WEB ENGINEERING FUNDAMENTALS

CSS Fundamentals

A Complete Beginner's Guide to Styling Modern Websites

Learn CSS from first principles. Understand how styles are applied, how selectors work, how the cascade determines final styles, and how to transform plain HTML into visually appealing, responsive web pages.

haas.dev • dev-roast-app.vercel.app

Table of Contents

1. Introduction
2. What Is CSS?
3. Why CSS Is Important
4. How CSS Works
5. Ways to Add CSS
6. CSS Syntax
7. Understanding Selectors
8. Cascade, Specificity, Inheritance
9. Common CSS Properties
10. Colors, Units, Typography
11. CSS Best Practices
12. Real World Example
13. Common Beginner Mistakes
14. Practical Action Plan
15. Mini Architecture Challenge
16. Key Takeaways
17. Summary Page
18. CSS Cheat Sheet
19. Related Resources
20. Recommended Next Learning Path

1. Introduction

Imagine opening a webpage that contains only black text on a white background.

Everything works, but it looks unfinished.

Now imagine the same webpage with:

- beautiful typography
- attractive colors
- spacing between sections
- responsive layouts
- modern buttons
- smooth hover effects

The difference is CSS.

HTML provides the structure of a webpage.

CSS controls how that structure looks.

Without CSS, websites would still function, but they would be difficult to use and visually unappealing.

Learning CSS is not about memorizing hundreds of properties. It is about understanding how browsers apply styles and how different rules work together.

2. What Is CSS?

CSS stands for Cascading Style Sheets.

It is the language used to control the presentation of HTML documents.

CSS defines:

- colors
- fonts
- spacing
- borders
- layouts
- positioning
- animations
- responsive behavior

HTML answers:

> What is this?

CSS answers:

> How should it look?

3. Why CSS Is Important

CSS improves both appearance and usability.

It helps developers:

- create professional interfaces
- maintain consistent designs
- separate design from content
- build responsive websites
- improve readability
- create reusable design systems

Modern websites rely heavily on CSS for user experience.

4. How CSS Works

The browser follows this process:

```
HTML
↓
Browser builds the DOM
↓
Browser downloads CSS
↓
Browser creates the CSSOM
↓
DOM + CSSOM
↓
Render Tree
↓
Layout
↓
Painting
```

Every visible style on a webpage comes from CSS rules applied during this process.

> Learn More
> Read How Browsers Work to understand how browsers combine HTML and CSS before rendering a webpage.

5. Ways to Add CSS

There are three common ways to use CSS.

Inline CSS

Styles are written directly inside an HTML element.

```
<h1 style="color: blue;">Welcome</h1>
```

Useful for quick testing.

Not recommended for large projects.

Internal CSS

Styles are written inside the <style> element.

```
<style>

h1{

color:blue;

}

</style>
```

Suitable for small webpages.

External CSS

Styles are stored in a separate .css file.

```
<link rel="stylesheet" href="style.css">
```

This is the professional approach.

Benefits include:

- reusable styles
- cleaner HTML
- easier maintenance
- better scalability

6. CSS Syntax

Every CSS rule consists of three parts.

```
selector{

property:value;

}
```

Example:

```
h1{

color:blue;

font-size:40px;

}
```

Here:

- h1 is the selector.
- color is a property.
- blue is the value.

7. Understanding Selectors

Selectors tell CSS which HTML elements to style.

Element Selector

Targets every element of the same type.

```
p{

color:black;

}
```

Class Selector

Targets elements sharing the same class.

```
.card{

padding:20px;

}
```

Classes are reusable.

Professional developers use classes extensively.

ID Selector

Targets one unique element.

```
#header{

background:black;

}
```

IDs should generally be unique within a page.

Universal Selector

Targets every element.

```
*{

margin:0;

padding:0;

}
```

Often used for CSS resets.

8. The Cascade, Specificity, and Inheritance

One of the biggest reasons beginners get confused is that multiple CSS rules can target the same element.

The browser decides which rule wins using three concepts.

Cascade

If two rules have equal importance, the later one usually wins.

Specificity

More specific selectors override less specific ones.

For example:

- An ID selector has higher priority than a class selector.
- A class selector has higher priority than an element selector.

Inheritance

Some properties automatically pass from parent elements to child elements.

Examples include:

- font-family
- color

Not every property is inherited.

Understanding these three concepts helps explain why styles sometimes behave unexpectedly.

9. Common CSS Properties

Some of the most frequently used properties are:

Text

- color
- font-size
- font-family
- font-weight
- text-align
- line-height

Background

- background-color
- background-image

Spacing

- margin
- padding

Borders

- border
- border-radius

Size

- width
- height
- max-width

Display

- display
- visibility

These properties appear in almost every web project.

10. Colors, Units, and Typography

CSS supports multiple ways to define colors.

Examples:

- Named colors
- HEX
- RGB
- RGBA
- HSL

Common measurement units include:

- px
- %
- em
- rem
- vw
- vh

Choosing appropriate units makes layouts more flexible.

Typography also plays a major role.

Good typography improves readability and user experience.

11. CSS Best Practices

Develop these habits early.

- Use external CSS files.
- Group related styles together.
- Use meaningful class names.
- Avoid unnecessary repetition.
- Keep selectors simple.
- Maintain consistent spacing and indentation.
- Build reusable components.

Clean CSS is easier to understand and maintain.

12. Real World Example

Imagine styling the homepage of haas.dev.

Without CSS:

- plain text
- no spacing
- default buttons
- unorganized layout

With CSS:

- consistent typography
- branded colors
- responsive cards
- clean navigation
- modern buttons
- professional layout

The content remains the same.

Only the presentation changes.

13. Common Beginner Mistakes

- Mixing HTML and CSS responsibilities.
- Using inline CSS everywhere.
- Choosing confusing class names.
- Ignoring specificity.
- Writing duplicate styles.
- Using fixed widths for every element.
- Copying CSS without understanding it.

14. Practical Action Plan

Create a simple webpage containing:

- heading
- paragraph
- button
- image
- navigation

Then create an external CSS file.

Style every element using:

- colors
- typography
- spacing
- borders

Do not copy designs.

Experiment with different properties until you understand their effects.

15. Mini Architecture Challenge

Challenge

Design the visual style for an online course landing page.

Before writing CSS, decide:

- primary colors
- typography
- spacing scale
- button styles
- card design
- navigation appearance

Create a simple style guide first.

Then implement it using reusable CSS classes.

16. Key Takeaways

- CSS controls presentation.
- HTML provides structure.
- External CSS is the professional approach.
- Selectors determine which elements receive styles.
- The cascade, specificity, and inheritance decide which styles are applied.
- Clean CSS improves scalability and maintainability.

17. Summary Page

CSS Cheat Sheet

CSS = Presentation

HTML = Structure

External CSS is preferred

Selectors target elements

Classes are reusable

IDs should be unique

Cascade resolves conflicts

Specificity determines priority

Inheritance passes some styles to children

Beginner CSS Workflow

```
Plan design
↓
Write semantic HTML
↓
Create external CSS
↓
Style typography
↓
Add colors
↓
Adjust spacing
↓
Test responsiveness
↓
Refactor duplicate styles
```

18. CSS Cheat Sheet

A quick-reference checklist you can keep beside your editor while you build.

- [ ] Use an external .css file linked in <head>
- [ ] Class names describe purpose, not appearance
- [ ] Avoid inline styles except for quick testing
- [ ] Keep specificity as low as possible
- [ ] Group related properties together (text, spacing, layout)
- [ ] Use rem/em for scalable typography and spacing
- [ ] Test the design at multiple screen widths
- [ ] Remove duplicate or unused CSS rules

19. Related Resources

HTML Fundamentals

Why read it: Understand the structure that CSS styles.

Semantic HTML

Why read it: Learn how meaningful HTML improves styling and accessibility.

How Browsers Work

Why read it: Understand how browsers build the CSSOM and render pages.

Responsive Web Design

Why read it: Learn how CSS adapts layouts for different screen sizes.

Anatomy of a Modern Website

Why read it: Understand the page components you'll style with CSS.

20. Recommended Next Learning Path

```
Step 1

HTML Fundamentals
↓
Step 2

Semantic HTML
↓
Step 3

CSS Fundamentals (Current PDF)
↓
Step 4

CSS Box Model
↓
Step 5

Flexbox
↓
Step 6

CSS Grid
↓
Step 7

Responsive Web Design
↓
Step 8

JavaScript Fundamentals
```

---

haas.dev • dev-roast-app.vercel.app