CSS Best Practices & Advanced Techniques

Session 9 Overview

This session focuses on professional CSS development practices, organization techniques, and advanced features to create maintainable and scalable stylesheets.

Table of Contents

CSS Organization

BEM Methodology

- **Block**: Standalone component (e.g., `.card`, `.button`)
- **Element**: Part of a block (e.g., `.card__title`, `.card__image`)
- **Modifier**: Variation of a block or element (e.g., `.button--primary`, `.card--featured`)

File Structure

```
styles/
├── base/           # Base styles (reset, typography, variables)
├── components/     # Reusable UI components
├── layout/         # Layout-specific styles
├── pages/          # Page-specific styles
├── themes/         # Theme variations
├── utils/          # Helper classes and mixins
└── main.css        # Main stylesheet that imports all others
```

Commenting and Documentation

```
/**
 * Section: Main Navigation
 * Description: Styles for the main navigation menu
 * Dependencies: _variables.scss, _mixins.scss
 */

/* Component: Button */
```

```css
.button {
  /* Base styles */
}

/* Modifier: Primary button */
.button--primary {
  /* Override styles */
}
```

CSS Variables

Defining Variables

```css
:root {
  /* Colors */
  --color-primary: #3498db;
  --color-secondary: #2ecc71;

  /* Typography */
  --font-main: 'Roboto', sans-serif;
  --font-size-base: 16px;

  /* Spacing */
  --spacing-unit: 1rem;
  --spacing-sm: calc(var(--spacing-unit) * 0.5);
  --spacing-md: var(--spacing-unit);
  --spacing-lg: calc(var(--spacing-unit) * 2);
}
```

Using Variables

```css
.button {
  background-color: var(--color-primary);
  font-family: var(--font-main);
  padding: var(--spacing-sm) var(--spacing-md);
}
```

Theming with CSS Variables

```css
/* Light theme (default) */
:root {
  --bg-color: #ffffff;
  --text-color: #333333;
}

/* Dark theme */
```

```css
[data-theme="dark"] {
  --bg-color: #1a1a1a;
  --text-color: #f5f5f5;
}

/* Apply theme */
body {
  background-color: var(--bg-color);
  color: var(--text-color);
  transition: background-color 0.3s, color 0.3s;
}
```

CSS Frameworks

Popular CSS Frameworks

1. **Bootstrap**
    - Comprehensive component library
    - Grid system
    - Responsive utilities
    - JavaScript plugins
2. **Tailwind CSS**
    - Utility-first approach
    - Highly customizable
    - No default theme
    - Just-in-Time compiler
3. **Bulma**
    - Flexbox-based
    - Modern and clean
    - No JavaScript dependencies

When to Use a Framework

- Rapid prototyping
- Team consistency
- Complex component needs
- Limited design resources

When to Avoid a Framework

- Highly customized designs
- Small projects with minimal styling
- Performance-critical applications
- Need for complete control

Performance Optimization

CSS Optimization Techniques

- **Minification**: Remove whitespace and comments
- **Critical CSS**: Inline above-the-fold styles
- **Code Splitting**: Load only necessary CSS
- **Purge Unused CSS**: Remove unused selectors

Tools

- **PostCSS**: Transform CSS with JavaScript
- **PurgeCSS**: Remove unused CSS
- **CSSNano**: CSS minifier
- **UnCSS**: Remove unused styles

Cross-Browser Compatibility

Common Issues and Fixes

1. **Vendor Prefixes**

   ```css
   .element {
     -webkit-border-radius: 5px;
     -moz-border-radius: 5px;
     border-radius: 5px;
   }
   ```

   Better: Use Autoprefixer

2. **Feature Detection**

   ```css
   @supports (display: grid) {
     .container {
       display: grid;
     }
   }
   ```

Accessibility

Best Practices

1. **Color Contrast**
   - Minimum 4.5:1 for normal text
   - 3:1 for large text (18pt+ or 14pt+bold)
   - Tools: WebAIM Contrast Checker

2. **Focus States**

   ```css
   a:focus, button:focus {
     outline: 3px solid #4d90fe;
     outline-offset: 2px;
   }
   ```

3. **Reduced Motion**

   ```css
   @media (prefers-reduced-motion: reduce) {
     * {
       animation-duration: 0.01ms !important;
       transition-duration: 0.01ms !important;
     }
   }
   ```

Homework

Project: Create a Design System

1. **Setup**
   - Create a new project with organized file structure
   - Set up a CSS preprocessor (Sass/LESS) if desired
2. **Design Tokens**
   - Define colors, typography, and spacing variables
   - Create a visual style guide
3. **Components**
   - Build 5+ reusable components using BEM
   - Include variations and states
4. **Documentation**
   - Document your design system
   - Include usage guidelines
   - Add code examples

Deliverables

1. GitHub repository with:
   - Organized CSS/Sass/LESS code
   - HTML examples
   - README with setup instructions

2. Live demo (GitHub Pages, Netlify, Vercel, etc.)

Resources

Documentation

- MDN Web Docs - CSS
- CSS Tricks
- BEM Methodology
- CSS Guidelines

Tools

- Autoprefixer
- PurgeCSS
- PostCSS
- Can I Use

Frameworks

- Bootstrap
- Tailwind CSS
- Bulma

Accessibility

- WebAIM
- A11Y Project
- WAI-ARIA Authoring Practices

Next Session: Advanced CSS Techniques and Final Project