**Exercise 1: Validate a Phone Number**

**Problem**: Write a regular expression to validate US phone numbers in the format (123) 456-7890 or 123-456-7890.

**Solution**:

const regex = /^\(?\d{3}\)?[-\s]?\d{3}[-\s]?\d{4}$/;

console.log(regex.test('(123) 456-7890')); // true

console.log(regex.test('123-456-7890')); // true

console.log(regex.test('123 456 7890')); // true

console.log(regex.test('1234567890')); // false

**Exercise 2: Validate an Email Address**

**Problem**: Write a regular expression to validate email addresses with the format username@domain.com. Ensure that the domain part contains at least one period.

**Solution**:

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

console.log(regex.test('user@example.com')); // true

console.log(regex.test('user@domain.co')); // true

console.log(regex.test('user@domain')); // false

console.log(regex.test('user@.com')); // false

**Exercise 3: Extract Dates from a String**

**Problem**: Write a regular expression to extract dates in the format DD/MM/YYYY from a string.

**Solution**:

const regex = /\b(\d{2})\/(\d{2})\/(\d{4})\b/g;

const str = 'The event is on 05/12/2024 and another one on 31/12/2024.';

const matches = [...str.matchAll(regex)];

console.log(matches.map(m => m[0])); // ['05/12/2024', '31/12/2024']

**Exercise 4: Validate a Strong Password**

**Problem**: Write a regular expression to validate a password that is at least 8 characters long, contains at least one uppercase letter, one lowercase letter, one digit, and one special character.

**Solution**:

const regex = /^(?=.\*[a-z])(?=.\*[A-Z])(?=.\*\d)(?=.\*[@$!%\*?&])[A-Za-z\d@$!%\*?&]{8,}$/;

console.log(regex.test('Password1!')); // true

console.log(regex.test('password123')); // false

console.log(regex.test('PASSWORD123!')); // false

console.log(regex.test('Pass1')); // false

**Exercise 5: Validate a Hexadecimal Color Code**

**Problem**: Write a regular expression to validate hexadecimal color codes. These can be in the format #RRGGBB or #RGB.

**Solution**:

const regex = /^#([0-9A-Fa-f]{3}|[0-9A-Fa-f]{6})$/;

console.log(regex.test('#FFFFFF')); // true

console.log(regex.test('#FFF')); // true

console.log(regex.test('#123ABC')); // true

console.log(regex.test('#123ABCG')); // false

**Exercise 6: Extract All URLs from a Text**

**Problem**: Write a regular expression to extract all URLs from a text. URLs should start with http:// or https://.

**Solution**:

const regex = /https?:\/\/[^\s]+/g;

const text = 'Visit our site at https://example.com and http://example.org for more information.';

const urls = text.match(regex);

console.log(urls); // ['https://example.com', 'http://example.org']

**Exercise 7: Match a Valid IPv4 Address**

**Problem**: Write a regular expression to match valid IPv4 addresses.

**Solution**:

const regex = /^(25[0-5]|2[0-4][0-9]|[01]?[0-9][0-9]?)\.(25[0-5]|2[0-4][0-9]|[01]?[0-9][0-9]?)\.(25[0-5]|2[0-4][0-9]|[01]?[0-9][0-9]?)\.(25[0-5]|2[0-4][0-9]|[01]?[0-9][0-9]?)$/;

console.log(regex.test('192.168.1.1')); // true

console.log(regex.test('255.255.255.255')); // true

console.log(regex.test('256.256.256.256')); // false

**Exercise 8: Match a Valid Roman Numeral**

**Problem**: Write a regular expression to match Roman numerals from I to MMXXI.

**Solution**:

const regex = /^(M{0,3})(CM|CD|D?C{0,3})(XC|XL|L?X{0,3})(IX|IV|V?I{0,3})$/;

console.log(regex.test('MMXXI')); // true

console.log(regex.test('IV')); // true

console.log(regex.test('MMMCMXC')); // true

console.log(regex.test('MMMM')); // false

**Exercise 9: Validate a US ZIP+4 Code**

**Problem**: Write a regular expression to validate US ZIP+4 codes (e.g., 12345-6789).

**Solution**:

const regex = /^\d{5}-\d{4}$/;

console.log(regex.test('12345-6789')); // true

console.log(regex.test('12345')); // false

console.log(regex.test('1234-5678')); // false

**Exercise 10: Match Words with Apostrophes**

**Problem**: Write a regular expression to match words that may contain apostrophes (e.g., "don't").

**Solution**:

const regex = /^[A-Za-z']+$/;

console.log(regex.test("don't")); // true

console.log(regex.test("John")); // true

console.log(regex.test("O'Conner")); // true

console.log(regex.test("O@Conner")); // false

**Exercise 11: Validate a Username**

**Problem**: Create a form where the username must be 5-15 characters long and contain only letters, numbers, and underscores. Display an error message in red if the username is invalid.

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

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

<title>Username Validation</title>

<style>

body {

font-family: Arial, sans-serif;

padding: 20px;

}

form {

max-width: 300px;

margin: auto;

}

label {

display: block;

margin-bottom: 8px;

}

input[type="text"] {

width: 100%;

padding: 8px;

margin-bottom: 10px;

border: 1px solid #ccc;

border-radius: 4px;

}

.error {

color: red;

display: none;

font-size: 0.875em;

}

input[type="submit"] {

background-color: #4CAF50;

color: white;

padding: 10px 20px;

border: none;

border-radius: 4px;

cursor: pointer;

}

input[type="submit"]:hover {

background-color: #45a049;

}

</style>

</head>

<body>

<form id="usernameForm">

<label for="username">Username:</label>

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

<span id="usernameError" class="error">Username must be 5-15 characters long and contain only letters, numbers, and underscores.</span>

<br><br>

<input type="submit" value="Submit">

</form>

<script>

const form = document.getElementById('usernameForm');

const usernameInput = document.getElementById('username');

const usernameError = document.getElementById('usernameError');

const usernameRegex = /^[a-zA-Z0-9\_]{5,15}$/;

form.addEventListener('submit', (event) => {

if (!usernameRegex.test(usernameInput.value)) {

usernameError.style.display = 'block';

event.preventDefault(); // Prevent form submission

} else {

usernameError.style.display = 'none';

}

});

</script>

</body>

</html>

**Exercise 12: Validate an Email Address**

**Problem**: Create a form to validate email addresses. Display an error message in red if the email is invalid.

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

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

<title>Email Validation</title>

<style>

body {

font-family: Arial, sans-serif;

padding: 20px;

}

form {

max-width: 300px;

margin: auto;

}

label {

display: block;

margin-bottom: 8px;

}

input[type="email"] {

width: 100%;

padding: 8px;

margin-bottom: 10px;

border: 1px solid #ccc;

border-radius: 4px;

}

.error {

color: red;

display: none;

font-size: 0.875em;

}

input[type="submit"] {

background-color: #4CAF50;

color: white;

padding: 10px 20px;

border: none;

border-radius: 4px;

cursor: pointer;

}

input[type="submit"]:hover {

background-color: #45a049;

}

</style>

</head>

<body>

<form id="emailForm">

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

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

<span id="emailError" class="error">Please enter a valid email address.</span>

<br><br>

<input type="submit" value="Submit">

</form>

<script>

const form = document.getElementById('emailForm');

const emailInput = document.getElementById('email');

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

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

form.addEventListener('submit', (event) => {

if (!emailRegex.test(emailInput.value)) {

emailError.style.display = 'block';

event.preventDefault(); // Prevent form submission

} else {

emailError.style.display = 'none';

}

});

</script>

</body>

</html>

**Exercise13: Validate a Phone Number**

**Problem**: Create a form to validate US phone numbers in the format (123) 456-7890 or 123-456-7890. Display an error message in red if the phone number is invalid.

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

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

<title>Phone Number Validation</title>

<style>

body {

font-family: Arial, sans-serif;

padding: 20px;

}

form {

max-width: 300px;

margin: auto;

}

label {

display: block;

margin-bottom: 8px;

}

input[type="text"] {

width: 100%;

padding: 8px;

margin-bottom: 10px;

border: 1px solid #ccc;

border-radius: 4px;

}

.error {

color: red;

display: none;

font-size: 0.875em;

}

input[type="submit"] {

background-color: #4CAF50;

color: white;

padding: 10px 20px;

border: none;

border-radius: 4px;

cursor: pointer;

}

input[type="submit"]:hover {

background-color: #45a049;

}

</style>

</head>

<body>

<form id="phoneForm">

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

<input type="text" id="phone" name="phone">

<span id="phoneError" class="error">Please enter a valid phone number (e.g., (123) 456-7890 or 123-456-7890).</span>

<br><br>

<input type="submit" value="Submit">

</form>

<script>

const form = document.getElementById('phoneForm');

const phoneInput = document.getElementById('phone');

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

const phoneRegex = /^\(?\d{3}\)?[-\s]?\d{3}[-\s]?\d{4}$/;

form.addEventListener('submit', (event) => {

if (!phoneRegex.test(phoneInput.value)) {

phoneError.style.display = 'block';

event.preventDefault(); // Prevent form submission

} else {

phoneError.style.display = 'none';

}

});

</script>

</body>

</html>

**Exercise 14: Validate a Date of Birth**

**Problem**: Create a form to validate a date of birth in the format DD/MM/YYYY. Display an error message in red if the date is invalid.

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

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

<title>Date of Birth Validation</title>

<style>

body {

font-family: Arial, sans-serif;

padding: 20px;

}

form {

max-width: 300px;

margin: auto;

}

label {

display: block;

margin-bottom: 8px;

}

input[type="text"] {

width: 100%;

padding: 8px;

margin-bottom: 10px;

border: 1px solid #ccc;

border-radius: 4px;

}

.error {

color: red;

display: none;

font-size: 0.875em;

}

input[type="submit"] {

background-color: #4CAF50;

color: white;

padding: 10px 20px;

border: none;

border-radius: 4px;

cursor: pointer;

}

input[type="submit"]:hover {

background-color: #45a049;

}

</style>

</head>

<body>

<form id="dobForm">

<label for="dob">Date of Birth (DD/MM/YYYY):</label>

<input type="text" id="dob" name="dob">

<span id="dobError" class="error">Please enter a valid date in DD/MM/YYYY format.</span>

<br><br>

<input type="submit" value="Submit">

</form>

<script>

const form = document.getElementById('dobForm');

const dobInput = document.getElementById('dob');

const dobError = document.getElementById('dobError');

const dobRegex = /^\d{2}\/\d{2}\/\d{4}$/;

form.addEventListener('submit', (event) => {

if (!dobRegex.test(dobInput.value)) {

dobError.style.display = 'block';

event.preventDefault(); // Prevent form submission

} else {

dobError.style.display = 'none';

}

});

</script>

</body>

</html>

**Exercise 15: Validate a Credit Card Number**

**Problem**: Create a form to validate credit card numbers with spaces (e.g., 1234 5678 9012 3456). Display an error message in red if the credit card number is invalid.

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

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

<title>Credit Card Validation</title>

<style>

body {

font-family: Arial, sans-serif;

padding: 20px;

}

form {

max-width: 300px;

margin: auto;

}

label {

display: block;

margin-bottom: 8px;

}

input[type="text"] {

width: 100%;

padding: 8px;

margin-bottom: 10px;

border: 1px solid #ccc;

border-radius: 4px;

}

.error {

color: red;

display: none;

font-size: 0.875em;

}

input[type="submit"] {

background-color: #4CAF50;

color: white;

padding: 10px 20px;

border: none;

border-radius: 4px;

cursor: pointer;

}

input[type="submit"]:hover {

background-color: #45a049;

}

</style>

</head>

<body>

<form id="creditCardForm">

<label for="creditCard">Credit Card Number:</label>

<input type="text" id="creditCard" name="creditCard">

<span id="creditCardError" class="error">Please enter a valid credit card number (e.g., 1234 5678 9012 3456).</span>

<br><br>

<input type="submit" value="Submit">

</form>

<script>

const form = document.getElementById('creditCardForm');

const creditCardInput = document.getElementById('creditCard');

const creditCardError = document.getElementById('creditCardError');

const creditCardRegex = /^(\d{4}\s?){3}\d{4}$/;

form.addEventListener('submit', (event) => {

if (!creditCardRegex.test(creditCardInput.value)) {

creditCardError.style.display = 'block';

event.preventDefault(); // Prevent form submission

} else {

creditCardError.style.display = 'none';

}

});

</script>

</body>

</html>

**Exercise 16:**

**Form Validation Exercise: Product Insertion Form**

**Objective**: Create a product insertion form with validation using HTML, CSS, and JavaScript. Implement regular expressions to validate input fields and display error messages in red if the validation fails.

**Requirements**:

1. **HTML Form**:
   * Include fields for the following product details:
     + Product Name (2-50 characters, letters and numbers only)
     + Price (positive number, up to 2 decimal places)
     + Category (dropdown with options: Electronics, Clothing, Home, Toys)
     + Stock Quantity (positive integer)
     + Brand (2-30 characters, letters and numbers only)
     + Model (2-30 characters, letters and numbers only)
     + Weight (positive number, up to 2 decimal places)
     + Color (2-30 characters, letters only)
     + Size (2-20 characters, letters and numbers only)
     + Description (5-500 characters)
2. **CSS**:
   * Style the form to be large, with a width of at least 800px, centered on the page.
   * Apply padding, border, and shadow to the form container.
   * Ensure all input fields, select dropdowns, and the textarea are styled consistently.
   * Error messages should be displayed in red below each field if the input does not meet the validation criteria.
   * Style the submit button to be large and provide a hover effect.
3. **JavaScript**:
   * Implement form validation using regular expressions.
   * Display error messages in red if the input does not match the expected format.
   * Prevent form submission if any field is invalid.

**Bonus**:

* Implement additional validation logic to handle edge cases and improve user experience.
* Ensure that the error messages are cleared when the input becomes valid.

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

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

<title>Product Insertion Form</title>

<style>

body {

font-family: Arial, sans-serif;

padding: 40px;

display: flex;

justify-content: center;

align-items: center;

height: 100vh;

background-color: #f4f4f4;

}

form {

width: 800px;

padding: 30px;

border: 2px solid #ddd;

border-radius: 8px;

background-color: #fff;

box-shadow: 0 0 15px rgba(0,0,0,0.1);

}

label {

display: block;

margin-bottom: 10px;

font-size: 1.2em;

}

input[type="text"], input[type="number"], select, textarea {

width: calc(100% - 22px);

padding: 12px;

margin-bottom: 15px;

border: 2px solid #ddd;

border-radius: 4px;

font-size: 1em;

}

textarea {

height: 100px;

resize: vertical;

}

.error {

color: red;

display: none;

font-size: 1em;

margin-bottom: 15px;

}

input[type="submit"] {

background-color: #4CAF50;

color: white;

padding: 15px 30px;

border: none;

border-radius: 4px;

cursor: pointer;

font-size: 1.2em;

}

input[type="submit"]:hover {

background-color: #45a049;

}

</style>

</head>

<body>

<form id="productForm">

<label for="name">Product Name:</label>

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

<span id="nameError" class="error">Product name is required (2-50 characters, letters and numbers only).</span>

<label for="price">Price:</label>

<input type="number" id="price" name="price" step="0.01">

<span id="priceError" class="error">Price must be a positive number (e.g., 19.99).</span>

<label for="category">Category:</label>

<select id="category" name="category">

<option value="">Select a category</option>

<option value="electronics">Electronics</option>

<option value="clothing">Clothing</option>

<option value="home">Home</option>

<option value="toys">Toys</option>

</select>

<span id="categoryError" class="error">Please select a category.</span>

<label for="stock">Stock Quantity:</label>

<input type="number" id="stock" name="stock">

<span id="stockError" class="error">Stock quantity must be a positive integer.</span>

<label for="brand">Brand:</label>

<input type="text" id="brand" name="brand">

<span id="brandError" class="error">Brand is required (2-30 characters, letters and numbers only).</span>

<label for="model">Model:</label>

<input type="text" id="model" name="model">

<span id="modelError" class="error">Model is required (2-30 characters, letters and numbers only).</span>

<label for="weight">Weight (kg):</label>

<input type="number" id="weight" name="weight" step="0.01">

<span id="weightError" class="error">Weight must be a positive number (e.g., 1.5).</span>

<label for="color">Color:</label>

<input type="text" id="color" name="color">

<span id="colorError" class="error">Color is required (2-30 characters, letters only).</span>

<label for="size">Size:</label>

<input type="text" id="size" name="size">

<span id="sizeError" class="error">Size is required (2-20 characters, letters and numbers only).</span>

<label for="description">Description:</label>

<textarea id="description" name="description"></textarea>

<span id="descriptionError" class="error">Description is required (5-500 characters).</span>

<input type="submit" value="Submit">

</form>

<script>

const form = document.getElementById('productForm');

const nameInput = document.getElementById('name');

const priceInput = document.getElementById('price');

const categorySelect = document.getElementById('category');

const stockInput = document.getElementById('stock');

const brandInput = document.getElementById('brand');

const modelInput = document.getElementById('model');

const weightInput = document.getElementById('weight');

const colorInput = document.getElementById('color');

const sizeInput = document.getElementById('size');

const descriptionTextarea = document.getElementById('description');

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

const priceError = document.getElementById('priceError');

const categoryError = document.getElementById('categoryError');

const stockError = document.getElementById('stockError');

const brandError = document.getElementById('brandError');

const modelError = document.getElementById('modelError');

const weightError = document.getElementById('weightError');

const colorError = document.getElementById('colorError');

const sizeError = document.getElementById('sizeError');

const descriptionError = document.getElementById('descriptionError');

// Regular Expressions

const nameRegex = /^[a-zA-Z0-9\s]{2,50}$/;

const priceRegex = /^\d+(\.\d{1,2})?$/;

const brandModelRegex = /^[a-zA-Z0-9\s]{2,30}$/;

const colorRegex = /^[a-zA-Z\s]{2,30}$/;

const sizeRegex = /^[a-zA-Z0-9\s]{2,20}$/;

const descriptionRegex = /^.{5,500}$/;

form.addEventListener('submit', (event) => {

let isValid = true;

if (!nameRegex.test(nameInput.value.trim())) {

nameError.style.display = 'block';

isValid = false;

} else {

nameError.style.display = 'none';

}

if (!priceRegex.test(priceInput.value.trim()) || parseFloat(priceInput.value) <= 0) {

priceError.style.display = 'block';

isValid = false;

} else {

priceError.style.display = 'none';

}

if (categorySelect.value === '') {

categoryError.style.display = 'block';

isValid = false;

} else {

categoryError.style.display = 'none';

}

if (isNaN(stockInput.value) || parseInt(stockInput.value) <= 0) {

stockError.style.display = 'block';

isValid = false;

} else {

stockError.style.display = 'none';

}

if (!brandModelRegex.test(brandInput.value.trim())) {

brandError.style.display = 'block';

isValid = false;

} else {

brandError.style.display = 'none';

}

if (!brandModelRegex.test(modelInput.value.trim())) {

modelError.style.display = 'block';

isValid = false;

} else {

modelError.style.display = 'none';

}

if (!weightRegex.test(weightInput.value.trim()) || parseFloat(weightInput.value) <= 0) {

weightError.style.display = 'block';

isValid = false;

} else {

weightError.style.display = 'none';

}

if (!colorRegex.test(colorInput.value.trim())) {

colorError.style.display = 'block';

isValid = false;

} else {

colorError.style.display = 'none';

}

if (!sizeRegex.test(sizeInput.value.trim())) {

sizeError.style.display = 'block';

isValid = false;

} else {

sizeError.style.display = 'none';

}

if (!descriptionRegex.test(descriptionTextarea.value.trim())) {

descriptionError.style.display = 'block';

isValid = false;

} else {

descriptionError.style.display = 'none';

}

if (!isValid) {

event.preventDefault(); // Prevent form submission

}

});

</script>

</body>

</html>

**Explanation:**

* **Regular Expressions**:
  + **Product Name** (nameRegex): Validates that the name contains 2-50 characters, including letters, numbers, and spaces.
  + **Price** (priceRegex): Ensures the price is a positive number with up to two decimal places.
  + **Brand and Model** (brandModelRegex): Validates that brand and model contain 2-30 characters, including letters, numbers, and spaces.
  + **Color** (colorRegex): Ensures the color contains 2-30 characters and includes only letters and spaces.
  + **Size** (sizeRegex): Validates that size contains 2-20 characters, including letters, numbers, and spaces.
  + **Description** (descriptionRegex): Ensures the description contains 5-500 characters.
* **Validation**:
  + Checks each field against its corresponding regular expression.
  + Displays an error message in red if the field does not meet the criteria.