

Intermediate CSS

1. What is Intermediate CSS?

Intermediate CSS builds on basic styling. While basic CSS covers simple properties (colors, fonts, margins), intermediate CSS teaches you how to take control over page layout, element behavior, & visual structure - making real websites, instead of plain text pages.

Key goals

- Understand how & why elements take up space
- Learn how CSS rules cascade & interact
- Use more advanced properties for layout & visual control

2. CSS Selectors Beyond the Basics

Type of Selectors

Selector	What it Targets	Example
Class	All elements with a given class	<code>.card { ... }</code>
ID	One unique element	<code>#header { ... }</code>
child	Direct children only	<code>ul > li { ... }</code>
Attribute	Elements with an attribute	<code>input[type='text'] { ... }</code>
Pseudo-class	Special States	<code>a: hover { ... }</code>
Descendant	Elements inside others	<code>nav ul li { ... }</code>

why selectors matter

The more precise your selector, the more control you have over which elements a rule applies to. Understanding selectors prevents styles from bleeding onto the wrong elements.

Pro Tip

The most specific selector wins when two rules conflict (unless !important is used - move on that later).

3. Specificity & the Cascade

CSS stands for Cascading Style Sheets - meaning rules "stack" & resolve in order.

Specificity Hierarchy (lowest \rightarrow highest)

1. Element Selector (p hi)
2. Class/attribute Selectors (.btn, [type="text"])
3. ID Selectors (#main)
4. Inline styles (style="...")
5. !important (override but risky)

specificity Example

```
p { color: blue; }
p.text { color: red; }
#main p { color: green; }
```

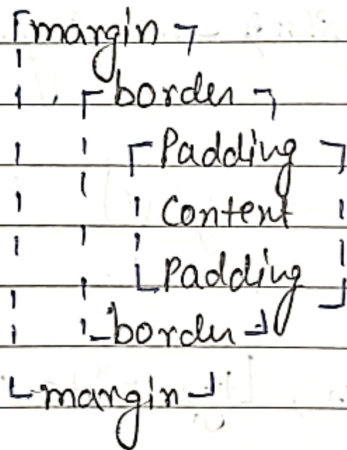
final color depends on which has higher specificity & where it appears

4. Box Model (most imp CSS concept)

Every HTML element is a box that includes

- Content text / images
- Padding space inside the border
- Border line around padding / content
- Margin space outside the border

Visualizing the box model helps you debug layout issues like unexpected spacing



Key properties

- padding : 20px ;
- border : 2px solid black ;
- margin : 10px auto ;

important By default width refers only to content; padding & border add to total size. You can change this with!

box-sizing : border-box ;

This makes widths include padding & border - often easier to work with

5 Display & Positioning

Display Types

values

Behavior

block

Takes full width, starts on new line

inline

only as wide as content, flows in line

inline-block

like inline but respects width / height

none

hidden from layout

Positioning

- static default normal flow
- relative offset relative to itself
- absolute removed from normal flow
- fixed stays in viewport
- sticky acts like relative until scroll threshold

```
header { position: fixed; top: 0; width: 100%; }
```

6. flexbox (often part of Intermediate CSS)

flexbox lets you arrange elements in a row or column & distribute space intelligently

* Parent flex container

- Container {

- display: flex;

- justify-content: space-between;

- align-items: center;

- }

- justify-content horizontal alignment

- align-items vertical alignment

* child flex items

- items {

- flex-grow: 1;

- }

flexbox is powerful for responsive layouts, navigation bars, card layouts & more.

7 Inheritance

CSS properties sometimes inherit from parent elements.

for eg. font families, color.

But other properties don't inherit (eg margin).

Understanding what does inherit helps avoid repetitive code.

8. CSS units

Units	Type	Use Case
Px	fixed	Precise control
%	Relative	Parent based layout
em, rem	font-relative	Scalable typography
vh, vw	Viewport	full-screen elements

Notes rem is relative to the root font size - useful for consistent scaling across

9 CSS shorthand Properties

CSS allows shorter syntax for multiple related properties.

margin: 10px 20px 30px 20px;

Shorthand saves code & reduces mistake.

10 Best Practices & Debugging

- ✓ Organise style with comments
- ✓ Group related rules together
- ✓ use consistent naming (Classes & IDs)
- ✓ Test styles in browsers dev tools
- ✓ Avoid !important unless absolutely necessary

11 Common Pitfalls (& How to avoid them)

X Unexpected Overlapping

often caused by position: absolute without parent context

fix! Set parent position: relative

X Elements Overflowing Container

Use:

overflow: auto;

X Text Too small on middle

Use:

font-size: 1em;

instead of fixed px values.

Summary (what intermediate CSS gives you)

- Precisely CSS target elements
- Control layout structure
- Understand space & flow
- Build responsive modern interfaces
- Produce clean, maintainable styles