HTML & CSS Cheat Sheet

A comprehensive reference guide for web development fundamentals

---

Getting Started

HTML (HyperText Markup Language) structures web content, while CSS (Cascading Style Sheets) controls presentation and layout. Together, they form the foundation of web development.

> 💡 **Quick Note:** HTML provides the structure (skeleton), CSS provides the styling (skin). Always write HTML first, then add CSS.

Basic HTML Document Structure

```html
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>My Web Page</title>
  <link rel="stylesheet" href="styles.css">
</head>
<body>
  <h1>Hello World!</h1>
  <p>This is my first web page.</p>
</body>
</html>
```

> 📄 **Document Tip:** Always include `<!DOCTYPE html>` at the top - it tells browsers to use modern HTML5 standards.

HTML Fundamentals

Headings and Text

```html
```

```
<!-- Headings (h1 is largest, h6 is smallest) -->
<h1>Main Title</h1>
<h2>Section Title</h2>
<h3>Subsection Title</h3>

<!-- Paragraphs and text formatting -->
<p>This is a paragraph with <strong>bold text</strong> and <em>italic text</em>.</p>
<p>You can also use <b>bold</b> and <i>italic</i> tags.</p>
<br> <!-- Line break -->
<hr> <!-- Horizontal rule -->
```

> 🏷️ **Heading Rule:** Use only one `<h1>` per page for SEO. Think of headings like an outline structure.

Lists

```
html

<!-- Unordered list (bullets) -->
<ul>
  <li>First item</li>
  <li>Second item</li>
  <li>Third item</li>
</ul>

<!-- Ordered list (numbers) -->
<ol>
  <li>Step one</li>
  <li>Step two</li>
  <li>Step three</li>
</ol>

<!-- Definition list -->
<dl>
  <dt>HTML</dt>
  <dd>HyperText Markup Language</dd>
  <dt>CSS</dt>
  <dd>Cascading Style Sheets</dd>
</dl>
```

> 📋 **List Tip:** You can nest lists inside other lists. Use `<ul>` for unordered items, `<ol>` for sequential steps.

Links and Navigation

```html
<!-- External link -->
<a href="https://www.example.com">Visit Example</a>

<!-- Internal link (same website) -->
<a href="about.html">About Page</a>

<!-- Link to section on same page -->
<a href="#contact">Go to Contact Section</a>

<!-- Email link -->
<a href="mailto:hello@example.com">Send Email</a>

<!-- Phone link -->
<a href="tel:+1234567890">Call Us</a>

<!-- Download link -->
<a href="document.pdf" download>Download PDF</a>
```

> 🔗 **Link Note:** Use `target="_blank"` to open links in new tabs, but add `rel="noopener noreferrer"` for security.

Images and Media

```html
```

```html
<!-- Image with alt text (always include!) -->
<img src="photo.jpg" alt="Description of image" width="300" height="200">

<!-- Responsive image -->
<img src="photo.jpg" alt="Description" style="max-width: 100%; height: auto;">

<!-- Audio -->
<audio controls>
  <source src="audio.mp3" type="audio/mpeg">
  Your browser does not support audio.
</audio>

<!-- Video -->
<video controls width="400">
  <source src="video.mp4" type="video/mp4">
  Your browser does not support video.
</video>
```

> 🖼️ **Image Tip:** Always include descriptive `alt` text for accessibility. Keep file sizes small for faster loading.

Tables

```html
```

```
<table>
  <thead>
    <tr>
      <th>Name</th>
      <th>Age</th>
      <th>City</th>
    </tr>
  </thead>
  <tbody>
    <tr>
      <td>John</td>
      <td>25</td>
      <td>New York</td>
    </tr>
    <tr>
      <td>Sarah</td>
      <td>30</td>
      <td>London</td>
    </tr>
  </tbody>
</table>
```

📊 **Table Note:** Use tables for tabular data only, not for layout. `<th>` headers help screen readers understand content.

Forms and Input

```
html
```

```html
<form action="/submit" method="POST">
  <!-- Text input -->
  <label for="name">Name:</label>
  <input type="text" id="name" name="name" required>

  <!-- Email input -->
  <label for="email">Email:</label>
  <input type="email" id="email" name="email" required>

  <!-- Password -->
  <label for="password">Password:</label>
  <input type="password" id="password" name="password">

  <!-- Textarea -->
  <label for="message">Message:</label>
  <textarea id="message" name="message" rows="4" cols="50"></textarea>

  <!-- Select dropdown -->
  <label for="country">Country:</label>
  <select id="country" name="country">
    <option value="us">United States</option>
    <option value="uk">United Kingdom</option>
    <option value="ca">Canada</option>
  </select>

  <!-- Radio buttons -->
  <fieldset>
    <legend>Gender:</legend>
    <input type="radio" id="male" name="gender" value="male">
    <label for="male">Male</label>
    <input type="radio" id="female" name="gender" value="female">
    <label for="female">Female</label>
  </fieldset>

  <!-- Checkboxes -->
  <input type="checkbox" id="newsletter" name="newsletter">
  <label for="newsletter">Subscribe to newsletter</label>

  <!-- Submit button -->
  <button type="submit">Submit Form</button>
</form>
```

> 📝 **Form Tip:** Always use `<label>` elements with `for` attributes - they make forms accessible and clickable.

Semantic HTML5 Elements

```html
<header>
  <nav>
    <ul>
      <li><a href="#home">Home</a></li>
      <li><a href="#about">About</a></li>
      <li><a href="#contact">Contact</a></li>
    </ul>
  </nav>
</header>

<main>
  <article>
    <header>
      <h1>Article Title</h1>
      <time datetime="2024-01-15">January 15, 2024</time>
    </header>
    <p>Article content goes here...</p>
  </article>

  <aside>
    <h2>Related Links</h2>
    <ul>
      <li><a href="#">Link 1</a></li>
      <li><a href="#">Link 2</a></li>
    </ul>
  </aside>
</main>

<footer>
  <p>&amp;copy; 2024 My Website. All rights reserved.</p>
</footer>
```

> 📐 **Semantic Tip:** Use semantic elements for better SEO and accessibility. They describe content meaning, not just appearance.

CSS Fundamentals

Linking CSS to HTML

```html
<!-- External CSS (recommended) -->
<link rel="stylesheet" href="styles.css">

<!-- Internal CSS -->
<style>
  body { background-color: lightblue; }
</style>

<!-- Inline CSS (avoid when possible) -->
<p style="color: red; font-size: 18px;">Styled paragraph</p>
```

> 🎨 **CSS Priority:** Inline styles override internal CSS, which overrides external CSS. External is best for maintainability.

CSS Selectors

```css

```

```css
/* Element selector */
p {
    color: blue;
    font-size: 16px;
}

/* Class selector */
.highlight {
    background-color: yellow;
    padding: 10px;
}

/* ID selector */
#header {
    background-color: navy;
    color: white;
}

/* Descendant selector */
nav ul li {
    list-style: none;
    display: inline-block;
}

/* Child selector */
.menu > li {
    margin-right: 20px;
}

/* Pseudo-classes */
a:hover {
    color: red;
    text-decoration: underline;
}

a:visited {
    color: purple;
}

/* Pseudo-elements */
p::first-line {
    font-weight: bold;
}
```

```css
p::before {
    content: "→ ";
    color: green;
}
```

> 🎯 **Selector Tip:** Use classes for styling multiple elements, IDs for unique elements. Keep specificity low for easier maintenance.

Text and Font Styling

```css
.text-styles {
    /* Font properties */
    font-family: 'Arial', sans-serif;
    font-size: 18px;
    font-weight: bold; /* or 100-900 */
    font-style: italic;
    line-height: 1.5; /* 1.5 times font size */
    letter-spacing: 1px;
    word-spacing: 2px;

    /* Text properties */
    color: #333333;
    text-align: center; /* left, right, center, justify */
    text-decoration: underline; /* none, underline, overline, line-through */
    text-transform: uppercase; /* lowercase, capitalize, uppercase */
    text-indent: 30px;
}
```

> 📝 **Font Note:** Always provide fallback fonts: `font-family: 'Custom Font', Arial, sans-serif`. Use web-safe fonts or Google Fonts.

Colors and Backgrounds

```css
```

```css
.color-examples {
  /* Color values */
  color: red; /* Named color */
  color: #ff0000; /* Hexadecimal */
  color: rgb(255, 0, 0); /* RGB */
  color: rgba(255, 0, 0, 0.5); /* RGB with transparency */
  color: hsl(0, 100%, 50%); /* HSL */
  color: hsla(0, 100%, 50%, 0.8); /* HSL with transparency */
}

.background-examples {
  /* Background colors */
  background-color: lightblue;

  /* Background images */
  background-image: url('image.jpg');
  background-size: cover; /* contain, cover, 100px 200px */
  background-position: center center;
  background-repeat: no-repeat; /* repeat, repeat-x, repeat-y */
  background-attachment: fixed; /* scroll, fixed */

  /* Background shorthand */
  background: url('image.jpg') no-repeat center center / cover;
}
```

> 🌈 **Color Tip:** Use HSL for easier color manipulation. RGBA/HSLA add transparency (alpha channel from 0-1).

Box Model and Layout

```css
```

```css
.box-model {
  /* Box dimensions */
  width: 300px;
  height: 200px;
  max-width: 100%; /* Responsive constraint */
  min-height: 150px;

  /* Padding (space inside element) */
  padding: 20px; /* All sides */
  padding: 10px 20px; /* Top/bottom left/right */
  padding: 10px 15px 20px 25px; /* Top right bottom left */

  /* Border */
  border: 2px solid #333;
  border-radius: 10px; /* Rounded corners */
  border-top: 1px dotted red;

  /* Margin (space outside element) */
  margin: 20px auto; /* Auto centers horizontally */
  margin-bottom: 30px;

  /* Box sizing */
  box-sizing: border-box; /* Includes padding and border in width */
}
```

> 📦 **Box Model:** Total width = width + padding + border + margin. Use `box-sizing: border-box` to include padding/border in width.

Display and Positioning

```css
```

```css
/* Display types */
.block { display: block; } /* Full width, new line */
.inline { display: inline; } /* Only content width, same line */
.inline-block { display: inline-block; } /* Best of both */
.none { display: none; } /* Completely hidden */

/* Flexbox container */
.flex-container {
    display: flex;
    justify-content: space-between; /* flex-start, center, space-around */
    align-items: center; /* flex-start, flex-end, stretch */
    flex-direction: row; /* column, row-reverse, column-reverse */
    flex-wrap: wrap; /* nowrap, wrap-reverse */
    gap: 20px; /* Space between items */
}

.flex-item {
    flex: 1; /* Grow to fill space */
    flex-shrink: 0; /* Don't shrink */
    flex-basis: 200px; /* Base size */
}

/* Grid container */
.grid-container {
    display: grid;
    grid-template-columns: 1fr 2fr 1fr; /* 3 columns */
    grid-template-rows: 100px auto 50px; /* 3 rows */
    grid-gap: 20px; /* gap between items */
    grid-template-areas:
        "header header header"
        "sidebar main main"
        "footer footer footer";
}

.grid-item {
    grid-area: header; /* Assign to named area */
    grid-column: 1 / 3; /* Span columns 1 to 3 */
    grid-row: 2 / 4; /* Span rows 2 to 4 */
}

/* Positioning */
.positioned {
    position: relative; /* Relative to normal position */
```

```
    top: 10px;
    left: 20px;
    z-index: 10; /* Stacking order */
}

.absolute {
    position: absolute; /* Relative to nearest positioned parent */
    top: 0;
    right: 0;
}

.fixed {
    position: fixed; /* Relative to viewport */
    bottom: 20px;
    right: 20px;
}

.sticky {
    position: sticky; /* Sticky when scrolling */
    top: 0;
}
```

> 🔧 **Layout Tip:** Use Flexbox for 1D layouts (rows/columns), Grid for 2D layouts. Flexbox is perfect for navigation and centering.

Responsive Design

```
css
```

```css
/* Mobile-first approach */
.responsive {
    width: 100%;
    padding: 10px;
    font-size: 14px;
}

/* Tablet styles */
@media screen and (min-width: 768px) {
    .responsive {
        width: 750px;
        margin: 0 auto;
        font-size: 16px;
    }
}

/* Desktop styles */
@media screen and (min-width: 1024px) {
    .responsive {
        width: 1000px;
        font-size: 18px;
    }
}

/* Print styles */
@media print {
    .no-print {
        display: none;
    }

    body {
        font-size: 12pt;
        color: black;
    }
}

/* Responsive images */
.responsive-image {
    max-width: 100%;
    height: auto;
}

/* Responsive text */
```

```css
.responsive-text {
    font-size: clamp(14px, 4vw, 24px); /* Min, preferred, max */
}
```

> 📱 **Responsive Note:** Start with mobile styles, then add larger screen styles. Use relative units (%, em, rem, vw, vh) for flexibility.

Animations and Transitions

```
css
```

```css
/* Transitions (smooth changes) */
.button {
    background-color: blue;
    color: white;
    padding: 10px 20px;
    border: none;
    border-radius: 5px;
    cursor: pointer;
    transition: background-color 0.3s ease, transform 0.2s ease;
}

.button:hover {
    background-color: darkblue;
    transform: translateY(-2px);
}

/* Keyframe animations */
@keyframes bounce {
    0%, 100% {
        transform: translateY(0);
    }
    50% {
        transform: translateY(-20px);
    }
}

.animated-element {
    animation: bounce 2s infinite ease-in-out;
}

/* Loading spinner example */
@keyframes spin {
    0% { transform: rotate(0deg); }
    100% { transform: rotate(360deg); }
}

.spinner {
    border: 4px solid #f3f3f3;
    border-top: 4px solid #3498db;
    border-radius: 50%;
    width: 40px;
    height: 40px;
```

```
    animation: spin 1s linear infinite;
}
```

> ✨ **Animation Tip:** Use transitions for user interactions, animations for ongoing effects. Keep animations subtle and purposeful.

Common CSS Patterns

Card Component

```css
.card {
    background: white;
    border-radius: 8px;
    box-shadow: 0 2px 10px rgba(0, 0, 0, 0.1);
    padding: 20px;
    margin: 20px;
    max-width: 300px;
    transition: transform 0.3s ease, box-shadow 0.3s ease;
}

.card:hover {
    transform: translateY(-5px);
    box-shadow: 0 4px 20px rgba(0, 0, 0, 0.15);
}

.card-title {
    margin: 0 0 10px 0;
    color: #333;
}

.card-content {
    color: #666;
    line-height: 1.6;
}
```

Navigation Bar

```css
```

```css
.navbar {
    display: flex;
    justify-content: space-between;
    align-items: center;
    padding: 1rem 2rem;
    background-color: #333;
    color: white;
}

.nav-menu {
    display: flex;
    list-style: none;
    margin: 0;
    padding: 0;
    gap: 2rem;
}

.nav-link {
    color: white;
    text-decoration: none;
    transition: color 0.3s ease;
}

.nav-link:hover {
    color: #ff6b6b;
}

/* Mobile menu */
@media screen and (max-width: 768px) {
    .nav-menu {
        flex-direction: column;
        position: absolute;
        top: 100%;
        left: 0;
        right: 0;
        background-color: #333;
        padding: 1rem;
        display: none;
    }

    .nav-menu.active {
        display: flex;
```

```
    }
}
```

> 🧭 **Navigation Tip:** Make navigation clear and consistent. Use hover effects and active states to guide users.

Center Content (Multiple Methods)

```css
/* Method 1: Flexbox */
.center-flex {
    display: flex;
    justify-content: center;
    align-items: center;
    min-height: 100vh;
}

/* Method 2: Grid */
.center-grid {
    display: grid;
    place-items: center;
    min-height: 100vh;
}

/* Method 3: Absolute positioning */
.center-absolute {
    position: absolute;
    top: 50%;
    left: 50%;
    transform: translate(-50%, -50%);
}

/* Method 4: Margin auto (horizontal only) */
.center-margin {
    max-width: 800px;
    margin: 0 auto;
}
```

Practical Example: Complete Web Page

```html
```

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>My Portfolio</title>
    <style>
        * {
            margin: 0;
            padding: 0;
            box-sizing: border-box;
        }

        body {
            font-family: 'Arial', sans-serif;
            line-height: 1.6;
            color: #333;
        }

        .container {
            max-width: 1200px;
            margin: 0 auto;
            padding: 0 20px;
        }

        header {
            background: linear-gradient(135deg, #667eea 0%, #764ba2 100%);
            color: white;
            text-align: center;
            padding: 4rem 0;
        }

        .hero h1 {
            font-size: 3rem;
            margin-bottom: 1rem;
            animation: fadeInUp 1s ease;
        }

        .hero p {
            font-size: 1.2rem;
            opacity: 0.9;
        }
```

```
    .section {
        padding: 4rem 0;
    }

    .cards {
        display: grid;
        grid-template-columns: repeat(auto-fit, minmax(300px, 1fr));
        gap: 2rem;
        margin-top: 2rem;
    }

    .card {
        background: white;
        padding: 2rem;
        border-radius: 10px;
        box-shadow: 0 5px 15px rgba(0, 0, 0, 0.1);
        transition: transform 0.3s ease;
    }

    .card:hover {
        transform: translateY(-10px);
    }

    @keyframes fadeInUp {
        from {
            opacity: 0;
            transform: translateY(30px);
        }
        to {
            opacity: 1;
            transform: translateY(0);
        }
    }

    @media (max-width: 768px) {
        .hero h1 {
            font-size: 2rem;
        }

        .section {
            padding: 2rem 0;
        }
    }
</style>
```

```
</head>
<body>
  <header>
    <div class="container">
      <div class="hero">
        <h1>John Doe</h1>
        <p>Web Developer & Designer</p>
      </div>
    </div>
  </header>

  <main>
    <section class="section">
      <div class="container">
        <h2>My Projects</h2>
        <div class="cards">
          <div class="card">
            <h3>Project 1</h3>
            <p>A responsive website built with HTML, CSS, and JavaScript.</p>
          </div>
          <div class="card">
            <h3>Project 2</h3>
            <p>An e-commerce platform with modern design principles.</p>
          </div>
          <div class="card">
            <h3>Project 3</h3>
            <p>A mobile-first web application with interactive features.</p>
          </div>
        </div>
      </div>
    </section>
  </main>
</body>
</html>
```

> 🎯 **Project Tip:** Start with a wireframe, then build mobile-first. Always test on multiple devices and browsers.

Quick Reference Tips

CSS Best Practices

- Use external stylesheets for better organization
- Follow a consistent naming convention (BEM, camelCase, etc.)
- Group related properties together
- Use CSS custom properties (variables) for consistency
- Minimize the use of `!important`
- Optimize for performance (minimize reflows/repaints)

> ⚡ **Performance Tip:** Put CSS in `<head>` and JavaScript before closing `</body>` tag for optimal loading.

Common CSS Units

Unit	Description	Best For
`px`	Pixels (absolute)	Borders, small details
`%`	Percentage of parent	Widths, responsive design
`em`	Relative to element's font-size	Padding, margins
`rem`	Relative to root font-size	Font sizes, spacing
`vw/vh`	Viewport width/height	Full-screen elements
`fr`	Fractional units (Grid)	Grid column/row sizing

> 📏 **Units Tip:** Use `rem` for typography, `%` or `vw/vh` for layouts, `px` for small details like borders.

Debugging CSS

- Use browser Developer Tools (F12)
- Check the computed styles panel
- Use border/background colors to visualize layouts
- Validate your HTML and CSS
- Test in multiple browsers

> 🐛 **Debug Tip:** Add `border: 1px solid red;` temporarily to see element boundaries and troubleshoot layout issues.

Accessibility Considerations

- Ensure sufficient color contrast (4.5:1 ratio minimum)
- Don't rely on color alone to convey information
- Make interactive elements keyboard accessible
- Use semantic HTML elements
- Provide alternative text for images
- Test with screen readers

> ♿ **A11y Note:** Good accessibility benefits everyone. Use tools like WAVE or axe to test your sites.

Next Steps

- Learn CSS preprocessors (Sass, Less)
- Explore CSS frameworks (Bootstrap, Tailwind)
- Master CSS Grid and Flexbox layouts
- Study modern CSS features (Container Queries, CSS Grid)
- Practice responsive design patterns
- Learn about CSS architecture (OOCSS, BEM, SMACSS)

This cheat sheet covers essential HTML and CSS concepts. Practice building real projects to reinforce these concepts and develop your web development skills.