

# 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: 'Weak' };
if (strength <= 4) return { level: 'medium', color: '#ffc107', text: 'Medium' };
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 email');
            }
        });
        input.addEventListener('input', function () {
            if (this.classList.contains('is-invalid') && validateEmail(this.value))
                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 phone number');
            }
        });
    });
}

```

```

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

// Password strength indicator
const passwordInputs = document.querySelectorAll('input[type="password"]');
passwordInputs.forEach(input => {
    // Create strength indicator
    const strengthDiv = document.createElement('div');
    strengthDiv.className = 'password-strength mt-2';
    strengthDiv.innerHTML =
        `
```

```
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
```