FSD LAB 3

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 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 validation is an essential aspect of web development to ensure that the data submitted by users is accurate, complete, and secure. There are various types of form validations :

Required validation, Length Validation, Numeric Validation, Email validation, Password Validation, Password Confirmation, Checkbox validation(Ex – terms and conditions), Date validation(DOB).

2. HTML Document Object Model.

The Document Object Model (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, text, etc. The DOM provides a way for programs to manipulate the structure, style, and content of HTML and XML documents.

In the context of HTML, the DOM is often referred to as the HTML DOM. The HTML DOM represents an HTML document as a tree structure in which each node corresponds to an HTML element, attribute, text, or comment. This tree structure allows developers to access, manipulate, and update the content and structure of an HTML document dynamically.

3. What is JQuery? Write various JQuery Selectors.

jQuery is a fast, small, and feature-rich JavaScript library. It simplifies HTML document traversal and manipulation, event handling, animation, and AJAX interactions. jQuery is designed to make things like HTML document traversal and manipulation, event handling, and animation much simpler with an easy-to-use API that works across a multitude of browsers.

jQuery uses a set of selectors to easily and efficiently select elements from the HTML document. Here are some common 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 specified ID attribute.

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

Class Selector:

Selects all elements with a specified class attribute.

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

Attribute Selector:

Selects elements based on the presence or value of a specific attribute.

Example: $("[name='myName']") selects elements with name="myName".

Multiple Selectors:

Allows you to combine multiple selectors into a single selector.

Example: $("p, .myClass, #myId") selects all <p> elements, elements with class="myClass", and the element with id="myId".

Descendant Selector:

Selects all elements that are descendants of a given ancestor.

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

Child Selector:

Selects all direct children of a specified parent.

Example: $("div > p") selects all <p> elements that are direct children of a <div>.

First Selector:

Selects the first element in the set of matched elements.

Example: $("p:first") selects the first <p> element.

Last Selector:

Selects the last element in the set of matched elements.

Example: $("p:last") selects the last <p> element.

Even and Odd Selectors:

Selects even or odd elements in a set of matched elements.

Example: $("tr:even") selects all even <tr> elements in a table.

Not Selector:

Selects all elements that do not match a specified selector.

Example: $("p:not(.myClass)") selects all <p> elements that do not have class="myClass".

FAQ:

1. Write 3 reasons why Form validations are important.
   Data Accuracy:
   Form validations help ensure that the data submitted by users is accurate and meets the expected format. This is essential for maintaining the integrity of the data stored in a database or processed by the server.
   Without proper validation, users might submit incomplete or incorrect information, leading to issues such as data inconsistencies, errors in processing, or incorrect results in applications.

   Enhanced User Experience:

Form validations contribute to a positive user experience by providing immediate feedback to users about the correctness of their inputs. When users receive prompt feedback on errors or incorrect inputs, they can correct mistakes quickly, reducing frustration and improving the overall usability of the website or application.
Well-designed form validations guide users through the input process, making it clear what information is required and how it should be formatted. This helps prevent user confusion and errors during data submission.

Security and Data Integrity:
Form validations play a crucial role in enhancing the security of web applications. By validating user inputs on the client side (using JavaScript) and revalidating on the server side, developers can prevent common security vulnerabilities such as SQL injection, cross-site scripting (XSS), and other injection attacks.
Validations help ensure that the data submitted does not contain malicious code or unintended characters that could be exploited by attackers. Additionally, they can prevent the submission of data that may disrupt the functionality of the application or compromise the integrity of the database.

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

This is our original code

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Modify Attribute Example</title>
</head>
<body>

  <p id="myParagraph" class="originalClass">This is a paragraph.</p>

  <script src="your-script.js"></script>

</body>
</html>
```

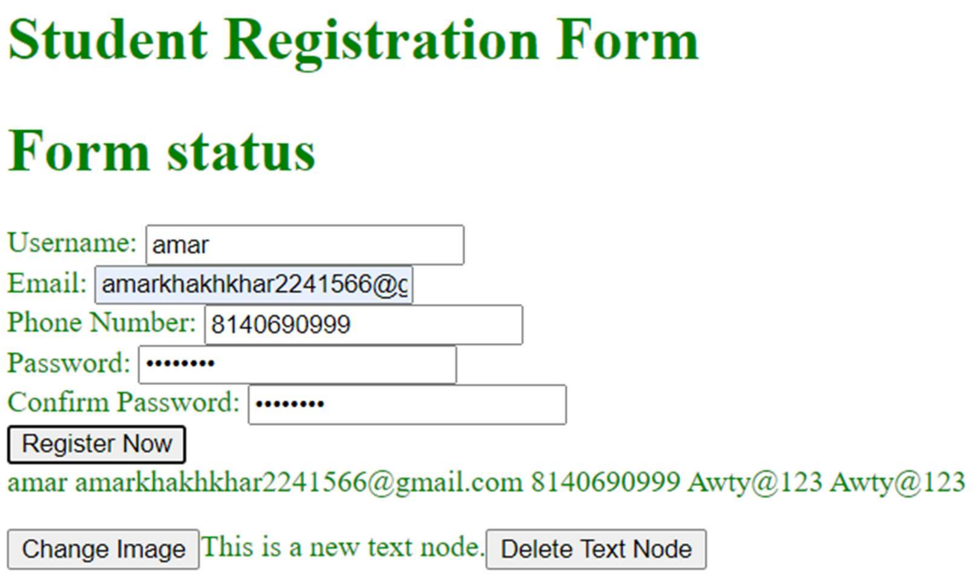
We modify it using DOM in following way

```
// Get the element by its ID
var myParagraph = document.getElementById("myParagraph");

// Display the current class attribute value
console.log("Current class attribute value:", myParagraph.className);

// Modify the class attribute value
myParagraph.className = "newClass";

// Display the updated class attribute value
console.log("Updated class attribute value:", myParagraph.className);
```

3. What is jQuery Ajax?

   jQuery Ajax (Asynchronous JavaScript and XML) refers to the set of techniques used in jQuery to make asynchronous HTTP requests to a server and handle the server's response. Despite its name, Ajax is not limited to XML, and modern usage often involves JSON (JavaScript Object Notation) as the preferred data format. Ajax allows web pages to be updated asynchronously by exchanging small amounts of data with the server behind the scenes. This leads to a more dynamic and responsive user experience.

Screenshots:

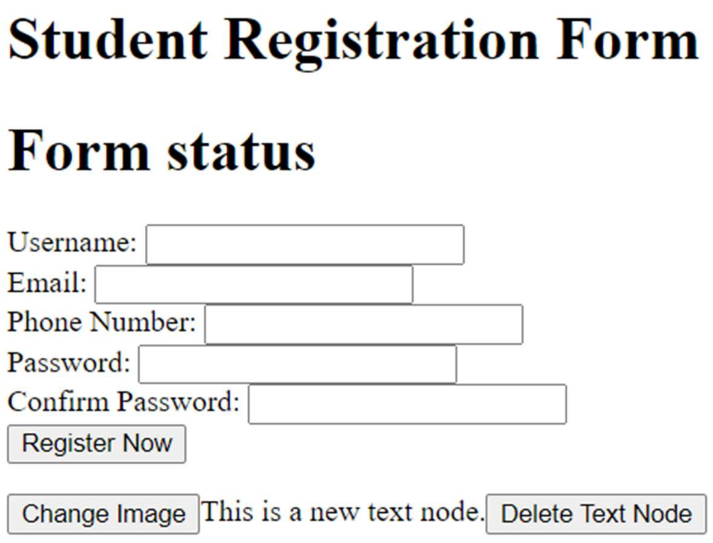


Student Registration Form

Form status

Username: amar
Email: amarkhakhkhar2241566@g
Phone Number: 8140690999
Password: ••••••••
Confirm Password: ••••••••
Register Now
amar amarkhakhkhar2241566@gmail.com 8140690999 Awty@123 Awty@123

Change Image This is a new text node. Delete Text Node



Student Registration Form

Form status

Username:
Email:
Phone Number:
Password:
Confirm Password:
Register Now

Change Image This is a new text node. Delete Text Node

CODE –

```
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Student Registration</title>
    <style>
```

```html
        /* Add CSS styles here */
        .error {
            color: red;
        }
    </style>
</head>

<body>
    <h1>Student Registration Form</h1>
    <form id="registrationForm">
        <div>
            <h1 id="status">Form status</h1>
        </div>
        <div>
            <label for="username">Username:</label>
            <input type="text" id="username" name="username" required>
        </div>
        <div>
            <label for="email">Email:</label>
            <input type="email" id="email" name="email" required
pattern="/^[a-zA-Z0-9._%+-]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,3}$/
            ">
        </div>
        <div>
            <label for="phone">Phone Number:</label>
            <input type="tel" id="phone" name="phone" required pattern="[0-
9]{10}">
        </div>
        <div>
            <label for="password">Password:</label>
            <input type="password" id="password" name="password" required
                pattern="^(?=.*[A-Z])(?=.*[0-9])(?=.*[&$#@]).{7,}$">
        </div>
        <div>
            <label for="confirmPassword">Confirm Password:</label>
            <input type="password" id="confirmPassword" name="confirmPassword"
required>
        </div>
        <button type="submit">Register</button>
        <div id="data">

        </div>
    </form>

    <p class="error" id="errorMessage"></p>

    <script>
        // JavaScript for form validation
        const form = document.getElementById("registrationForm");
        const errorMessage = document.getElementById("errorMessage");

        form.addEventListener("submit", function (event) {
            event.preventDefault();

            const username = document.getElementById("username").value;
            const email = document.getElementById("email").value;
            const phone = document.getElementById("phone").value;
            const password = document.getElementById("password").value;
            const confirmPassword =
document.getElementById("confirmPassword").value;

            if (username === "" || email === "" || phone === "" || password
=== "" || confirmPassword === "" || password != confirmPassword) {
                document.body.style.color = 'red'
                errorMessage.innerHTML = "Validation failed. Please check your
input.";
            }
            else {
                document.body.style.color = 'green'
            }

            // Display error message if validation fails
        });

        // Change image source on button click
        const changeImageBtn = document.createElement("button");
        changeImageBtn.textContent = "Change Image";
        document.body.appendChild(changeImageBtn);

        changeImageBtn.addEventListener("click", function () {
            const imageElement = document.createElement("img");
            imageElement.src = "new-image.jpg";
            document.body.appendChild(imageElement);
        });

        // Add a text node
        const textNode = document.createTextNode("This is a new text node.");
        document.body.appendChild(textNode);

        // Delete a node
        const deleteBtn = document.createElement("button");
        deleteBtn.textContent = "Delete Text Node";
        document.body.appendChild(deleteBtn);

        deleteBtn.addEventListener("click", function () {
            document.body.removeChild(textNode);
        });
    </script>

    <!-- Include jQuery from CDN -->
    <script src="https://code.jquery.com/jquery-3.6.0.min.js"></script>
    <script>
        // jQuery operations
        $(document).ready(function () {
            // Change button text using jQuery
            $("button[type='submit']").text("Register Now");

            // Set background-image using jQuery CSS property
            $("body").css("background-image", "url('background.jpg')");

            // Access HTML form data using jQuery
            $("form").submit(function (event) {

                const form = document.getElementById("data");
                event.preventDefault();
                const formData = $(this).serializeArray();
                jQuery.each( formData, function( i, field ) {
      $( "#data" ).append( field.value + " " );
    } );
            });

            // Add attribute using jQuery
            $("img").attr("alt", "New Image");
        });
    </script>
</body>

</html>
```