**PART A**

**Experiment No. 01**

**A.1 AIM:**

Form validation using html and javascipt validators and formatting form using CSS and Bootstrap

**A.2 Pre requisite:**

Basic Knowledge of HTML ,CSS ,Regex,BootStrap and Javascript tags

**A.3 Outcome:**

After successful completion of this experiment students will be able to understand to concept of Form validation using

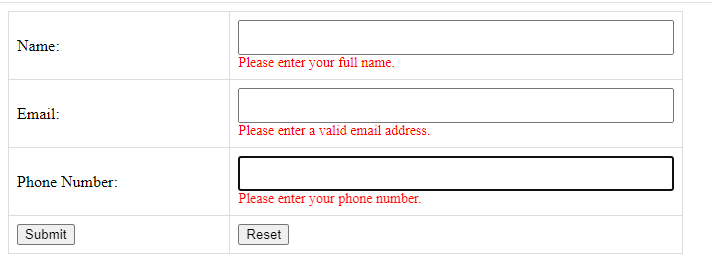
**Problem1: create a web page for a website as give below**

1. Create a Simple form as show below

* **Full Name Field**:
  + Input field for entering the full name (size: 25 characters).
  + Error Handling: Displays error messages if the name is missing or too short.
* **Email Field**:
  + Label for the email input field.
  + Input field for entering an email address (size: 25 characters).
  + Error Handling: Displays error messages if the email is invalid or missing.
* **Phone Number Field**:
  + Label for the phone number input field.
  + Input field for entering a phone number (size: 25 characters).
  + Error Handling: Displays error messages if the phone number is missing or contains non-digit characters.
* **Submit Button**:
  + Button to submit the form.
* **Reset Button**:
  + Button to reset the form fields to their default values.



On submitting the form without any data it show



**2 Apply the Validation Rules given below**

* **Name Validation**:
  + Checks if the name field is empty and displays an error if true.
  + Ensures the name has at least 2 characters.
* **Email Validation**:
  + Ensures the email field is not empty and matches a valid email pattern using a regular expression.
* **Phone Number Validation**:
  + Checks if the phone number field is empty and displays an error if true.
  + Ensures the phone number contains only digits using the regular expression ^\d+$.

Prepare the document. Save and close the file and name it as **EXP01\_Roll no\_Batch no.**

**Problem2: using the layout of above web page create a separate web page and apply bootstrap for submit and reset button as well as the tittle of the Form**

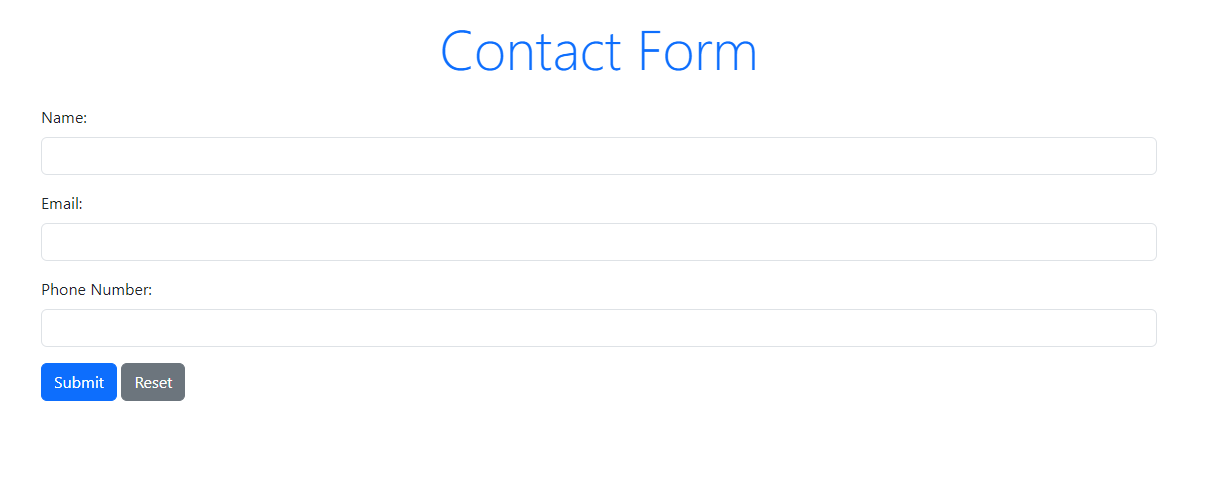
Include Bootstrap CSS or download the CSS and scripts from <https://getbootstrap.com/>

    <!-- Bootstrap CSS -->

    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/css/bootstrap.min.css" rel="stylesheet">

<!-- Bootstrap JS (with Popper) -->

    <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/js/bootstrap.bundle.min.js"></script>

****

**For changing the style of the button use and formatting the title of the form**

**Use**

  <div class="mb-3">

                <button type="submit" class="btn btn-primary">Submit</button>

                <button type="reset" class="btn btn-secondary">Reset</button>

            </div>

<h2 class="text-center text-primary mb-4 display-4">Contact Form</h2>

**PART B**

(PART B: TO BE COMPLETED BY STUDENTS)

(Students must submit the soft copy as per following segments within two hours of the practical. The soft copy must be uploaded on the Blackboard or emailed to the concerned lab in charge faculties at the end of the practical in case the there is no Black board access available)

|  |  |
| --- | --- |
| Roll No. : S020 | Name: Husain Chhil |
| Class : MBATech DS | Batch : J1 |
| Date of Experiment : 200924 | Date/Time of Submission : 200924 |
| Grade : |  |

**B.1 Code:**

**Problem 1**

<!DOCTYPE *html*>

<html *lang*="en">

<head>

    <meta *charset*="UTF-8">

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

    <title>Simple Form</title>

    <style>

        body {

            font-family: Arial, sans-serif;

            margin: 50px;

        }

*.error* {

            color: red;

            font-size: 0.9em;

        }

        input[*type*="text"], input[*type*="email"] {

            width: 300px;

            padding: 10px;

            margin: 10px 0;

        }

        button {

            padding: 10px 20px;

            margin: 5px;

        }

    </style>

</head>

<body>

    <h2>Contact Form</h2>

    <form *id*="contactForm">

        <label *for*="fullName">Full Name:</label><br>

        <input *type*="text" *id*="fullName" *name*="fullName" *maxlength*="25">

        <div *class*="error" *id*="nameError"></div><br>

        <label *for*="email">Email:</label><br>

        <input *type*="email" *id*="email" *name*="email" *maxlength*="25">

        <div *class*="error" *id*="emailError"></div><br>

        <label *for*="phone">Phone Number:</label><br>

        <input *type*="text" *id*="phone" *name*="phone" *maxlength*="25">

        <div *class*="error" *id*="phoneError"></div><br>

        <button *type*="submit">Submit</button>

        <button *type*="reset" *onclick*="resetForm()">Reset</button>

    </form>

    <script>

        document.getElementById('contactForm').addEventListener('submit', function(*event*) {

*event*.preventDefault(); // Prevent the default form submission

            // Clear previous error messages

            document.getElementById('nameError').innerText = '';

            document.getElementById('emailError').innerText = '';

            document.getElementById('phoneError').innerText = '';

            // Get input values

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

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

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

            // Validation flags

            let isValid = true;

            // Name validation

            if (fullName === '' || fullName.length < 2) {

                document.getElementById('nameError').innerText = 'Full Name must be at least 2 characters long.';

                isValid = false;

            }

            // Email validation

            const emailPattern = /^[^\s@]+@[^\s@]+\.[^\s@]+$/;

            if (email === '' || !emailPattern.test(email)) {

                document.getElementById('emailError').innerText = 'Please enter a valid email address.';

                isValid = false;

            }

            // Phone number validation

            const phonePattern = /^\d+$/;

            if (phone === '' || !phonePattern.test(phone)) {

                document.getElementById('phoneError').innerText = 'Phone Number must contain only digits.';

                isValid = false;

            }

            // If valid, show success message (can be replaced with actual submission logic)

            if (isValid) {

                alert('Form submitted successfully!');

            }

        });

        function resetForm() {

            document.getElementById('nameError').innerText = '';

            document.getElementById('emailError').innerText = '';

            document.getElementById('phoneError').innerText = '';

        }

    </script>

</body>

</html>

**Problem 2**

<!DOCTYPE *html*>

<html *lang*="en">

<head>

    <meta *charset*="UTF-8">

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

    <title>Simple Form</title>

    <!-- Bootstrap CSS -->

    <link *href*="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/css/bootstrap.min.css" *rel*="stylesheet">

    <style>

*.error* {

            color: red;

            font-size: 0.9em;

        }

    </style>

</head>

<body>

    <div *class*="container mt-5">

        <h2 *class*="text-center text-primary mb-4 display-4">Contact Form</h2>

        <form *id*="contactForm">

            <div *class*="mb-3">

                <label *for*="fullName" *class*="form-label">Full Name:</label>

                <input *type*="text" *class*="form-control" *id*="fullName" *name*="fullName" *maxlength*="25">

                <div *class*="error" *id*="nameError"></div>

            </div>

            <div *class*="mb-3">

                <label *for*="email" *class*="form-label">Email:</label>

                <input *type*="email" *class*="form-control" *id*="email" *name*="email" *maxlength*="25">

                <div *class*="error" *id*="emailError"></div>

            </div>

            <div *class*="mb-3">

                <label *for*="phone" *class*="form-label">Phone Number:</label>

                <input *type*="text" *class*="form-control" *id*="phone" *name*="phone" *maxlength*="25">

                <div *class*="error" *id*="phoneError"></div>

            </div>

            <div *class*="mb-3">

                <button *type*="submit" *class*="btn btn-primary">Submit</button>

                <button *type*="reset" *class*="btn btn-secondary" *onclick*="resetForm()">Reset</button>

            </div>

        </form>

    </div>

    <!-- Bootstrap JS (with Popper) -->

    <script *src*="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/js/bootstrap.bundle.min.js"></script>

    <script>

        document.getElementById('contactForm').addEventListener('submit', function(*event*) {

*event*.preventDefault(); // Prevent the default form submission

            // Clear previous error messages

            document.getElementById('nameError').innerText = '';

            document.getElementById('emailError').innerText = '';

            document.getElementById('phoneError').innerText = '';

            // Get input values

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

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

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

            // Validation flags

            let isValid = true;

            // Name validation

            if (fullName === '' || fullName.length < 2) {

                document.getElementById('nameError').innerText = 'Full Name must be at least 2 characters long.';

                isValid = false;

            }

            // Email validation

            const emailPattern = /^[^\s@]+@[^\s@]+\.[^\s@]+$/;

            if (email === '' || !emailPattern.test(email)) {

                document.getElementById('emailError').innerText = 'Please enter a valid email address.';

                isValid = false;

            }

            // Phone number validation

            const phonePattern = /^\d+$/;

            if (phone === '' || !phonePattern.test(phone)) {

                document.getElementById('phoneError').innerText = 'Phone Number must contain only digits.';

                isValid = false;

            }

            // If valid, show success message (can be replaced with actual submission logic)

            if (isValid) {

                alert('Form submitted successfully!');

            }

        });

        function resetForm() {

            document.getElementById('nameError').innerText = '';

            document.getElementById('emailError').innerText = '';

            document.getElementById('phoneError').innerText = '';

        }

    </script>

</body>

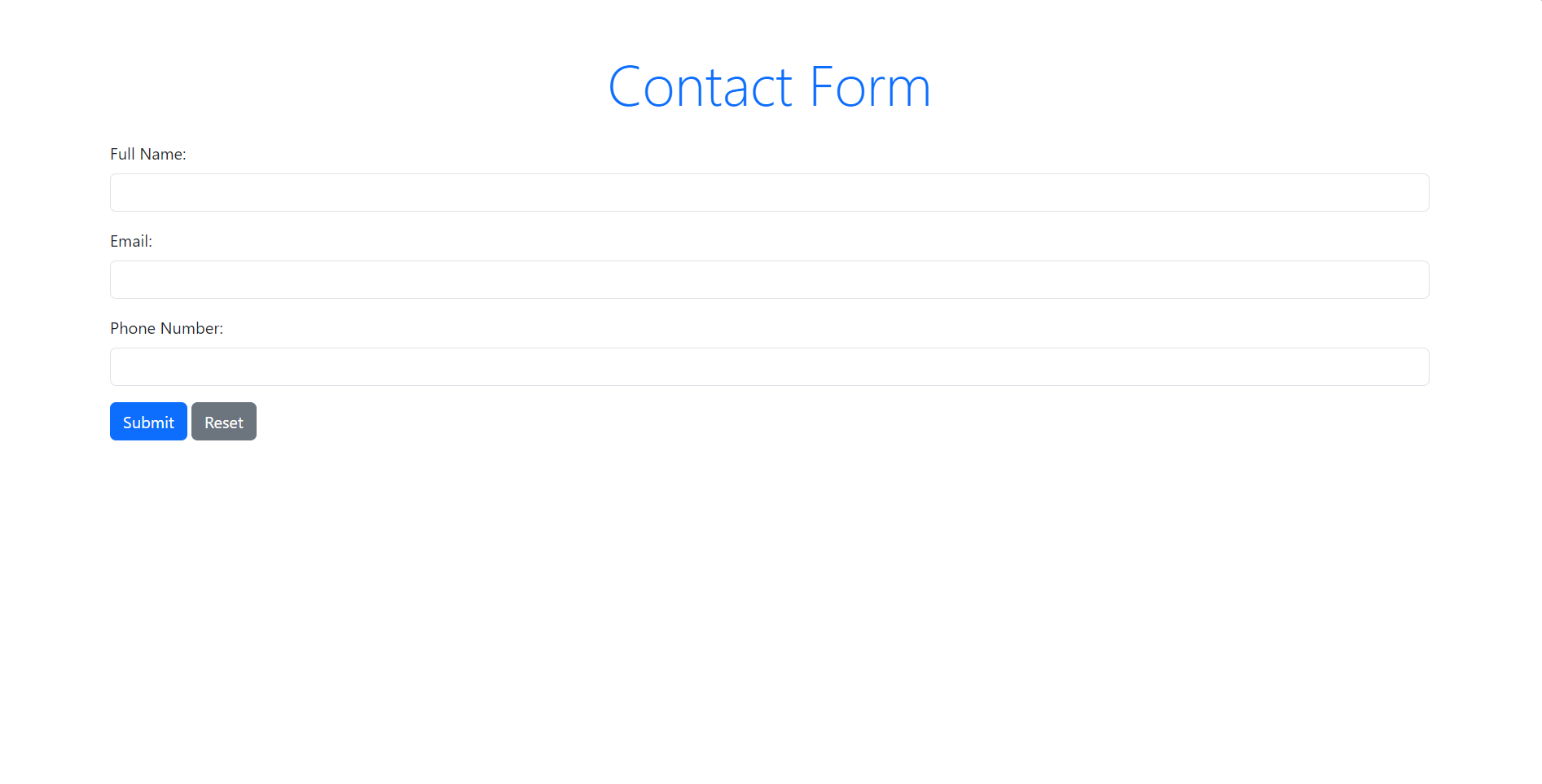
</html>

**B.2 Output**

Problem 1



Problem 2



**B.3 Conclusion:**

In this practical exercise, I have attained a comprehensive understanding of JavaScript functions and the Bootstrap framework. The key outcomes achieved include:

* **Understanding JavaScript Functions**: I learned about different types of functions, including function declarations, function expressions, and arrow functions. This understanding allowed me to appreciate how functions are structured in JavaScript and their applicability in various contexts.
* **Application of Bootstrap**: I gained hands-on experience using Bootstrap for responsive web design. This included knowledge of its grid system, components, and utility classes, which enhanced my ability to create visually appealing and functional web pages.

**B.3 Observations and Learning:**

Throughout the practical, I made several observations and gained valuable insights:

* **JavaScript Functions**:
  + **Function Declarations**: I observed that function declarations are hoisted, meaning they can be called before they are defined in the code.
  + **Function Expressions**: I learned that function expressions are not hoisted, which emphasizes the importance of the order in which functions are defined.
  + **Arrow Functions**: The concise syntax of arrow functions simplifies function definitions, especially for callbacks and array methods.

**Bootstrap Framework**:

* **Responsive Design**: I found that Bootstrap's responsive grid system is instrumental in creating layouts that adapt to different screen sizes seamlessly.
* **Component Utilization**: Utilizing pre-built components like buttons, forms, and navigation bars saved time and ensured consistency in design.
* **Customization**: I learned how to customize Bootstrap styles using CSS, which allowed for personalized branding while leveraging Bootstrap's robust framework.

**B.4 Question of Curiosity**

**Q1. Explain Types of JavaScript Functions?** JavaScript functions can be categorized into several types:

1. **Function Declarations**: Defined using the function keyword, these functions are hoisted and can be called before their definition.

function greet() {

console.log("Hello!");

}

1. **Function Expressions**: These are functions defined within an expression, not hoisted, and can be assigned to variables.

const greet = function() {

console.log("Hello!");

};

1. **Arrow Functions**: Introduced in ES6, these provide a shorter syntax for writing functions. They do not have their own this, making them suitable for use as callbacks.

const greet = () => {

console.log("Hello!");

};

1. **IIFE (Immediately Invoked Function Expression)**: A function that runs as soon as it is defined. This is useful for encapsulating code.

(function() {

console.log("This runs immediately!");

})();

**Q2. What is Bootstrap?** Bootstrap is an open-source front-end framework designed to simplify the process of web development. It provides a collection of CSS and JavaScript components that facilitate the creation of responsive and visually appealing websites. Initially developed by Twitter, Bootstrap allows developers to use pre-defined styles and components, which speeds up the design process and enhances cross-browser compatibility.

**Q3. What are the uses of Bootstrap?** Bootstrap has numerous uses, including:

1. **Responsive Design**: Bootstrap's grid system and responsive utilities help create layouts that adapt to various screen sizes, ensuring optimal viewing on devices like smartphones, tablets, and desktops.
2. **Pre-Built Components**: It offers a wide array of pre-designed components such as buttons, navigation bars, modals, forms, and alerts, allowing developers to build functional UI elements quickly.
3. **Consistency**: By providing a uniform design framework, Bootstrap ensures consistency across different web pages and projects, enhancing the user experience.
4. **Customization**: Bootstrap can be easily customized through its built-in variables and CSS classes, enabling developers to create a unique look while retaining functionality.
5. **Cross-Browser Compatibility**: Bootstrap is designed to work seamlessly across major browsers, minimizing the need for extensive testing and adjustments.