# FSD ASSIGNMENT - 3

By-Yashvardhan Tekavade
PB-15 Batch-1
Panel-1 TY-CSF

---

**Client-side Form Validations using JavaScript, DOM real-time update, and JQuery to develop Ajax-based applications.**

## Aim:

Write a program to perform the following form validations using JavaScript:

- All fields mandatory,
- Phone number, Email Address, Zip code Validation etc.

Include JavaScript to access and manipulate Document Object Model (DOM) objects in an HTML web page.

Include JQuery to develop to develop your application as an Ajax-based application.

## Objectives:

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

## Theory

### Different types of form validations.

Form validation is a critical part of web development that helps ensure that the data submitted by users through forms is accurate, complete, and meets the required criteria. There are several types of form validations, each serving a specific purpose:

- **Client-Side Validation:**
  - **Required Fields:** Ensure that users fill in all the mandatory fields.
  - **Data Types:** Validate that the data entered matches the expected data type (e.g., numbers, email addresses, dates).
  - **Length Limits:** Check that the input doesn't exceed a specified length.
  - **Pattern Matching:** Verify that data follows a specific format (e.g., phone numbers, postal codes).
  - **Password Strength:** Ensure that passwords meet complexity criteria (e.g., minimum length, special characters).

- **Confirmation:** Confirm that certain inputs (e.g., password confirmation) match the original input.
- **Checkbox/Radio Button Selection:** Ensure users select at least one option.
- **File Upload:** Validate file types, sizes, and required attachments.

- **Server-Side Validation:**
  - **Data Integrity:** Confirm that data submitted to the server is consistent with business rules and constraints.
  - **Database Validation:** Ensure that data entered conforms to the database schema and doesn't violate constraints.
  - **Unique Values:** Check for uniqueness of values (e.g., unique usernames, email addresses).
  - **Security:** Validate inputs to prevent SQL injection, XSS (Cross-Site Scripting), and other security vulnerabilities.
  - **Authentication:** Verify user credentials during login and session management.

- **Real-Time Validation:**
  - **As-You-Type Validation:** Provide immediate feedback to users as they type (e.g., checking username availability).
  - **Auto-Suggest:** Offer suggestions or corrections based on user input.

- **Conditional Validation:**
  - **Dependency Validation:** Validate fields conditionally based on the value of other fields (e.g., state field depends on country selection).
  - **Dynamic Validation:** Change validation rules or messages dynamically based on user actions.

- **Date and Time Validation:**
  - **Date Range:** Ensure dates are within a specified range.
  - **Future/Past Dates:** Validate that dates are in the future or the past as required.
  - **Date Calculations:** Verify dates based on calculations (e.g., end date is after the start date).

- **Custom Validation Rules:**

Implement custom validation logic to enforce specific business rules or requirements unique to your application.

- **Captcha and Bot Detection:**

  Employ CAPTCHA or reCAPTCHA to verify that the form is being submitted by a human and not a bot.

- **Error Handling and Feedback:**

  Provide clear and user-friendly error messages to help users understand and correct their input errors.

- **Localization and Internationalization:**

  Tailor validation messages to the user's preferred language or locale.

- **Accessibility Validation:**

  Ensure that your forms are accessible to users with disabilities, conforming to WCAG (Web Content Accessibility Guidelines) standards

**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, with each element in the HTML document as a node in the tree. This hierarchical structure allows developers to access, manipulate, and modify the content and structure of a web page using programming languages like JavaScript.
Key points about the HTML DOM:

- **Tree Structure:** The HTML DOM represents an HTML document as a tree structure, with the <html> element at the root. Each HTML element (e.g., <head>, <body>, <div>, <p>, etc.) is represented as a node in the tree, and attributes and text content are represented as properties of these nodes.
- **Nodes:** There are various types of nodes in the HTML DOM, including:
  - **Element nodes:** Represent HTML elements.
  - **Attribute nodes:** Represent attributes of elements.
  - **Text nodes:** Contain the text content of elements.
  - **Comment nodes:** Represent comments in the HTML source code.
- **Hierarchy:** The DOM tree structure reflects the nesting of elements in the HTML document. Child nodes are contained within their parent nodes, creating a hierarchical

relationship. For example, a <p> element within a <div> element would be a child node of the <div> element.

- **Access and Manipulation:** Developers can access and manipulate the HTML DOM using JavaScript or other scripting languages. This allows for dynamic updates to web pages, such as changing content, modifying attributes, adding or removing elements, and responding to user interactions.
- **Events:** The HTML DOM enables the handling of events (e.g., user clicks, keyboard input) by attaching event listeners to elements. These event listeners can trigger specific actions or functions when events occur, enhancing interactivity.
- **Traversal:** Developers can navigate the DOM tree by moving between nodes. Common traversal methods include accessing parent nodes, child nodes, sibling nodes, and descendant nodes.

**What is JQuery? Write various JQuery Selectors.**

jQuery is a popular JavaScript library designed to simplify the process of interacting with HTML documents, manipulating the DOM (Document Object Model), and handling events. It provides a set of functions and methods that make it easier for developers to perform common tasks such as DOM traversal, event handling, animations, and AJAX (Asynchronous JavaScript and XML) requests.
One of the key features of jQuery is its support for powerful and flexible selectors. jQuery selectors allow you to easily target and manipulate HTML elements in a document. Here are some common jQuery selectors:

- **Element Selector:** Selects all elements with a specified element name.
  - Example: $("p") selects all <p> elements on the page.
- **ID Selector:** Selects a single element with a specified ID attribute.
  - Example: $("#myElement") selects the element with id="myElement".
- **Class Selector:** Selects all elements with a specified class name.
  - Example: $(".myClass") selects all elements with class="myClass".
- **Attribute Selector:** Selects elements based on their attributes.
  - Example: $("[data-type='example']") selects elements with data-type="example".
- **Multiple Selector:** Combines multiple selectors into a single selection.
  - Example: $("p, .myClass, #myElement") selects paragraphs, elements with class="myClass", and the element with id="myElement".
- **Descendant Selector:** Selects elements that are descendants of another element.
  - Example: $("ul li") selects all <li> elements that are descendants of <ul> elements.
- **Child Selector:** Selects immediate child elements.
  - Example: $("ul > li") selects all <li> elements that are immediate children of <ul> elements.

- **First Selector:** Selects the first element that matches the selector.
  - Example: $("p:first") selects the first <p> element on the page.
- **Last Selector:** Selects the last element that matches the selector.
  - Example: $("p:last") selects the last <p> element on the page.
- **Even Selector:** Selects even-indexed elements in a group.
  - Example: $("li:even") selects all even-indexed <li> elements.
- **Odd Selector:** Selects odd-indexed elements in a group.
  - Example: $("li:odd") selects all odd-indexed <li> elements.
- **Not Selector:** Selects elements that do not match a specified selector.
  - Example: $("p:not(.special)") selects all <p> elements that do not have class="special".
- **Contains Selector:** Selects elements that contain specific text.
  - Example: $("p:contains('Lorem ipsum')") selects all <p> elements that contain the text "Lorem ipsum."

These are just some of the common jQuery selectors. jQuery's selector engine allows for a wide range of complex and flexible selections, making it a powerful tool for manipulating and interacting with web page elements.

**FAQs**

1. **Write 3 reasons why Form validations are important.**

Form validations are crucial for web applications and websites for several reasons:

- Data Accuracy and Integrity: Form validations ensure that the data submitted by users is accurate and meets the required criteria. With proper validation, users can enter correct or complete information, leading to data errors and inconsistencies in your database. Validating user input helps maintain data accuracy and integrity, which is critical for the proper functioning of applications and the reliability of the information collected.
- Enhanced User Experience: Proper form validation contributes to a positive user experience. When users encounter validation errors, clear and descriptive error messages guide them to correct their input, preventing frustration and confusion. This reduces the likelihood of users abandoning the form or website due to frustration, leading to higher user satisfaction and engagement.
- Security and Protection Against Attacks: Form validation plays a crucial role in web security. Without validation, malicious users can exploit vulnerabilities in your application by injecting harmful data or executing attacks like SQL injection or Cross-Site Scripting (XSS). Validating user input helps prevent these security risks by rejecting or sanitizing potentially dangerous data, ensuring the safety of your application and its users' data.

**2. Give an example of how to modify an attribute value using DOM.**

You can modify an attribute value of an HTML element using the DOM (Document Object Model) and JavaScript. Here's an example of how to change the src attribute of an <img> element:
In this example:

- We have an <img> element with the ID "myImage" and an original src attribute pointing to "original.jpg."
- Inside the <script> element, we use document.getElementById("myImage") to obtain a reference to the <img> element by its ID.
- We then modify the src attribute of the image element using imageElement.src = "new-image.jpg";, changing the image source to "new-image.jpg."
- Additionally, we change the alt attribute using imageElement.alt = "New Image Description"; to update the image description.

```
<!DOCTYPE html>

<html>

<head>

    <title>Modify Attribute Example</title>

</head>

<body>

    <img id="myImage" src="original.jpg" alt="Original Image">



    <script>

        // Get a reference to the image element by its ID

        var imageElement = document.getElementById("myImage");



        // Modify the 'src' attribute to change the image source

        imageElement.src = "new-image.jpg";
        // You can also modify other attributes, such as 'alt'

        imageElement.alt = "New Image Description";

    </script>

</body>

</html>
```

After running this code, the image on the web page will display the new image, and the image's description will also be updated. This demonstrates how you can use the DOM to modify attribute values of HTML elements dynamically using JavaScript.

**3. What is jQuery Ajax?**

jQuery Ajax (Asynchronous JavaScript and XML) is a set of methods and functionalities provided by the jQuery library to simplify the process of making asynchronous HTTP requests to a server. It allows you to exchange data with a server or retrieve data from a server without having to refresh the entire web page. While the name suggests XML, jQuery Ajax is not limited to XML data; it can handle various data formats, including JSON, HTML, text, and more.
Here are some key features and components of jQuery Ajax:

- **$.ajax() Method:** The $.ajax() method is the core of jQuery Ajax. It is a versatile method that allows you to customize the details of an Ajax request, including the HTTP method (GET, POST, PUT, DELETE, etc.), URL, data to send, headers, and more.
- **Shorthand Methods:** jQuery provides shorthand methods for common types of Ajax requests, such as $.get(), $.post(), $.getJSON(), $.load(), and $.ajaxSetup(). These methods simplify making common requests and handling responses.
- **Promises:** jQuery Ajax methods return promises (deferred objects), making it easier to work with asynchronous operations. You can use .done(), .fail(), and .always() to handle success, error, and completion events.
- **Data Formats:** jQuery Ajax can handle various data formats, including JSON, XML, HTML, and plain text. You can specify the expected data type using the data type option.
- **Cross-Origin Requests:** jQuery Ajax supports making cross-origin requests, either by enabling Cross-Origin Resource Sharing (CORS) on the server or by using JSONP (JSON with Padding) for cross-domain requests.

**Problem Statement:**

Write a program to design Student registration form by using HTML, CSS having following fields: Username, Email, Phone number, Password, Confirm Password and write external javascript code to achieve following validations

- Fields should not be empty. If spaces are entered those should be considered empty
- Phone number must accept only numeric values and it should be 10 digits
- Password length must be at least 7 and it should contain at least one capital letter, one digit and one special character from the set (&,$,#@)
- Value entered in password field and confirm password fields must match

Email address must contain @ sign and a ., there should be few letters before the @ sign, there should be three letters between @ sign and a . There must be 3 or 2 letters after the . (hint: Use regular expression)

Write a client-side script with JavaScript to access and manipulate Document Object Model (DOM) objects in an HTML web page. Develop a dynamic web page using javascript and DOM. Make use of the following for accessing elements

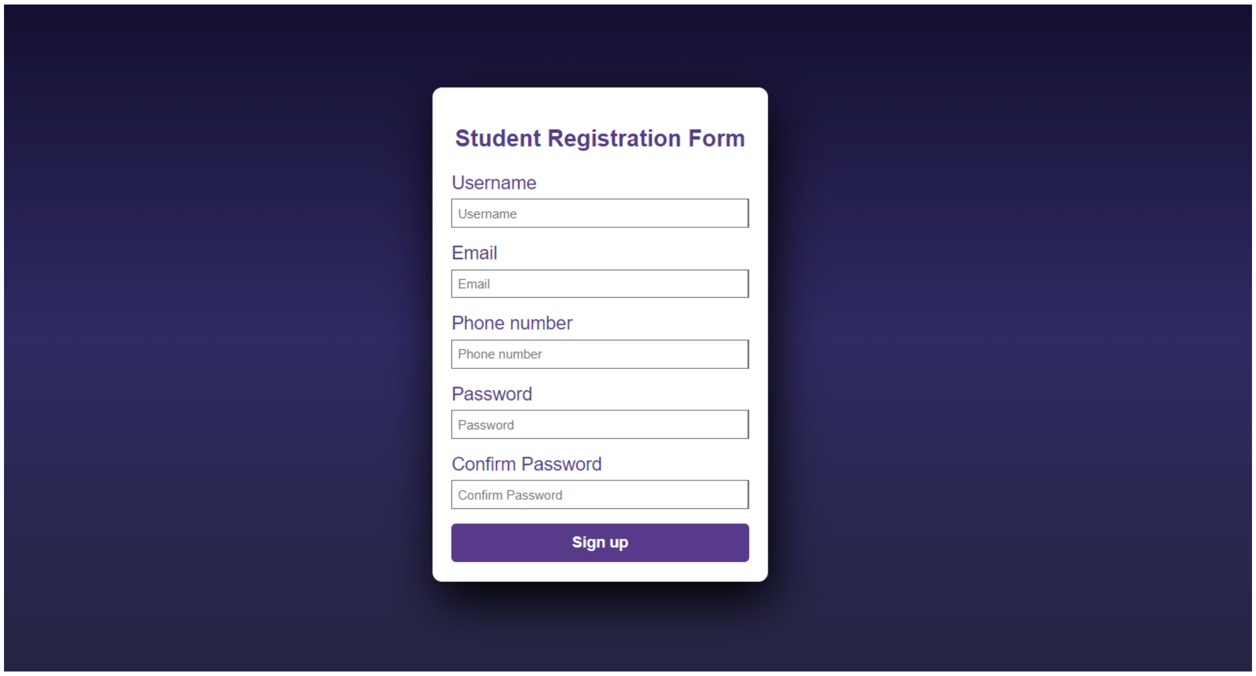
- getElementById, getElementsByTagName,getElementsByClassName
- Change the text using innerHTML property
- Change the CSS properties like color, position of a particular element on the page
- Change the image source after clicking on a button
- Add a text node and attach it to a parent node
- Delete a node

Include jQuery to perform following operations:

- Change button text using jQuery.
- Set background-image using jQuery CSS property.
- Access HTML form data using jQuery.
- Add attribute using jQuery

Use this reference link for jQuery :
https://www.w3resource.com/jquery-exercises/part1/index.php

**Output: Screenshots of the output to be attached.**


Student Registration Form

Username: Username

Email: Email

Phone number: Phone number

Password: Password

Confirm Password: Confirm Password

GITHUB LINK:- [https://github.com/yashtekavade/fsd/tree/main/3](https://github.com/yashtekavade/fsd/tree/main/3)