**Web Technologies Laboratory 03**

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

**Aim:** Write a program to perform following form validations using JavaScript: a) All fields mandatory,

b) 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 basic functioning of DOM objects.
3. To learn how to apply various techniques to implement it.

**Theory:**

**1.Different types of form validations.**

Form validation is essential for ensuring that the data entered by users is correct and meets the required criteria before submission. There are several types of form validations, each serving different purposes. Here's an overview of the most common types:

# 1. Required Field Validation

Ensures that a user does not leave a required field empty. This is often implemented using the required attribute in HTML. html

<input type="text" name="username" required>

# 2. Format Validation

Checks if the input matches a specific format, such as email addresses, phone numbers, or credit card numbers. This can be achieved using HTML5 input types and pattern attributes.

* **Email**: Validates that the input is in the correct email format.

html

<input type="email" name="email" required>

* **Pattern**: Uses regular expressions to validate custom formats.

html

<input type="text" name="phone" pattern="[0-9]{3}-[0-9]{3}-[0-9]{4}" required>

# 3. Length Validation

Ensures that the input meets specific length requirements, such as minimum and maximum number of characters.

html

<input type="text" name="username" minlength="3" maxlength="15" required>

# 4. Number Validation

Validates numerical inputs to ensure they fall within a specific range or meet certain criteria.

This includes checking for integer values, decimals, and range constraints.

* **Number Input**: Ensures the value is a number. html

<input type="number" name="age" min="1" max="120" required>

* **Range Input**: Allows a value within a specific range. html

<input type="range" name="volume" min="0" max="100" required>

# 5. Date and Time Validation

Validates date and time inputs to ensure they fall within acceptable ranges or formats.

* **Date**: Validates date formats.

html

<input type="date" name="birthdate" required>

* **Time**: Validates time formats. html

<input type="time" name="appointment" required>

# 6. Confirmation Validation

Ensures that two fields match, such as confirming a password or email address. This typically involves custom validation logic. html

<input type="password" name="password" id="password" required>

<input type="password" name="confirmPassword" id="confirmPassword" required>

<script>

document.getElementById('confirmPassword').addEventListener('input', function() { var password = document.getElementById('password').value;

var confirmPassword = this.value; if (password !== confirmPassword) { this.setCustomValidity('Passwords do not match');

} else {

this.setCustomValidity('');

}

});

</script>

# 7. Custom Validation

Allows for more complex validation scenarios that are not covered by standard HTML attributes. This involves writing JavaScript functions to validate the form data according to custom rules. html

<input type="text" name="username" id="username" required>

<script>

document.getElementById('username').addEventListener('input', function() { var username = this.value; if (username.length < 3) {

this.setCustomValidity('Username must be at least 3 characters long');

} else {

this.setCustomValidity('');

}

});

</script>

# 8. Asynchronous Validation

Checks the input against a server or database asynchronously to ensure it meets certain criteria, such as checking if a username is already taken. html

<input type="text" name="username" id="username" required>

<script>

document.getElementById('username').addEventListener('blur', function() { var username = this.value;

fetch('/check-username?username=' + username)

.then(response => response.json())

.then(data => { if (data.taken) {

this.setCustomValidity('Username is already taken');

} else {

this.setCustomValidity('');

}

});

});

</script>

# 9. Client-Side vs. Server-Side Validation

* **Client-Side Validation**: Performed in the browser before the form is submitted. It provides immediate feedback to users and improves user experience. It includes HTML5 validation, JavaScript validation, and libraries like jQuery Validation.
* **Server-Side Validation**: Performed on the server after form submission. It ensures that the data is correct and secure, even if client-side validation is bypassed. It is crucial for security and data integrity.

By using these types of validations, you can ensure that the data collected through your forms is accurate, complete, and in the correct format, leading to better user experience and data integrity.

**2.HTML Document Object Model.**

The HTML DOM is a programming interface for web documents. It represents the structure of a document as a tree of objects, where each object corresponds to a part of the document, such as elements, attributes, and text. The DOM allows programs and scripts to access, modify, and manipulate the content and structure of a web page dynamically.

**Key Concepts**

# 1. Document Object Model (DOM) Tree

The DOM represents an HTML document as a tree structure where:

* **Nodes**: Each part of the document (elements, attributes, text) is a node in the tree. o **Root Node**: The root of the tree is the document object.
* **Element Nodes**: Represent HTML elements (e.g., <div>, <p>, <a>).
* **Text Nodes**: Represent the text content within elements.
* **Attribute Nodes**: Represent attributes of elements (e.g., class, id).

For example, in the following HTML snippet:

html

<div id="container">

<h1>Title</h1>

<p>Paragraph text.</p>

</div>

The DOM tree structure would be: o document

* + html
  + head
  + body
  + div (id="container")
  + h1 (Title)
  + p (Paragraph text.)

▪

# 2. Accessing DOM Elements

You can access and manipulate DOM elements using JavaScript. Common methods and properties include: o **document.getElementById(id)**: Returns the element with the specified id.

javascript

var element = document.getElementById('container'); o **document.getElementsByClassName(className)**: Returns a collection of elements with the specified class name.

javascript

var elements = document.getElementsByClassName('my-class'); o **document.getElementsByTagName(tagName)**: Returns a collection of elements with the specified tag name.

javascript

var paragraphs = document.getElementsByTagName('p'); o **document.querySelector(selector)**: Returns the first element that matches a CSS selector.

javascript

var firstParagraph = document.querySelector('p'); o **document.querySelectorAll(selector)**: Returns a NodeList of elements that match a CSS selector.

javascript

var allParagraphs = document.querySelectorAll('p');

# 3. Manipulating DOM Elements

You can modify elements, their attributes, and their content using JavaScript: o **Change Text Content**:

javascript

var heading = document.getElementById('heading'); heading.textContent = 'New Title'; o **Change HTML Content**: javascript var container = document.getElementById('container'); container.innerHTML = '<h2>New Content</h2>'; o **Change Attributes**: javascript var link = document.getElementById('myLink'); link.href = 'https://www.new-url.com'; o **Add or Remove Classes**: javascript

var element = document.getElementById('myElement');

element.classList.add('new-class'); element.classList.remove('old-class'); o **Create and Append New Elements**: javascript var newDiv = document.createElement('div'); newDiv.textContent = 'This is a new div';

document.body.appendChild(newDiv);

4. **Event Handling**

You can respond to user interactions and other events using JavaScript:

# o Add Event Listeners: javascript

var button = document.getElementById('myButton'); button.addEventListener('click', function() {

alert('Button clicked!');

});

o **Handle Events Inline**:

html

<button onclick="alert('Button clicked!')">Click Me</button>

5. **Traversing the DOM**

You can navigate through the DOM tree to access related elements:

# o Parent Node: javascript

var child = document.getElementById('child'); var parent = child.parentNode; o **Child Nodes**: javascript

var parent = document.getElementById('parent'); var children = parent.childNodes; o **Sibling Nodes**: javascript

var sibling = document.getElementById('sibling'); var previous = sibling.previousSibling; var next = sibling.nextSibling;

# Summary

The HTML DOM provides a structured way to represent and interact with HTML documents. It allows developers to access, modify, and manipulate web pages dynamically, creating interactive and dynamic user experiences. Understanding the DOM is essential for web development, as it forms the foundation for client-side scripting and dynamic content updates.

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

jQuery is a popular, fast, and feature-rich JavaScript library that simplifies tasks such as HTML document traversal, event handling, animation, and AJAX interactions. It provides an easy-touse API that works across various browsers, making JavaScript development more accessible and efficient.

# Key Features of jQuery

* **Simplified Syntax**: Provides a concise syntax for common JavaScript tasks.
* **Cross-Browser Compatibility**: Handles browser differences and inconsistencies.
* **Event Handling**: Simplifies the process of attaching and managing events.
* **DOM Manipulation**: Facilitates easy manipulation of HTML elements and content.
* **AJAX**: Simplifies the process of making asynchronous HTTP requests.
* **Animation**: Offers built-in methods for animating elements.
* **Utilities**: Includes various utility functions to streamline coding.

# jQuery Selectors

jQuery selectors are used to select HTML elements that you want to manipulate. Here’s an overview of different jQuery selectors:

1. **Basic Selectors** o **\***: Selects all elements.

javascript

$('\*').css('color', 'blue'); // Sets the text color of all elements to blue o **element**: Selects all elements with the specified tag name.

javascript

$('p').hide(); // Hides all <p> elements o **#id**: Selects the element with the specified id.

javascript

$('#myElement').show(); // Shows the element with id="myElement" o **.class**: Selects all elements with the specified class name.

javascript

$('.myClass').addClass('active'); // Adds the class "active" to all elements with class="myClass"

2. **Attribute Selectors** o **[attribute]**: Selects elements with the specified attribute.

javascript

$('[type]').val('example'); // Sets the value of all elements with a 'type' attribute o **[attribute=value]**: Selects elements with the specified attribute value.

javascript

$('input[type="text"]').css('background-color', 'yellow'); // Changes the background color of text inputs

o **[attribute^=value]**: Selects elements with the specified attribute value that starts with a given value.

javascript

$('a[href^="https"]').attr('target', '\_blank'); // Sets the target to \_blank for all links starting with

'https' o **[attribute$=value]**: Selects elements with the specified attribute value that ends with a given value.

javascript

$('img[src$=".jpg"]').fadeIn(); // Fades in all images ending with '.jpg' o **[attribute\*=value]**: Selects elements with the specified attribute value containing a given value.

javascript

$('a[href\*=example]').css('color', 'red'); // Sets the color of links containing 'example' in their href

3. **Hierarchy Selectors** o **ancestor descendant**: Selects all descendant elements of the specified ancestor.

javascript

$('#container p').css('font-size', '16px'); // Sets the font size of all <p> elements inside #container o **ancestor > child**: Selects direct child elements of the specified ancestor.

javascript

$('.parent > .child').hide(); // Hides direct children with class "child" of elements with class

"parent" o **prev + next**: Selects the element immediately following the specified previous element.

javascript

$('h1 + p').css('color', 'green'); // Sets the color of the <p> element immediately following an <h1> element o **prev ~ siblings**: Selects all sibling elements following the specified previous element.

javascript

$('h1 ~ p').hide(); // Hides all <p> elements following an <h1> element

4. **Form Selectors** o **:input**: Selects all form input elements.

javascript

$(':input').val(''); // Clears the value of all form inputs o **:text**: Selects all text input elements.

javascript

$(':text').css('border', '1px solid red'); // Adds a red border to all text inputs o **:checked**: Selects all checked input elements (radio buttons, checkboxes).

javascript

$(':checked').prop('disabled', true); // Disables all checked inputs o **:selected**: Selects all selected options in a dropdown menu.

javascript

$('option:selected').css('font-weight', 'bold'); // Makes selected options bold

5. **Content Selectors** o **:first**: Selects the first element in the set of matched elements.

javascript

$('li:first').css('font-weight', 'bold'); // Makes the first <li> element bold

o **:last**: Selects the last element in the set of matched elements.

javascript

$('li:last').css('font-style', 'italic'); // Makes the last <li> element italic o **:eq(index)**: Selects the element at the specified index.

javascript

$('div:eq(2)').hide(); // Hides the third <div> element (0-based index) o **:odd**: Selects odd-indexed elements.

javascript

$('tr:odd').css('background-color', '#f2f2f2'); // Sets background color for odd rows in a table o **:even**: Selects even-indexed elements.

javascript

$('tr:even').css('background-color', '#e6e6e6'); // Sets background color for even rows in a table

Summary

jQuery selectors are powerful tools for selecting and manipulating HTML elements. By using these selectors, you can easily target elements based on their type, attributes, position, or relation to other elements, enabling dynamic and interactive web experiences.

**FAQ:**

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

Form validations are crucial for several reasons:

# 1. Improves Data Accuracy

Form validations ensure that the data entered by users meets the required format and constraints before it is submitted. This helps in preventing incorrect or incomplete data from being sent to the server. For example, validations can ensure that an email address is in the correct format or that a phone number contains only numeric characters. This improves the quality of the data collected and reduces the need for manual data correction or cleanup.

# 2. Enhances User Experience

Validations provide immediate feedback to users, guiding them to correct errors before submission. This helps in:

* **Preventing Form Submission Errors**: Users receive real-time notifications about incorrect or missing information, which can be fixed instantly, reducing frustration and the likelihood of form abandonment.
* **Streamlining the Submission Process**: By addressing issues as users fill out the form, validations make the process smoother and more intuitive, leading to a more satisfying user experience.

# 3. Increases Security

Validations help protect your application from malicious inputs and attacks by ensuring that user data meets specific criteria before it is processed. For instance:

* **Preventing Injection Attacks**: Input validation can help prevent SQL injection or script injection by ensuring that inputs do not contain harmful code or special characters.
* **Protecting Against Invalid Inputs**: Ensuring that inputs are in the correct format helps in avoiding errors that could lead to system vulnerabilities or crashes.

In summary, form validations are essential for maintaining data integrity, enhancing the user experience, and ensuring security, making them a critical aspect of web development and user interface design.

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

To modify an attribute value using the DOM, you can use JavaScript to access and change the attribute of an HTML element. Here's an example:

# Example Scenario

Suppose you have an HTML element with an id of myImage and a src attribute for an image: html

<img id="myImage" src="old-image.jpg" alt="Old Image">

# JavaScript Code to Modify the src Attribute

You can use JavaScript to modify the src attribute of this image element: javascript

// Access the element by its ID

var imgElement = document.getElementById('myImage');

// Modify the 'src' attribute

imgElement.setAttribute('src', 'new-image.jpg');

# Explanation

**1.Access the Element**: document.getElementById('myImage') retrieves the image element with the id of myImage.

**2.Modify the Attribute**: setAttribute('src', 'new-image.jpg') changes the value of the src attribute to 'new-image.jpg'.

After running this JavaScript code, the HTML element will be updated to:

html

<img id="myImage" src="new-image.jpg" alt="Old Image">

This example demonstrates how to modify an attribute's value using the DOM's setAttribute method.

**3. What is jQuery Ajax?**

jQuery AJAX (Asynchronous JavaScript and XML) is a technique for creating asynchronous web requests in jQuery, which allows you to interact with a server without needing to reload the entire page. AJAX is used to send and receive data from a web server asynchronously, enabling more dynamic and responsive web applications**.**

# Key Features of jQuery AJAX

1. Asynchronous Communication: Allows web pages to update content dynamically without requiring a full page refresh, improving user experience and responsiveness.
2. Simplified Syntax: Provides a simple and intuitive API for making HTTP requests and handling responses, making it easier to work with AJAX in jQuery.
3. Data Formats: Supports various data formats, including JSON, XML, HTML, and text, enabling flexible data exchange between the client and server.

Common jQuery AJAX Methods

1. .ajax()

The .ajax() method is the core method for making AJAX requests. It provides a high degree of flexibility and control over the request. javascript $.ajax({ url: 'server-endpoint.php', type: 'GET', // or 'POST'

dataType: 'json', // Expected data type from the server success: function(response) {

// Handle success

console.log(response);

},

error: function(jqXHR, textStatus, errorThrown) {

// Handle error

console.error('Error:', textStatus, errorThrown);

}

});

1. .get()

The .get() method is a shorthand for making GET requests. It is used to retrieve data from a server. javascript

Copy code

$.get('server-endpoint.php', { key: 'value' }, function(response) {

// Handle success console.log(response);

}, 'json'); // Specify the expected data type

1. .post()

The .post() method is a shorthand for making POST requests. It is used to send data to a server. javascript

Copy code

$.post('server-endpoint.php', { key: 'value' }, function(response) {

// Handle success console.log(response);

}, 'json'); // Specify the expected data type

1. .getJSON()

The .getJSON() method is a shorthand for making GET requests that expect a JSON response. javascript

$.getJSON('server-endpoint.php', { key: 'value' }, function(response) {

// Handle success console.log(response);

});

1. .load()

The .load() method is used to load HTML content from the server and insert it into a selected element. javascript

$('#container').load('content.html', function(response, status, xhr) { if (status == 'error') {

console.error('Error loading content:', xhr.status, xhr.statusText);

}

});

Example of jQuery AJAX

Here’s an example of making a simple GET request using jQuery AJAX: html

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>jQuery AJAX Example</title>

<script src="https://code.jquery.com/jquery-3.6.0.min.js"></script>

</head>

<body>

<button id="loadData">Load Data</button>

<div id="result"></div>

<script>

$('#loadData').on('click', function() {

$.ajax({

url: 'https://api.example.com/data',

type: 'GET', dataType: 'json', success: function(data) {

$('#result').html('<p>' + data.message + '</p>');

},

error: function(jqXHR, textStatus, errorThrown) { $('#result').html('<p>Error: ' + textStatus + '</p>');

}

});

});

</script>

</body>

</html>

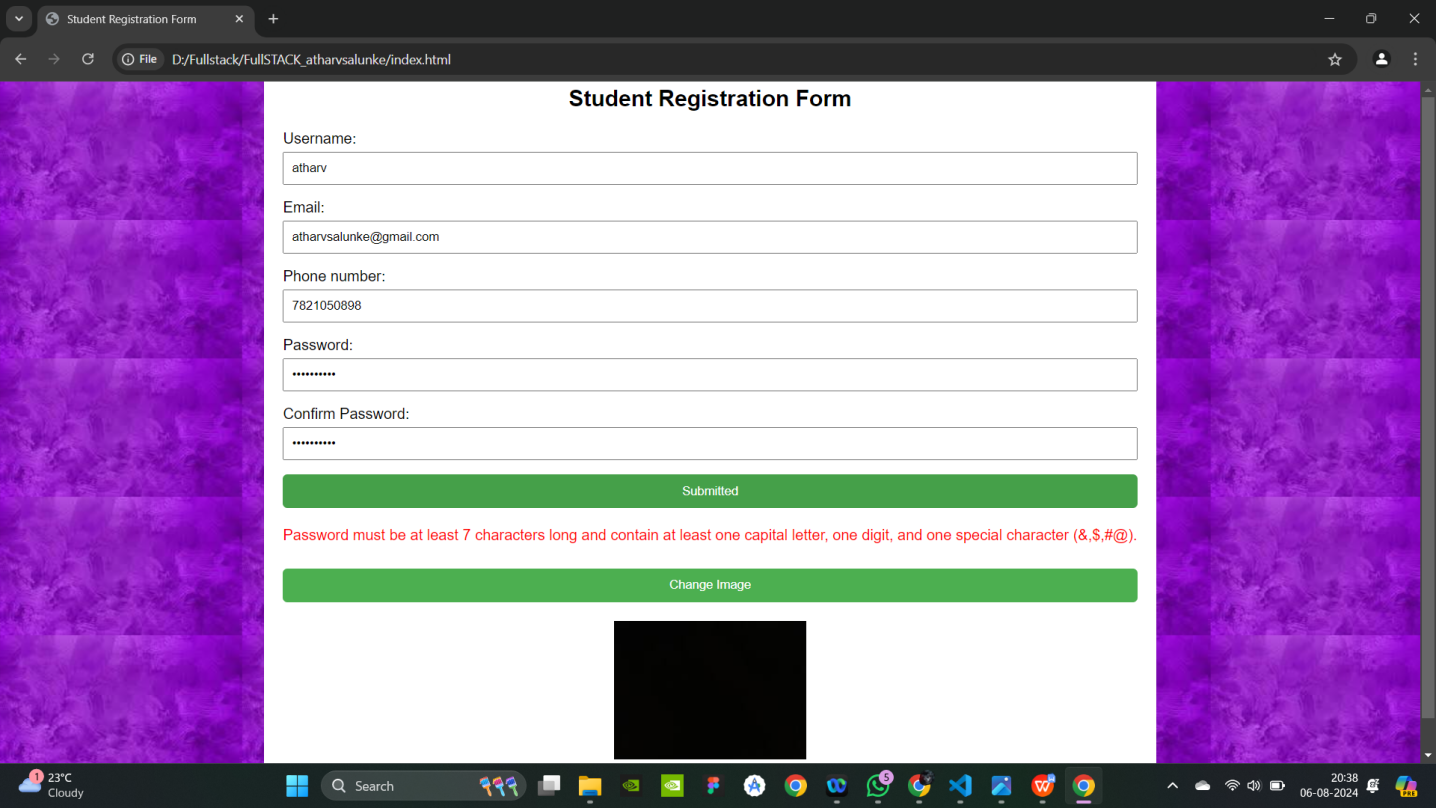
In this example:

* Clicking the button triggers an AJAX GET request to https://api.example.com/data.
* On success, the response is displayed in the #result div.
* On error, an error message is shown.

Summary

jQuery AJAX simplifies the process of making asynchronous requests and handling responses, enabling dynamic updates and interactions in web applications without full page reloads. It offers various methods for different types of requests and data formats, making it a powerful tool for modern web development.

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



**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/jqueryexercises/part1/index.php