Module 1: CSS - Comprehensive Study Notes

Table of Contents

---

1. Introduction to CSS

What is CSS?

CSS (Cascading Style Sheets) is the language used to style and layout web pages. While HTML provides the structure, CSS makes it beautiful.

Definition Breakdown:

- **Cascading:** Styles can override each other based on specificity and order
- **Style:** The visual appearance and layout
- **Sheets:** Files containing styling rules

Real-Life Analogy: If building a website is like building a house:

- **HTML** = The structure (walls, doors, windows, rooms)
- **CSS** = The interior design (paint colors, furniture placement, lighting, decorations)
- **JavaScript** = The functionality (electricity, plumbing, smart features)

Why Do We Need CSS?

Without CSS, every website would look like a plain text document from 1995. CSS allows you to:

- Control colors, fonts, and sizes
- Create layouts and position elements
- Add animations and transitions
- Make websites responsive on different devices
- Improve user experience and visual appeal

Real-World Example:

WITHOUT CSS:
Plain text, black and white, no spacing, everything left-aligned

WITH CSS:
Beautiful colors, perfect spacing, organized layout, attractive design

The Role of CSS in Web Development

Example: Look at any modern website (Amazon, Netflix, Instagram):

- Product cards with shadows and hover effects → CSS
- Navigation menus that highlight on hover → CSS
- Responsive layout that adapts to mobile → CSS
- Smooth animations when scrolling → CSS
- Beautiful typography and spacing → CSS

---

2. CSS Syntax and Inclusion Methods

CSS Syntax Structure

Every CSS rule follows this pattern:

```
selector {
   property: value;
   property: value;
}
```

Breaking it down:

- **Selector:** Which HTML element to style
- **Property:** What aspect to change (color, size, etc.)
- **Value:** How to change it

Real-Life Analogy: Like giving instructions to an interior designer:

- **Selector:** "All bedrooms" (which rooms)
- **Property:** "wall color" (what to change)
- **Value:** "light blue" (how to change it)

Example:

```
h1 {
   color: blue;
   font-size: 32px;
   text-align: center;
}
```

Translation: "Make all h1 headings blue, 32 pixels in size, and centered."

Three Ways to Include CSS

Method 1: Inline CSS (Directly in HTML)

Syntax:

```
<h1 style="color: blue; font-size: 32px;">Hello World</h1>
```

Real-Life Analogy: Like writing notes directly on a document with a pen.

Pros:

✅ Quick for testing

✅ Highest priority (overrides other styles)

Cons:

❌ Hard to maintain

❌ Can't reuse styles

❌ Makes HTML messy

❌ No separation of concerns

When to Use: Only for quick tests or email HTML (where external CSS doesn't work).

Example:

```
<p style="color: red; font-weight: bold;">This is a red, bold paragraph.</p>
<p style="color: red; font-weight: bold;">This is another red, bold paragraph.</p>
<!-- Notice we have to repeat the same styles! BAD PRACTICE -->
```

Method 2: Internal CSS (In the <head> section)

Syntax:

```
<!DOCTYPE html>
<html>
<head>
   <style>
      h1 {
         color: blue;
         font-size: 32px;
      }
      p {
         color: gray;
         line-height: 1.6;
      }
   </style>
</head>
<body>
   <h1>Welcome</h1>
   <p>This is a paragraph.</p>
</body>
</html>
```

Real-Life Analogy: Like having a style guide at the beginning of a document.

Pros:

✅ Styles apply to entire page

✅ Better than inline

✅ Good for single-page websites

Cons:

❌ Styles only work on one HTML page

❌ Can't share styles across multiple pages

❌ Makes HTML file larger

When to Use: Small websites with only one page, or page-specific styles.

Method 3: External CSS (Separate CSS File) ⭐ RECOMMENDED

Step 1: Create a CSS file (e.g., `styles.css`)

```
/* styles.css */
h1 {
    color: blue;
    font-size: 32px;
}

p {
    color: gray;
    line-height: 1.6;
}
```

Step 2: Link it in your HTML

```
<!DOCTYPE html>
<html>
<head>
    <link rel="stylesheet" href="styles.css">
</head>
<body>
    <h1>Welcome</h1>
    <p>This is a paragraph.</p>
</body>
</html>
```

Real-Life Analogy: Like having a company brand guidebook that everyone references.

Pros:

✅ Can be used across multiple pages

✅ Clean separation of HTML and CSS

✅ Easy to maintain

✅ Browser caches the file (faster loading)

✅ Multiple people can work on CSS separately

Cons:

❌ Requires an extra HTTP request (minimal impact)

When to Use: Always! This is the professional standard.

Best Practice Example:

```
project/
├── index.html
├── about.html
├── contact.html
└── css/
    └── styles.css   ← One CSS file for all pages
```

---

3. Selectors, Properties, and Values

Understanding Selectors

Selectors tell the browser which HTML elements to style.

1. Element Selector (Type Selector)

Targets: All elements of a specific type

Syntax:

```
elementname {
    property: value;
}
```

Example:

```
p {
    color: blue;
}

h1 {
    font-size: 36px;
}
```

```
a {
   text-decoration: none;
}
```

Result: All <p> tags are blue, all <h1> tags are 36px, all links have no underline.

Real-Life Analogy: "Paint ALL bedrooms yellow" - affects every bedroom in the house.

2. Class Selector

Targets: All elements with a specific class attribute

Syntax:

```
.classname {
   property: value;
}
```

HTML:

```
<p class="highlight">This paragraph is highlighted.</p>
<p>This paragraph is normal.</p>
<p class="highlight">This paragraph is also highlighted.</p>
```

CSS:

```
.highlight {
   background-color: yellow;
   padding: 10px;
}
```

Result: Only paragraphs with class="highlight" get the yellow background.

Real-Life Analogy: "Paint all rooms marked as 'guest rooms' with blue" - only affects rooms with that label.

Naming Convention:

```
.btn-primary { }      /* Good: descriptive */
.error-message { }    /* Good: clear purpose */
.main-nav { }         /* Good: uses hyphens */
```

```
.p1 { }              /* Bad: not descriptive */
.redText { }         /* Bad: describes appearance */
```

3. ID Selector

Targets: One unique element (ID should be unique on page)

Syntax:

```
#idname {
   property: value;
}
```

HTML:

```
<div id="header">This is the header</div>
<div id="main-content">This is the main content</div>
```

CSS:

```
#header {
   background-color: navy;
   color: white;
}

#main-content {
   padding: 20px;
}
```

Real-Life Analogy: "Paint the master bedroom (specific, unique room) with purple."

Class vs ID - When to Use:

- **Class (`.`)**: Use for multiple elements that share styles (buttons, cards, tags)
- **ID (`#`)**: Use for one unique element per page (header, footer, specific section)

Rule of Thumb:

- Use classes 95% of the time
- IDs have higher specificity and can cause issues

4. Universal Selector

Targets: All elements on the page

Syntax:

```
* {
    margin: 0;
    padding: 0;
    box-sizing: border-box;
}
```

Use Case: Commonly used for CSS reset to remove browser default styles.

Real-Life Analogy: "Paint EVERYTHING in the house" - affects every single thing.

5. Descendant Selector

Targets: Elements inside other elements

Syntax:

```
parent child {
    property: value;
}
```

Example:

```
<div class="card">
    <h2>Card Title</h2>
    <p>Card description</p>
</div>

<h2>Regular Title</h2>
```

CSS:

```
.card h2 {
    color: blue;
}
```

Result: Only the `<h2>` inside `.card` turns blue, not the regular h2.

Real-Life Analogy: "Paint only the chairs that are in the dining room."

6. Multiple Selectors (Grouping)

Targets: Multiple elements with same styles

Syntax:

```
selector1, selector2, selector3 {
   property: value;
}
```

Example:

```
h1, h2, h3 {
   font-family: Arial, sans-serif;
   color: #333;
}
```

Result: All headings get the same font and color.

Real-Life Analogy: "Paint the kitchen, bathroom, and hallway with white" - one instruction for multiple rooms.

7. Attribute Selector

Targets: Elements with specific attributes

Example:

```
/* Links that open in new tab */
a[target="_blank"] {
   color: red;
}

/* Email input fields */
input[type="email"] {
   border: 2px solid blue;
}

/* Images with specific alt text */
img[alt="logo"] {
   width: 100px;
}
```

8. Pseudo-classes

Targets: Elements in a specific state

Common Pseudo-classes:

```
/* When mouse hovers over link */
a:hover {
    color: red;
    text-decoration: underline;
}

/* First child element */
li:first-child {
    font-weight: bold;
}

/* Last child element */
li:last-child {
    border-bottom: none;
}

/* Nth child */
li:nth-child(2) {
    color: blue;
}

/* Even rows in a table */
tr:nth-child(even) {
    background-color: #f2f2f2;
}

/* When input is focused */
input:focus {
    border-color: blue;
    outline: none;
}
```

Real-Life Example: Navigation Menu

```
.nav-link {
    color: gray;
```

```
    padding: 10px;
}

.nav-link:hover {
    color: blue;
    background-color: #f0f0f0;
}

.nav-link:active {
    color: darkblue;
}
```

Common CSS Properties and Values

Text Properties

```
/* Font family */
font-family: Arial, Helvetica, sans-serif;

/* Font size */
font-size: 16px;        /* pixels */
font-size: 1.5em;       /* relative to parent */
font-size: 1.5rem;      /* relative to root */

/* Font weight */
font-weight: normal;    /* 400 */
font-weight: bold;      /* 700 */
font-weight: 600;       /* semi-bold */

/* Font style */
font-style: italic;
font-style: normal;

/* Text alignment */
text-align: left;
text-align: center;
text-align: right;
text-align: justify;

/* Text decoration */
text-decoration: none;          /* Remove underline */
text-decoration: underline;
text-decoration: line-through;
```

```
/* Text transform */
text-transform: uppercase;       /* HELLO */
text-transform: lowercase;       /* hello */
text-transform: capitalize;      /* Hello */

/* Line height (spacing between lines) */
line-height: 1.6;              /* 1.6 times font size */

/* Letter spacing */
letter-spacing: 2px;

/* Word spacing */
word-spacing: 5px;
```

Real-World Example: Article Text

```
.article {
    font-family: Georgia, serif;
    font-size: 18px;
    line-height: 1.8;
    color: #333;
    text-align: justify;
}
```

Color Properties

```
/* Text color */
color: red;
color: #FF0000;         /* Hex code */
color: rgb(255, 0, 0);  /* RGB */
color: rgba(255, 0, 0, 0.5);  /* RGB with transparency */

/* Background color */
background-color: blue;
background-color: #0000FF;
background-color: rgba(0, 0, 255, 0.3);  /* 30% opacity */
```

Color Formats Explained:

1. **Named Colors:** `red`, `blue`, `green` (limited options)
2. **Hex Codes:** `#FF0000` (most common)
    - First 2 digits: Red (00-FF)
    - Middle 2 digits: Green (00-FF)

    - Last 2 digits: Blue (00-FF)
3. **RGB:** `rgb(255, 0, 0)` (values 0-255)
4. **RGBA:** `rgba(255, 0, 0, 0.5)` (A = Alpha/transparency, 0-1)

Real-Life Analogy: Like mixing paint colors - RGB tells you how much red, green, and blue to mix.

Background Properties

```
/* Background color */
background-color: #f0f0f0;

/* Background image */
background-image: url('image.jpg');

/* Background repeat */
background-repeat: no-repeat;     /* Don't repeat */
background-repeat: repeat-x;      /* Repeat horizontally */
background-repeat: repeat-y;      /* Repeat vertically */

/* Background position */
background-position: center;
background-position: top right;
background-position: 50% 50%;

/* Background size */
background-size: cover;           /* Cover entire element */
background-size: contain;         /* Fit inside element */
background-size: 100px 100px;     /* Specific size */

/* Shorthand */
background: #f0f0f0 url('image.jpg') no-repeat center/cover;
```

Real-World Example: Hero Section

```
.hero {
    background-image: url('hero-bg.jpg');
    background-size: cover;
    background-position: center;
    background-repeat: no-repeat;
    height: 500px;
    color: white;
}
```

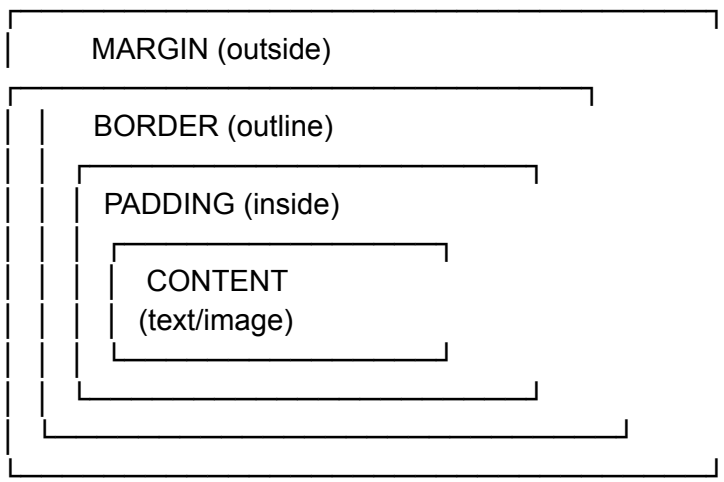
---

4. The Box Model

Understanding the Box Model

Every HTML element is a rectangular box. The box model describes how space is calculated around elements.

The Four Parts:



Real-Life Analogy: Think of a framed picture:

- **Content:** The actual photograph
- **Padding:** The white matting around the photo
- **Border:** The frame itself
- **Margin:** Space between the frame and other pictures on the wall

1. Content

The actual content (text, images, etc.)

```
.box {
   width: 300px;
   height: 200px;
}
```

2. Padding

Space between content and border (INSIDE the element)

```
/* All sides */
padding: 20px;

/* Vertical | Horizontal */
padding: 10px 20px;

/* Top | Right | Bottom | Left (clockwise) */
padding: 10px 15px 20px 25px;

/* Individual sides */
padding-top: 10px;
padding-right: 15px;
padding-bottom: 20px;
padding-left: 25px;
```

Example:

```
.card {
    padding: 20px;
    background-color: #f0f0f0;
}
```

Result: Creates 20px of breathing room inside the card between the edge and the content.

3. Border

Line around the element

```
/* All properties */
border: 2px solid black;

/* Individual properties */
border-width: 2px;
border-style: solid;    /* solid, dashed, dotted, double, none */
border-color: black;

/* Individual sides */
border-top: 2px solid red;
border-right: 1px dashed blue;
border-bottom: 3px double green;
border-left: 2px dotted orange;
```

```
/* Border radius (rounded corners) */
border-radius: 10px;              /* All corners */
border-radius: 10px 20px;         /* Top-left & bottom-right | Top-right & bottom-left */
border-radius: 50%;               /* Perfect circle (if width = height) */
```

Real-World Examples:

```
/* Card with border */
.card {
    border: 1px solid #ddd;
    border-radius: 8px;
}

/* Button with rounded corners */
.button {
    border: 2px solid blue;
    border-radius: 25px;
}

/* Avatar (circle) */
.avatar {
    width: 100px;
    height: 100px;
    border-radius: 50%;
    border: 3px solid white;
}
```

4. Margin

Space OUTSIDE the element (between elements)

```
/* All sides */
margin: 20px;

/* Vertical | Horizontal */
margin: 10px 20px;

/* Top | Right | Bottom | Left */
margin: 10px 15px 20px 25px;

/* Individual sides */
margin-top: 10px;
```

```
margin-right: 15px;
margin-bottom: 20px;
margin-left: 25px;

/* Auto (center element horizontally) */
margin: 0 auto;
```

Common Use Cases:

```
/* Center a container */
.container {
    width: 1200px;
    margin: 0 auto;  /* Top/bottom: 0, Left/right: auto */
}

/* Space between sections */
section {
    margin-bottom: 40px;
}

/* Remove margin from last element */
.list-item:last-child {
    margin-bottom: 0;
}
```

Box-Sizing Property

Problem: By default, padding and border are ADDED to width/height.

```
/* Default behavior */
.box {
    width: 300px;
    padding: 20px;
    border: 5px solid black;
}
/* Actual width = 300 + 40 (padding) + 10 (border) = 350px! */
```

Solution: Use `box-sizing: border-box`

```
* {
    box-sizing: border-box;
}
```

```
.box {
    width: 300px;
    padding: 20px;
    border: 5px solid black;
}
/* Now actual width = 300px (padding and border included) */
```

Real-Life Analogy:

- Default: "I want a 300px box, PLUS padding and border"
- Border-box: "I want a 300px box INCLUDING padding and border"

Best Practice: Always use this CSS reset:

```
* {
    margin: 0;
    padding: 0;
    box-sizing: border-box;
}
```

Complete Box Model Example

```
<div class="product-card">
    <img src="product.jpg" alt="Product">
    <h3>Product Name</h3>
    <p>$29.99</p>
    <button>Add to Cart</button>
</div>

.product-card {
    width: 300px;
    padding: 20px;               /* Space inside */
    border: 1px solid #ddd;      /* Border around */
    border-radius: 10px;         /* Rounded corners */
    margin: 20px;                /* Space outside */
    background-color: white;
    box-shadow: 0 2px 8px rgba(0,0,0,0.1);
}
```