirmPasswordInput.value === '') {
        confirmPasswordError.textContent = 'Confirm Password is required';
        return false;
    } else if (passwordInput.value !== confirmPasswordInput.value) {
        confirmPasswordError.textContent = 'Passwords do not match';
        return false;
    } else {
        confirmPasswordError.textContent = '';
        return true;
    }
}
// Add event listeners for validation
nameInput.addEventListener('blur', validateName);
emailInput.addEventListener('blur', validateEmail);
passwordInput.addEventListener('blur', validatePassword);
confirmPasswordInput.addEventListener('blur', validateConfirmPassword);
// Add submit event listener for form validation
form.addEventListener('submit', function(event) {
    if (!validateName() || !validateEmail() || !validatePassword() || !validateConfirmPassword()) {
        event.preventDefault();
        nameInput.focus();
    }
```

Output :


File | C:/Users/LUFFY/Desktop/FsdDemo/FSD%203%20Input%20validation/index.html

Input Validation

Name:
Mayur
Email:
mayurbehere7432
Email is invalid
Password:
•••••••
Password must be at least 8 characters long and contain at least one uppercase letter, one lowercase letter, and one number
Confirm Password:
•••••••

LINKs :

Github repo = https://github.com/MayurBehere/FsdSem5/tree/main/FSD%203%20Input%20validation
Project View = file:///C:/Users/LUFFY/Desktop/FSD/FsdSem5/FSD%203%20Input%20validation/index.html

Result :

Input Validation using Javascript done.