HTML5, CSS, and JavaScript Best Practices Guide

Project Structure & Organization

Recommended Folder Structure with Descriptions

```
project-name/                    # Root project directory
├── index.html                   # Main entry point, homepage
│
├── assets/                      # Static assets (CSS, JS, images, fonts, media)
│   │
│   ├── css/                     # All stylesheets and CSS-related files
│   │   ├── main.css             # Main stylesheet, imports all other CSS
│   │   ├── reset.css            # CSS reset/normalize styles
│   │   ├── variables.css        # CSS custom properties (:root variables)
│   │   ├── base.css             # Base HTML element styles
│   │   ├── layout.css           # Grid systems, containers, layout patterns
│   │   │
│   │   ├── components/          # Component-specific styles (BEM methodology)
│   │   │   ├── buttons.css      # .btn, .btn--primary, .btn__icon
│   │   │   ├── cards.css        # .card, .card__header, .card--featured
│   │   │   ├── forms.css        # .form, .form__input, .form--inline
│   │   │   ├── navigation.css   # .nav, .nav__item, .nav--mobile
│   │   │   ├── modal.css        # .modal, .modal__content, .modal--large
│   │   │   └── header.css       # .site-header, .site-header__logo
│   │   │
│   │   └── utilities/           # Helper/utility classes
│   │       ├── spacing.css      # .mt-1, .p-2, .mx-auto (margin/padding)
│   │       ├── typography.css   # .text-lg, .font-bold, .text-center
│   │       ├── colors.css       # .text-primary, .bg-secondary
│   │       ├── display.css      # .hidden, .flex, .grid, .block
│   │       └── responsive.css   # .sm:hidden, .md:flex (media queries)
```

```
│   │
│   ├── js/                     # All JavaScript files
│   │   ├── main.js             # Main JS entry point, app initialization
│   │   ├── config.js           # App configuration, constants, settings
│   │   │
│   │   ├── modules/            # Feature-based modules (business logic)
│   │   │   ├── auth.js         # Authentication logic, login/logout
│   │   │   ├── api.js          # API calls, data fetching
│   │   │   ├── router.js       # Client-side routing (if SPA)
│   │   │   ├── search.js       # Search functionality
│   │   │   ├── cart.js         # Shopping cart logic
│   │   │   └── analytics.js    # Analytics tracking, event logging
│   │   │
│   │   └── utils/              # Helper functions and utilities
│   │       ├── dom.js          # DOM manipulation helpers
│   │       ├── validation.js   # Form validation functions
│   │       ├── formatters.js   # Date, currency, text formatting
│   │       ├── helpers.js      # General utility functions
│   │       ├── storage.js      # Local storage wrapper functions
│   │       └── constants.js    # App-wide constants, enums
│   │
│   ├── images/                 # All image assets organized by type
│   │   ├── icons/              # SVG icons, icon fonts, small graphics
│   │   │   ├── arrow.svg       # Individual SVG icons
│   │   │   ├── user.svg        # User interface icons
│   │   │   └── social-icons/   # Social media icons
│   │   │
│   │   ├── photos/             # Photography, hero images, backgrounds
│   │   │   ├── hero-bg.jpg     # Large background images
│   │   │   ├── team/           # Team member photos
│   │   │   └── products/       # Product photography
│   │   │
│   │   └── graphics/           # Illustrations, logos, design elements
│   │       ├── logo.svg        # Company logo (preferably SVG)
│   │       ├── illustrations/  # Custom illustrations
│   │       └── patterns/       # Background patterns, textures
│   │
│   ├── fonts/                  # Custom font files
│   │   ├── inter/              # Font family folders
│   │   │   ├── inter-regular.woff2
│   │   │   ├── inter-bold.woff2
```

```
│   │   │   └── inter-italic.woff2
│   │   └── roboto/
│   │       ├── roboto-regular.woff2
│   │       └── roboto-bold.woff2
│   │
│   └── media/                    # Audio, video, and other media files
│       ├── videos/               # Video files (.mp4, .webm)
│       ├── audio/                # Audio files (.mp3, .wav)
│       └── documents/            # PDFs, documents for download
│
├── pages/                        # Additional HTML pages
│   ├── about.html                # About us page
│   ├── contact.html              # Contact page
│   ├── products.html             # Products listing page
│   ├── blog/                     # Blog-related pages
│   │   ├── index.html            # Blog homepage
│   │   └── post.html             # Individual blog post template
│   └── auth/                     # Authentication pages
│       ├── login.html            # Login page
│       ├── register.html         # Registration page
│       └── forgot-password.html  # Password reset page
│
├── components/                   # Reusable HTML components/partials
│   ├── header.html               # Site header component
│   ├── footer.html               # Site footer component
│   ├── navigation.html           # Navigation menu component
│   ├── sidebar.html              # Sidebar component
│   ├── modal.html                # Modal dialog template
│   ├── product-card.html         # Product card component
│   └── forms/                    # Form components
│       ├── contact-form.html     # Contact form component
│       ├── search-form.html      # Search form component
│       └── newsletter.html       # Newsletter signup form
│
└── docs/                         # Project documentation
    ├── README.md                 # Project overview, setup instructions
    ├── CONTRIBUTING.md           # Contribution guidelines
    ├── CHANGELOG.md              # Version history and changes
    ├── style-guide.md            # Code style guide and conventions
    ├── api-documentation.md      # API documentation (if applicable)
    ├── deployment.md             # Deployment instructions
```

```
└── design-system/          # Design system documentation
    ├── colors.md           # Color palette and usage
    ├── typography.md       # Font usage and hierarchy
    └── components.md       # Component specifications
```

File Naming Conventions

- Use **kebab-case** for files: `header-navigation.css`, `user-profile.js`
- Be descriptive but concise: `product-card.css` not `pc.css`
- Use consistent prefixes: `page-`, `component-`, `util-`

HTML5 Best Practices

Semantic HTML Structure

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Page Title - Site Name</title>
</head>
<body>
  <header class="site-header">
    <nav class="main-navigation" aria-label="Main navigation">
      <!-- Navigation content -->
    </nav>
  </header>

  <main class="main-content">
    <section class="hero-section">
      <!-- Hero content -->
    </section>

    <section class="features-section">
      <!-- Features content -->
    </section>
  </main>

  <aside class="sidebar">
```

```
    <!-- Sidebar content -->
  </aside>

  <footer class="site-footer">
    <!-- Footer content -->
  </footer>
</body>
</html>
```

HTML Naming Conventions

Class Names (BEM Methodology)

- **Block**: `.card`
- **Element**: `.card__title`, `.card__content`
- **Modifier**: `.card--featured`, `.card__title--large`

```
<article class="product-card product-card--featured">
  <h2 class="product-card__title">Product Name</h2>
  <p class="product-card__description">Product description</p>
  <button class="product-card__button product-card__button--primary">
    Buy Now
  </button>
</article>
```

ID Names

- Use **kebab-case**: `#main-navigation`, `#user-profile-form`
- Be specific and unique: `#contact-form-email` not `#email`
- Reserve for JavaScript hooks or form labels

Data Attributes

- Use **kebab-case**: `data-user-id`, `data-scroll-target`
- Prefix with purpose: `data-js-` for JavaScript hooks, `data-test-` for testing

```
<button data-js-modal-trigger data-modal-target="signup-modal">
  Sign Up
</button>
```

CSS Best Practices

CSS Reset/Normalize

```
/* Modern CSS Reset */
*, *::before, *::after {
  box-sizing: border-box;
}

* {
  margin: 0;
  padding: 0;
}

html {
  -webkit-text-size-adjust: 100%;
}

body {
  line-height: 1.5;
  -webkit-font-smoothing: antialiased;
}

img, picture, video, canvas, svg {
  display: block;
  max-width: 100%;
}

input, button, textarea, select {
  font: inherit;
}
```

CSS Custom Properties (:root)

```
:root {
  /* Colors */
  --color-primary: #3b82f6;
  --color-primary-dark: #1e40af;
  --color-secondary: #64748b;
  --color-success: #10b981;
```

```
--color-error: #ef4444;
--color-warning: #f59e0b;

/* Neutral Colors */
--color-white: #ffffff;
--color-gray-50: #f9fafb;
--color-gray-100: #f3f4f6;
--color-gray-900: #111827;
--color-black: #000000;

/* Typography */
--font-family-primary: 'Inter', system-ui, sans-serif;
--font-family-secondary: 'Merriweather', serif;
--font-family-mono: 'Fira Code', monospace;

/* Font Sizes */
--font-size-xs: 0.75rem;
--font-size-sm: 0.875rem;
--font-size-base: 1rem;
--font-size-lg: 1.125rem;
--font-size-xl: 1.25rem;
--font-size-2xl: 1.5rem;
--font-size-3xl: 1.875rem;

/* Spacing */
--spacing-xs: 0.25rem;
--spacing-sm: 0.5rem;
--spacing-md: 1rem;
--spacing-lg: 1.5rem;
--spacing-xl: 2rem;
--spacing-2xl: 3rem;

/* Borders */
--border-radius-sm: 0.25rem;
--border-radius-md: 0.375rem;
--border-radius-lg: 0.5rem;
--border-radius-full: 9999px;

/* Shadows */
--shadow-sm: 0 1px 2px rgba(0, 0, 0, 0.05);
--shadow-md: 0 4px 6px rgba(0, 0, 0, 0.1);
--shadow-lg: 0 10px 15px rgba(0, 0, 0, 0.1);

/* Breakpoints */
```

```
  --breakpoint-sm: 640px;
  --breakpoint-md: 768px;
  --breakpoint-lg: 1024px;
  --breakpoint-xl: 1280px;
}
```

HTML Element Styling

```
html {
  font-size: 100%; /* 16px base */
  scroll-behavior: smooth;
}

body {
  font-family: var(--font-family-primary);
  font-size: var(--font-size-base);
  line-height: 1.6;
  color: var(--color-gray-900);
  background-color: var(--color-white);
}

/* Headings */
h1, h2, h3, h4, h5, h6 {
  font-weight: 600;
  line-height: 1.2;
  margin-bottom: var(--spacing-md);
}

h1 { font-size: var(--font-size-3xl); }
h2 { font-size: var(--font-size-2xl); }
h3 { font-size: var(--font-size-xl); }

/* Links */
a {
  color: var(--color-primary);
  text-decoration: none;
  transition: color 0.2s ease;
}

a:hover {
  color: var(--color-primary-dark);
  text-decoration: underline;
```

```
}

/* Lists */
ul, ol {
  margin-bottom: var(--spacing-md);
  padding-left: var(--spacing-lg);
}

/* Images */
img {
  height: auto;
  border-radius: var(--border-radius-md);
}
```

CSS Organization & Architecture

1. ITCSS (Inverted Triangle CSS) Layer Order

```
/* 1. Settings - Variables and configuration */
@import 'settings/variables';
@import 'settings/breakpoints';

/* 2. Tools - Mixins and functions */
@import 'tools/mixins';
@import 'tools/functions';

/* 3. Generic - Reset and normalize */
@import 'generic/reset';
@import 'generic/normalize';

/* 4. Elements - Base HTML elements */
@import 'elements/headings';
@import 'elements/links';
@import 'elements/forms';

/* 5. Objects - Layout patterns */
@import 'objects/container';
@import 'objects/grid';
@import 'objects/media';

/* 6. Components - UI components */
@import 'components/buttons';
@import 'components/cards';
```

```
@import 'components/navigation';

/* 7. Utilities - Helper classes */
@import 'utilities/spacing';
@import 'utilities/typography';
@import 'utilities/display';
```

2. Component-Based CSS

```
/* Button Component */
.btn {
  display: inline-flex;
  align-items: center;
  justify-content: center;
  padding: var(--spacing-sm) var(--spacing-md);
  border: 2px solid transparent;
  border-radius: var(--border-radius-md);
  font-weight: 600;
  text-decoration: none;
  transition: all 0.2s ease;
  cursor: pointer;
}

.btn--primary {
  background-color: var(--color-primary);
  color: var(--color-white);
}

.btn--primary:hover {
  background-color: var(--color-primary-dark);
  transform: translateY(-1px);
  box-shadow: var(--shadow-md);
}

.btn--secondary {
  background-color: transparent;
  color: var(--color-primary);
  border-color: var(--color-primary);
}

.btn--large {
  padding: var(--spacing-md) var(--spacing-lg);
```

```
  font-size: var(--font-size-lg);
}
```

Responsive Design Best Practices

```
/* Mobile-first approach */
.container {
  width: 100%;
  max-width: 1200px;
  margin: 0 auto;
  padding: 0 var(--spacing-md);
}

/* Tablet */
@media (min-width: 768px) {
  .container {
    padding: 0 var(--spacing-lg);
  }

  .grid {
    display: grid;
    grid-template-columns: repeat(2, 1fr);
    gap: var(--spacing-lg);
  }
}

/* Desktop */
@media (min-width: 1024px) {
  .grid {
    grid-template-columns: repeat(3, 1fr);
    gap: var(--spacing-xl);
  }
}
```

JavaScript Best Practices

Modern JavaScript Best Practices

Variable Declarations - Use const & let, Avoid var

```
//    Avoid var - function scoped, can be redeclared
var userName = 'john'; // Don't use this

//    Use const for values that won't be reassigned
const API_URL = 'https://api.example.com';
const userConfig = {
  theme: 'dark',
  notifications: true
};

//    Use let for values that will be reassigned
let currentUser = null;
let isLoading = false;
let retryCount = 0;

//    const with objects/arrays (contents can still change)
const users = [];
users.push({ name: 'John', age: 30 }); // This is fine
userConfig.theme = 'light'; // This is fine
```

Arrow Functions for Callbacks

```
//    Use arrow functions for short callbacks
const numbers = [1, 2, 3, 4, 5];

// Map with arrow function
const doubled = numbers.map(num => num * 2);
const squares = numbers.map(num => num ** 2);

// Filter with arrow function
const evenNumbers = numbers.filter(num => num % 2 === 0);
const largeNumbers = numbers.filter(num => num > 3);

// Event listeners with arrow functions
button.addEventListener('click', (event) => {
  event.preventDefault();
  handleButtonClick();
});

// Async arrow functions
const fetchUserData = async (userId) => {
  try {
```

```
    const response = await fetch(`/api/users/${userId}`);
    return await response.json();
  } catch (error) {
    console.error('Error fetching user:', error);
    throw error;
  }
};

//    Use regular functions for methods that need 'this' context
const userService = {
  users: [],

  addUser(userData) {
    this.users.push(userData); // 'this' refers to userService
  },

  // Or use arrow functions with explicit context
  removeUser: (userId) => {
    userService.users = userService.users.filter(user => user.id !== user
  }
};
```

Modern Iteration - Prefer for...of, .map(), .forEach() over for loops

```
const users = [
  { id: 1, name: 'Alice', active: true },
  { id: 2, name: 'Bob', active: false },
  { id: 3, name: 'Charlie', active: true }
];

//    Avoid traditional for loops when possible
for (let i = 0; i < users.length; i++) {
  console.log(users[i].name);
}

//    Use for...of for simple iteration
for (const user of users) {
  console.log(user.name);
}

//    Use forEach for side effects (no return value needed)
users.forEach(user => {
  console.log(`User: ${user.name}, Active: ${user.active}`);
```

```
});

//    Use map when you need to transform data
const userNames = users.map(user => user.name);
const activeUsers = users.map(user => ({
  ...user,
  status: user.active ? 'online' : 'offline'
}));

//    Use filter for conditional selection
const activeUsersList = users.filter(user => user.active);

//    Use reduce for aggregation
const totalActiveUsers = users.reduce((count, user) => {
  return user.active ? count + 1 : count;
}, 0);

//    Chain array methods for complex operations
const activeUserNames = users
  .filter(user => user.active)
  .map(user => user.name)
  .sort();

//    Use for...in for object properties
const userSettings = {
  theme: 'dark',
  language: 'en',
  notifications: true
};

for (const setting in userSettings) {
  console.log(`${setting}: ${userSettings[setting]}`);
}

//    Or use Object methods for better control
Object.entries(userSettings).forEach(([key, value]) => {
  console.log(`${key}: ${value}`);
});
```

Always Handle Errors with try/catch

```
//    Async function error handling
const handleApiRequest = async (endpoint, data) => {
```

```
  try {
    const response = await fetch(endpoint, {
      method: 'POST',
      headers: {
        'Content-Type': 'application/json',
        'Authorization': `Bearer ${getAuthToken()}`
      },
      body: JSON.stringify(data)
    });

    if (!response.ok) {
      throw new Error(`HTTP error! status: ${response.status}`);
    }

    const result = await response.json();
    return result;

  } catch (error) {
    // Handle different error types
    if (error.name === 'TypeError') {
      console.error('Network error:', error.message);
      throw new Error('Network connection failed');
    } else if (error.message.includes('HTTP error')) {
      console.error('API error:', error.message);
      throw new Error('Server request failed');
    } else {
      console.error('Unexpected error:', error);
      throw new Error('An unexpected error occurred');
    }
  }
};

//    Promise chain error handling
const loadUserProfile = (userId) => {
  return fetchUserData(userId)
    .then(userData => {
      if (!userData) {
        throw new Error('User not found');
      }
      return processUserData(userData);
    })
    .then(processedData => {
      renderUserProfile(processedData);
      return processedData;
```

```
    })
    .catch(error => {
      console.error('Failed to load user profile:', error);
      showErrorMessage('Unable to load user profile');
      throw error; // Re-throw if needed by calling code
    });
};

//    Multiple async operations with error handling
const initializeUserDashboard = async (userId) => {
  try {
    // Use Promise.allSettled for independent operations
    const results = await Promise.allSettled([
      fetchUserData(userId),
      fetchUserPreferences(userId),
      fetchUserNotifications(userId)
    ]);

    const [userResult, prefsResult, notificationsResult] = results;

    // Handle each result individually
    if (userResult.status === 'fulfilled') {
      renderUserInfo(userResult.value);
    } else {
      console.error('Failed to load user data:', userResult.reason);
      showPartialError('User information unavailable');
    }

    if (prefsResult.status === 'fulfilled') {
      applyUserPreferences(prefsResult.value);
    } else {
      console.error('Failed to load preferences:', prefsResult.reason);
      applyDefaultPreferences();
    }

    if (notificationsResult.status === 'fulfilled') {
      displayNotifications(notificationsResult.value);
    } else {
      console.error('Failed to load notifications:', notificationsResult.
    }

  } catch (error) {
    console.error('Critical error initializing dashboard:', error);
    showCriticalError('Unable to load dashboard');
```

```
  }
};

//    Custom error classes for better error handling
class ValidationError extends Error {
  constructor(message, field) {
    super(message);
    this.name = 'ValidationError';
    this.field = field;
  }
}

class NetworkError extends Error {
  constructor(message, status) {
    super(message);
    this.name = 'NetworkError';
    this.status = status;
  }
}

const validateAndSubmitForm = async (formData) => {
  try {
    // Validation
    if (!formData.email) {
      throw new ValidationError('Email is required', 'email');
    }
    if (!isValidEmail(formData.email)) {
      throw new ValidationError('Invalid email format', 'email');
    }

    // Submission
    const response = await submitForm(formData);
    return response;

  } catch (error) {
    if (error instanceof ValidationError) {
      highlightField(error.field);
      showFieldError(error.field, error.message);
    } else if (error instanceof NetworkError) {
      showNetworkError(`Server error: ${error.status}`);
    } else {
      showGenericError('An unexpected error occurred');
    }
    throw error;
```

```
  }
};
```

ES Modules - Avoid Global Scope Pollution

```
// userService.js - Export specific functions
export const userService = {
  async getUser(id) {
    const response = await fetch(`/api/users/${id}`);
    return response.json();
  },

  async updateUser(id, userData) {
    const response = await fetch(`/api/users/${id}`, {
      method: 'PUT',
      headers: { 'Content-Type': 'application/json' },
      body: JSON.stringify(userData)
    });
    return response.json();
  },

  async deleteUser(id) {
    await fetch(`/api/users/${id}`, { method: 'DELETE' });
  }
};

// utils.js - Export utility functions
export const formatDate = (date) => {
  return new Intl.DateTimeFormat('en-US', {
    year: 'numeric',
    month: 'long',
    day: 'numeric'
  }).format(new Date(date));
};

export const debounce = (func, wait) => {
  let timeout;
  return function executedFunction(...args) {
    const later = () => {
      clearTimeout(timeout);
      func(...args);
    };
    clearTimeout(timeout);
```

```
    timeout = setTimeout(later, wait);
  };
};

export const throttle = (func, limit) => {
  let inThrottle;
  return function() {
    const args = arguments;
    const context = this;
    if (!inThrottle) {
      func.apply(context, args);
      inThrottle = true;
      setTimeout(() => inThrottle = false, limit);
    }
  };
};

//    constants.js - Export constants
export const API_ENDPOINTS = {
  USERS: '/api/users',
  POSTS: '/api/posts',
  COMMENTS: '/api/comments'
};

export const HTTP_STATUS = {
  OK: 200,
  CREATED: 201,
  BAD_REQUEST: 400,
  UNAUTHORIZED: 401,
  NOT_FOUND: 404,
  SERVER_ERROR: 500
};

//    main.js - Import and use modules
import { userService } from './userService.js';
import { formatDate, debounce } from './utils.js';
import { API_ENDPOINTS, HTTP_STATUS } from './constants.js';

//    Named imports for specific functions
import {
  validateEmail,
  validatePassword,
  sanitizeInput
} from './validation.js';
```

```
//    Default export example
// components/UserCard.js
const UserCard = (userData) => {
  return {
    render() {
      const cardElement = document.createElement('div');
      cardElement.className = 'user-card';
      cardElement.innerHTML = `
        <h3>${userData.name}</h3>
        <p>${userData.email}</p>
        <p>Joined: ${formatDate(userData.joinDate)}</p>
      `;
      return cardElement;
    },

    update(newData) {
      Object.assign(userData, newData);
      this.render();
    }
  };
};

export default UserCard;

//    Import default export
import UserCard from './components/UserCard.js';

//    Dynamic imports for code splitting
const loadAdvancedFeatures = async () => {
  try {
    const { AdvancedChart } = await import('./advanced-features.js');
    return new AdvancedChart();
  } catch (error) {
    console.error('Failed to load advanced features:', error);
    return null;
  }
};

//    Module initialization pattern
const initializeApp = async () => {
  try {
    // Load core modules
    const currentUser = await userService.getCurrentUser();
```

```
    // Initialize components
    const userCard = new UserCard(currentUser);
    document.getElementById('user-profile').appendChild(userCard.render()

    // Setup event listeners with debouncing
    const searchInput = document.getElementById('search');
    const debouncedSearch = debounce((query) => {
      performSearch(query);
    }, 300);

    searchInput.addEventListener('input', (e) => {
      debouncedSearch(e.target.value);
    });

  } catch (error) {
    console.error('Failed to initialize app:', error);
    showErrorMessage('Application failed to load');
  }
};

// Start the app
document.addEventListener('DOMContentLoaded', initializeApp);
```

File Organization

```
// main.js - Entry point
import { initializeApp } from './modules/app.js';
import { setupNavigation } from './components/navigation.js';
import { loadUtilities } from './utils/helpers.js';

document.addEventListener('DOMContentLoaded', () => {
  initializeApp();
  setupNavigation();
});
```

Naming Conventions

Variables and Functions

```
// Use camelCase
const userName = 'john_doe';
const userAge = 25;
const isLoggedIn = true;

// Use descriptive names
const fetchUserData = async (userId) => {
  // Function implementation
};

const calculateTotalPrice = (items, taxRate) => {
  // Function implementation
};

// Constants in UPPER_SNAKE_CASE
const API_BASE_URL = 'https://api.example.com';
const MAX_RETRY_ATTEMPTS = 3;
```

Classes and Constructors

```
// PascalCase for classes
class UserProfile {
  constructor(userData) {
    this.firstName = userData.firstName;
    this.lastName = userData.lastName;
    this.email = userData.email;
  }

  getFullName() {
    return `${this.firstName} ${this.lastName}`;
  }
}

// Factory functions
const createUserCard = (userData) => {
  return {
    render() {
      // Render logic
    },
    update(newData) {
      // Update logic
    }
```

```
  };
};
```

Modern JavaScript Patterns

Module Pattern

```
// userService.js
export const userService = {
  async getUser(id) {
    try {
      const response = await fetch(`/api/users/${id}`);
      if (!response.ok) {
        throw new Error(`HTTP error! status: ${response.status}`);
      }
      return await response.json();
    } catch (error) {
      console.error('Error fetching user:', error);
      throw error;
    }
  },

  async updateUser(id, userData) {
    // Update implementation
  }
};
```

DOM Manipulation Best Practices

```
// Use modern DOM methods
const elements = {
  navigation: document.querySelector('[data-js="navigation"]'),
  toggleButton: document.querySelector('[data-js="nav-toggle"]'),
  menuItems: document.querySelectorAll('[data-js="nav-item"]')
};

// Event delegation
document.addEventListener('click', (event) => {
  const trigger = event.target.closest('[data-js="modal-trigger"]');
  if (trigger) {
    const modalId = trigger.dataset.modalTarget;
```

```
    openModal(modalId);
  }
});

// Clean event listeners
const handleResize = () => {
  // Resize logic
};

window.addEventListener('resize', handleResize);

// Clean up when needed
const cleanup = () => {
  window.removeEventListener('resize', handleResize);
};
```

Error Handling

```
// Async/await with proper error handling
const loadUserData = async (userId) => {
  try {
    const user = await userService.getUser(userId);
    renderUserProfile(user);
  } catch (error) {
    if (error.name === 'AbortError') {
      console.log('Request was cancelled');
    } else {
      showErrorMessage('Failed to load user data');
      console.error('User data error:', error);
    }
  }
};

// Global error handling
window.addEventListener('error', (event) => {
  console.error('Global error:', event.error);
  // Log to error tracking service
});

window.addEventListener('unhandledrejection', (event) => {
  console.error('Unhandled promise rejection:', event.reason);
  event.preventDefault();
});
```

Performance & Optimization

CSS Performance

```
/* Avoid expensive properties */
.efficient-animation {
  /* Use transform instead of changing layout properties */
  transform: translateX(0);
  transition: transform 0.3s ease;
}

.efficient-animation:hover {
  transform: translateX(10px);
}

/* Use will-change sparingly */
.animated-element {
  will-change: transform;
}

/* Remove will-change after animation */
.animation-complete {
  will-change: auto;
}
```

JavaScript Performance

```
// Debounce expensive operations
const debounce = (func, wait) => {
  let timeout;
  return function executedFunction(...args) {
    const later = () => {
      clearTimeout(timeout);
      func(...args);
    };
    clearTimeout(timeout);
    timeout = setTimeout(later, wait);
  };
};
```

```
const handleSearch = debounce((query) => {
  // Expensive search operation
}, 300);

// Use DocumentFragment for multiple DOM insertions
const addMultipleItems = (items) => {
  const fragment = document.createDocumentFragment();

  items.forEach(item => {
    const element = document.createElement('div');
    element.textContent = item.name;
    fragment.appendChild(element);
  });

  container.appendChild(fragment);
};
```

Code Quality & Maintainability

Comments and Documentation

```
/**
 * Calculates the total price including tax
 * @param {number} basePrice - The base price before tax
 * @param {number} taxRate - The tax rate as a decimal (e.g., 0.08 for 8%
 * @returns {number} The total price including tax
 */
const calculateTotalPrice = (basePrice, taxRate) => {
  if (typeof basePrice !== 'number' || typeof taxRate !== 'number') {
    throw new Error('Both parameters must be numbers');
  }

  return basePrice * (1 + taxRate);
};
```

Testing Considerations

```
// Write testable code
export const mathUtils = {
  add: (a, b) => a + b,
  multiply: (a, b) => a * b,
  calculatePercentage: (value, percentage) => (value * percentage) / 100
};

// Separate concerns
export const domUtils = {
  createElement: (tag, className, textContent) => {
    const element = document.createElement(tag);
    if (className) element.className = className;
    if (textContent) element.textContent = textContent;
    return element;
  }
};
```

Accessibility Best Practices

HTML Accessibility

```
<!-- Use semantic HTML -->
<button class="btn btn--primary" aria-describedby="btn-help">
  Submit Form
</button>
<div id="btn-help" class="sr-only">
  This will submit your form data
</div>

<!-- Form accessibility -->
<form>
  <div class="form-group">
    <label for="email" class="form-label">
      Email Address
      <span class="required" aria-label="required">*</span>
    </label>
    <input
      type="email"
      id="email"
      name="email"
```

```
      class="form-input"
      aria-describedby="email-error"
      aria-required="true"
    >
    <div id="email-error" class="form-error" role="alert">
      <!-- Error message will appear here -->
    </div>
  </div>
</form>
```

CSS Accessibility

```
/* Focus styles */
*:focus {
  outline: 2px solid var(--color-primary);
  outline-offset: 2px;
}

/* Skip to content link */
.skip-link {
  position: absolute;
  top: -40px;
  left: 6px;
  background: var(--color-primary);
  color: var(--color-white);
  padding: 8px;
  text-decoration: none;
  z-index: 1000;
}

.skip-link:focus {
  top: 6px;
}

/* Screen reader only content */
.sr-only {
  position: absolute;
  width: 1px;
  height: 1px;
  padding: 0;
  margin: -1px;
  overflow: hidden;
```

```
    clip: rect(0, 0, 0, 0);
    white-space: nowrap;
    border: 0;
  }
```

This guide provides a solid foundation for writing clean, maintainable, and performant HTML5, CSS, and JavaScript code. Remember to adapt these practices to your specific project needs and team preferences.