

Resource File: form-validation.js

1. Executive Summary

A supporting file used by the application for configuration or data storage.

2. Code Logic & Functionality

Contains static data or configuration parameters read by the application at runtime.

3. Key Concepts & Definitions

- **Static Asset:** A file that is not generated dynamically (e.g., images, text files).
- **Configuration:** Settings that determine the behavior of the software.

4. Location Details

Path: static\js\form-validation.js **Type:** .JS File

5. Source Code Preview (Snippet)

Running typical software analysis on this file:

```
/**
 * Professional Form Validation & UX Enhancements
 * Client-side validation with visual feedback
 */

// Real-time Email Validation
function validateEmail(email) {
    const regex = /^[a-zA-Z0-9._%+-]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,}$/;
    return regex.test(email);
}

// Real-time Phone Validation
function validatePhone(phone) {
    const regex = /^[0-9+\s-]{10,15}$/;
    return regex.test(phone);
}

// Password Strength Checker
function checkPasswordStrength(password) {
    let strength = 0;
    if (password.length >= 8) strength++;
    if (password.length >= 12) strength++;
    if (/[a-z]/.test(password)) strength++;
    if (/[A-Z]/.test(password)) strength++;
    if (/[0-9]/.test(password)) strength++;
```

```

    if (/^[^a-zA-Z0-9]/.test(password)) strength++;

    if (strength <= 2) return { level: 'weak', color: '#dc3545', text:
    if (strength <= 4) return { level: 'medium', color: '#ffc107', text:
    return { level: 'strong', color: '#28a745', text: 'Strong' };
}

// Add visual feedback to input fields
function addFieldFeedback(input, isValid, message = '') {
    const feedback = input.nextElementSibling;

    if (isValid) {
        input.classList.remove('is-invalid');
        input.classList.add('is-valid');
        if (feedback && feedback.classList.contains('invalid-feedback'))
            feedback.style.display = 'none';
    }
    else {
        input.classList.remove('is-valid');
        input.classList.add('is-invalid');
        if (feedback && feedback.classList.contains('invalid-feedback'))
            feedback.textContent = message;
            feedback.style.display = 'block';
        }
    }
}

// Initialize form validation on page load
document.addEventListener('DOMContentLoaded', function () {

    // Email fields validation
    const emailInputs = document.querySelectorAll('input[type="email"]')
    emailInputs.forEach(input => {
        input.addEventListener('blur', function () {
            if (this.value) {
                const isValid = validateEmail(this.value);
                addFieldFeedback(this, isValid, 'Please enter a valid e
            }
        });

        input.addEventListener('input', function () {
            if (this.classList.contains('is-invalid') && validateEmail(
                addFieldFeedback(this, true);
            }
        });
    });

    // Phone fields validation
    const phoneInputs = document.querySelectorAll('input[name="phone"]')
    phoneInputs.forEach(input => {
        input.addEventListener('blur', function () {
            if (this.value) {
                const isValid = validatePhone(this.value);

```

```

        addFieldFeedback(this, isValid, 'Please enter a valid p
    }
    });
});

// Password strength indicator
const passwordInputs = document.querySelectorAll('input[type="passw
passwordInputs.forEach(input => {
    // Create strength indicator
    const strengthDiv = document.createElement('div');
    strengthDiv.className = 'password-strength mt-2';
    strengthDiv.innerHTML = `
        <div class="d-flex justify-content-between align-items-cent
            <small class="text-muted">Password Strength:</small>
            <small class="strength-text fw-bold">--</small>
        </div>
        <div class="progress" style="height: 4px;">
            <div class="progress-bar" role="progressbar" style="wid
        </div>
    `;
    input.parentNode.insertBefore(strengthDiv, input.nextSibling);

    input.addEventListener('input', function () {
        const strength = checkPasswordStrength(this.value);
        const progressBar = strengthDiv.querySelector('.progress-ba
        const strengthText = strengthDiv.querySelector('.strength-t

        if (this.value.length > 0) {
            const widthMap = { weak: '33%', medium: '66%', strong:
            progressBar.style.width = widthMap[strength.level];
            progressBar.style.backgroundColor = strength.color;
            strengthText.textContent = strength.text;
            strengthText.style.color = strength.color;
            strengthDiv.style.display = 'block';
        } else {
            strengthDiv.style.display = 'none';
        }
    });
});

// Password confirmation matching
const confirmInputs = document.querySelectorAll('input[name="confir
confirmInputs.forEach(input => {
    input.addEventListener('input', function () {
        const password = document.querySelector('input[name="passwo
        if (this.value) {
            const isValid = this.value === password;
            addFieldFeedback(this, isValid, 'Passwords do not match
        }
    });
});

// Required field indicators

```

```

const requiredInputs = document.querySelectorAll('input[required],
requiredInputs.forEach(input => {
  // Add asterisk to labels
  const label = input.previousElementSibling;
  if (label && label.tagName === 'LABEL' && !label.querySelector(
    label.innerHTML += ' <span class="text-danger">*</span>';
  }

  // Validate on blur
  input.addEventListener('blur', function () {
    if (!this.value.trim()) {
      addFieldFeedback(this, false, 'This field is required')
    } else {
      if (!this.classList.contains('is-invalid')) {
        addFieldFeedback(this, true);
      }
    }
  });
});

```

... [Code Truncated for Documentation Readability - See Source File for