The Beginner's Guide to CSS

Steven Mcananey

The Beginner’s Guide to CSS

Steven Mcananey

Published by Steven Mcananey, 2025.

THE BEGINNER’S GUIDE TO CSS

First edition. June 28, 2025.

ISBN: 979-8231263608

Written by Steven Mcananey.

TABLE OF CONTENTS

CHAPTER 1: WHAT IS CSS AND WHY IT MATTERS

What is CSS?

CSS stands for **Cascading Style Sheets**. If HTML is the **skeleton** of a webpage — providing structure like headings, paragraphs, and images — then CSS is the **clothing and makeup**. It's what makes your website **look good**.

Without CSS, every website would look like a plain Word document: black text, white background, unstyled links, and no layout. CSS brings the visual magic — colors, fonts, spacing, alignment, layout, and animations.

A Quick Example

Here's a simple HTML page: html

<!DOCTYPE html> <html>

<head>

<title>My First Page</title> </head>

<body>

<h1>Hello, world!</h1> This is my first website. </body>

</html>

Now, let's style it with CSS: html

<head>

<style>

body {

background-color: #f0f0f0; font-family: Arial, sans-serif; color: #333;

text-align: center;

}

h1 {

color: #0066cc;

}

</style>

</head>

With just a few lines of CSS, we changed the entire **feel** of the page.

How CSS Works with HTML

CSS and HTML work together — HTML **describes** the content, and CSS **styles** it.

Let’s compare the two roles with a metaphor:

HTML	CSS
A cake recipe	The decorations on the cake
The structure of a house	The paint, furniture, and wallpaper
The ingredients list	The way the dish is plated

They’re both important — but they do different things.

Types of CSS: Inline, Internal, External

There are three main ways to write CSS. Let’s go over each one:

1. Inline CSS

You can write CSS directly inside an HTML element using the style attribute:
html

CopyEdit

This is red text. **Pros**:

- Quick and easy for small changes **Cons**:
- Messy and hard to maintain for bigger projects

---

2. Internal CSS

You can put CSS inside a <style> tag within the <head> of your HTML: html

CopyEdit

<head>

<style>

p {

color: blue;

}

</style>

</head>

Pros:

- Good for small pages or single files **Cons**:
- Not reusable across multiple pages

---

3. External CSS (Most Common)

You write your CSS in a separate .css file and link it to your HTML: style.css
css

CopyEdit

p {

color: green;

}

index.html html

<head>

<link rel="stylesheet" href="style.css"> </head>

Pros:

- Clean, organized, and reusable **Cons**:
- Requires managing multiple files (but it’s worth it!)

💡 What Does “Cascading” Mean?

The “Cascading” in CSS refers to how **rules are applied in order of importance**.

Imagine three people giving instructions to style a button:

1. A default browser rule says: "Make all buttons gray."
2. Your external CSS says: "No, make buttons blue."
3. An inline style says: "Actually, make this button red."

The browser will use the **most specific rule**, so the button becomes **red**.

This “cascade” lets you **override** less specific rules with more targeted ones.

Setting Up Your First CSS File

Let’s build a real example. Follow these steps:

1. Create a folder

Make a new folder called my-first-website.

2. Create your HTML file

Create a file named index.html and add this code: html

CopyEdit

<!DOCTYPE html> <html>

<head>

<title>My First Styled Page</title> <link rel="stylesheet" href="style.css"> </head>

<body>

<h1>Welcome!</h1> This page uses CSS. </body>

</html>

3. Create your CSS file

In the same folder, create a file named style.css and add: css

CopyEdit

body {

background-color: #e0f7fa; font-family: Georgia, serif; text-align: center;

```
}

h1 {

color: #00796b;

}

p {

color: #004d40;

}
```

4. Open your HTML file in a browser

Double-click index.html. You'll see your styled webpage!

✅ Key Takeaways

- **CSS controls how your webpage looks**, while HTML controls the content and structure.

- You can write CSS **inline**, **internally**, or **externally**.

- **External CSS** is the cleanest and most reusable method.

● CSS is called “Cascading” because it applies rules in a specific order of priority.

● You can start using CSS with just a text editor and a web browser!

Quick Practice

Try changing the background color in your CSS file to something else (like lightpink). Save and refresh the page. What changes?

Coming Up Next...

In **Chapter 2**, we’ll break down the **syntax of CSS** and how to use **selectors** to target specific parts of your webpage. You’ll learn how to apply styles to elements, classes, and IDs — the building blocks of all CSS styling.

CHAPTER 2: CSS SYNTAX AND SELECTORS

What You'll Learn in This Chapter:

- How CSS is written (its basic grammar)
- How to target different elements using selectors
- The difference between element, class, and ID selectors
- Tips to write clean, readable, and scalable CSS

Let's dive in!

CSS Syntax: The Basic Building Block

CSS is written in **rules**. A CSS rule consists of a **selector** and a **declaration block**.

css

CopyEdit

selector {

property: value;

}

Let's break that down:

Example:

css

CopyEdit

p {

color: blue;

font-size: 16px;

}

Part	Description
p	The **selector** (targets all <p> tags)
color and font-size	The **properties**
blue and 16px	The **values**
{}	The **declaration block**

You can include multiple properties inside one block — just separate them with a semicolon ;.

What Are Selectors?

Selectors tell the browser **what part of the page** you want to style.

There are several kinds, but here are the three most important ones for beginners:

1. Element Selector

This targets all HTML elements of a specific type.

css

CopyEdit

h1 {

color: purple;

}

☞ This changes **all <h1> headings** on the page to purple.

---

2. Class Selector

This targets any element with a **class** attribute. Use a **dot (.)** before the class name.

HTML: html

CopyEdit

This is special text. CSS: css

CopyEdit

.highlight {

background-color: yellow; }

☞ This styles only elements with class="highlight".

✅ You can **reuse classes** on multiple elements.

---

3. ID Selector

This targets one unique element using its **ID**. Use a **hash (#)** before the ID name.

HTML: html

CopyEdit

Welcome to my website. CSS: css

CopyEdit

#intro {

font-style: italic; }

⚠ IDs should be **unique** — only used once per page.

Comparing Element vs Class vs ID

Selector Type	Syntax	Target Scope
Element	p	All tags

Class	.my-class	All elements with that class
ID	#my-id	One specific element

Combining Selectors

You can make your CSS more powerful by combining selectors.

**Example 1: Target all <p> elements inside a **

css

CopyEdit

div p {

color: green;

}

☞ Affects only <p> tags that are **inside** a .

Example 2: Style multiple elements at once

css

CopyEdit

h1, h2, h3 {

font-family: "Verdana", sans-serif; }

☞ All three heading levels will have the same font.

CSS Best Practices (Start Good Habits Early!)

1. **Use classes** for reusable styling (instead of inline or IDs).

2. **Be descriptive** with class names:
   - ✖ Bad: .a1, .blue
   - ✓ Good: .main-button, .error-message

3. **Group similar styles** together to stay organized.

4. **Keep your CSS in a separate file** if your project has more than one page.

Real-Life Practice

Here’s a small exercise to try out what you've just learned.

Step 1: Create an HTML file:

html

CopyEdit

<!DOCTYPE html> <html>

<head>

<link rel="stylesheet" href="style.css"> <title>CSS Selectors Practice</title> </head>

<body>

<h1 id="main-title">Welcome!</h1> This text is highlighted. This text is normal. </body>

</html>

Step 2: Create a CSS file (style.css):

css

CopyEdit

/* Element selector */

p {

font-family: Arial; color: #333;

}

/* Class selector */

.highlight {

background-color: yellow; font-weight: bold;

}

/* ID selector */

#main-title {

color: navy;

font-size: 36px;

}

💡 Try adding another <p class="highlight"> to see how CSS reuses that class style!

✅ Key Takeaways

- CSS rules are made of **selectors** and **declaration blocks**.
- Use **element selectors** for general styling.
- Use **class selectors** for reusable styles across multiple elements.
- Use **ID selectors** for unique elements only.
- Combine selectors to target elements more specifically and efficiently.

Coming Up Next...

In **Chapter 3**, we'll take a deep dive into **styling text** — fonts, colors, alignment, spacing, and how to make your typography look amazing across

devices.

CHAPTER 3: STYLING TEXT AND FONTS

What You'll Learn in This Chapter:

- How to change fonts, sizes, and text color
- How to align, space, and decorate text
- How to style links
- How to use web fonts like **Google Fonts**
- Tips for creating readable, attractive typography

Let's bring your text to life!

Why Typography Matters

Text is the **heart** of most websites. Whether it's a blog, product page, or portfolio, clear and attractive text makes your site look professional and readable.

CSS gives you full control over how your text **looks and feels**.

Basic Text Properties

Let's look at the most commonly used CSS properties for text.

1. color

Sets the color of the text.

css

CopyEdit

p {

color: darkslategray; }

✅ You can use:

- Color names (like red, blue)
- HEX codes (like #ff6600)
- RGB values (like rgb(0, 128, 255))
- HSL (like hsl(200, 100%, 40%))

---

2. font-family

Sets the typeface (font) for your text.

css

CopyEdit

body {

font-family: Arial, sans-serif; }

Always use a **fallback font** like this: css

CopyEdit

font-family: 'Georgia', serif; There are five basic **font families**:

- **Serif** (e.g. Times New Roman)
- **Sans-serif** (e.g. Arial)
- **Monospace** (e.g. Courier New)
- **Cursive** (e.g. Comic Sans MS)
- **Fantasy** (rarely used)

---

3. font-size

Changes the size of text.

css

CopyEdit

h1 {

font-size: 32px;

}

✅ Units you can use:

- px (pixels – fixed size)
- em (relative to parent size)
- rem (relative to root <html> size)
- % (relative to parent)

Example:

css

CopyEdit

p {

font-size: 1.2rem;

}

---

4. font-weight

Controls how bold the text is.

css

CopyEdit

h1 {

font-weight: bold; }

p {

font-weight: 300; /* Lighter */

}

Common values:

- normal
- bold
- Numerical: 100 (thin) to 900 (thick)

---

5. text-align

Controls horizontal alignment of text.

css

CopyEdit

p {

text-align: center; }

Other values:

- left (default)
- right
- justify (aligns to both sides)

---

6. line-height

Controls spacing between lines.

css

CopyEdit

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

Use this to make text easier to read, especially for long paragraphs.

✨ Styling Links

Links are styled with **pseudo-classes** — these let you style based on interaction.

css

CopyEdit

```
a {
color: blue;
text-decoration: none; }
a:hover {
color: orange;
text-decoration: underline; }
```

Pseudo-Class	Description
:hover	When mouse is over the link
:visited	After the link has been clicked
:active	When the link is clicked (mouse held down)

🆕 Using Google Fonts

Default fonts can be boring. **Google Fonts** offers hundreds of free, stylish fonts you can add easily.

Step-by-step:

1. Go to:https://fonts.google.com
2. Choose a font (e.g., *Roboto*)
3. Click "Get embed code"
4. Copy the link and paste it in your <head>:

html

CopyEdit

<link href="https://fonts.googleapis.com/css2?family=Roboto&display=swap" rel="stylesheet">

1. Then apply the font in CSS:

css

CopyEdit

body {

font-family: 'Roboto', sans-serif; }

Real-Life Practice: Style a Paragraph

HTML:

html

CopyEdit

Welcome to my stylish website! CSS: css

CopyEdit

.lead {

color: #2c3e50;

font-family: 'Georgia', serif; font-size: 20px;

line-height: 1.6;

text-align: justify; }

You just combined multiple properties to make the text more attractive and readable!

✅ Key Takeaways

- Use color, font-family, font-size, and line-height to shape your typography.

- Use text-align to center or justify your content.

- Style links with a:hover for nice interactions.

- Google Fonts is a great way to upgrade your website’s look.

- Always test your font choices for readability on different devices.

Coming Up Next...

In **Chapter 4**, we'll dive into **colors, backgrounds, and borders** — and start making your site feel like a real design, not just styled text. Get ready for gradients, images, and box decoration!

CHAPTER 4: COLORS, BACKGROUNDS, AND BORDERS

What You'll Learn in This Chapter:

- How to apply colors using different formats (name, HEX, RGB, HSL)
- How to set background colors, images, and gradients
- How to add borders and customize them
- How to use rounded corners and shadows to add depth

Let's transform your site from flat to fabulous!

Applying Colors in CSS

Color is one of the most powerful tools in design. CSS gives you **four** main ways to apply colors to text, backgrounds, borders, and more:

---

1. Color Names

Simple and intuitive — great for basic styling.

```
css

CopyEdit

h1 {

color: red;

}
```

There are over 140 standard color names like blue, gray, darkgreen, coral, and more.

Tip: Try exploring w3schools color names to see them all.

---

2. HEX Codes

The most common method — great for precise colors.

css

CopyEdit

```
p {
color: #3498db;
}
```

- A HEX code begins with # followed by 6 characters (0–9, A–F).
- Example: #ffffff is white, #000000 is black.

Shortcut: If all 6 characters are repeated (e.g., #ffcc00), you can write it as #fc0.

---

3. RGB Values

Defines color by mixing **Red, Green, Blue** values (0–255).

css

CopyEdit

```
div {
color: rgb(255, 99, 71); /* Tomato */
}
```

- rgb(0, 0, 0) = black
- rgb(255, 255, 255) = white
- You can also add **transparency** using rgba():

css

CopyEdit

```
background-color: rgba(0, 0, 0, 0.5); /* Semi-transparent black */
```

---

4. HSL Values

HSL = Hue, Saturation, Lightness. It’s more human-friendly!

css

CopyEdit

body {

color: hsl(200, 100%, 40%); }

- Hue: 0–360 (color on the color wheel)
- Saturation: 0–100% (intensity)
- Lightness: 0–100% (brightness)

You can also use hsla() for transparency.

Setting Backgrounds

1. Background Color

css

CopyEdit

body {

background-color: #f9f9f9; }

This sets the page’s background to a light gray tone.

---

2. Background Image

css

CopyEdit

body {

background-image: url('images/background.jpg'); background-size: cover; background-repeat: no-repeat; background-position: center; }

Property	What it does
background-size: cover;	Makes the image fill the screen
background-repeat: no-repeat;	Prevents tiling
background-position: center;	Centers the image

⚠ Always test background images on mobile screens — they can shrink or crop unexpectedly.

---

3. Background Gradient

Gradients let you fade between two or more colors — without using an image.

css

CopyEdit

header {

background: linear-gradient(to right, #00c6ff, #0072ff); }

Other directions:

- to bottom
- to left
- to top right, etc.

You can also do **radial gradients**: css

CopyEdit

div {

background: radial-gradient(circle, #f06, white); }

Adding Borders

Borders help define sections and highlight content.

css

CopyEdit

.box {

border: 2px solid black; }

The **border shorthand** format is: css

CopyEdit

border: [width] [style] [color]; Examples:

css

CopyEdit

border: 1px dashed gray; border: 3px double #ff6600;

Border Styles:

- solid
- dashed
- dotted
- double
- groove, ridge, inset, outset

Border Sides

You can apply borders to specific sides: css

CopyEdit

border-top: 2px solid red; border-bottom: 1px dotted #ccc; This gives you more control over layout and design.

Rounded Corners with border-radius

Smooth out boxy edges and make your design look more modern.

css

CopyEdit

.card {

border-radius: 10px; }

You can apply it to just one corner: css

CopyEdit

border-top-left-radius: 10px; 💡 Want a perfect circle? Make the radius 50% on a square element: css

CopyEdit

.profile-pic {

border-radius: 50%; }

💡 Adding Depth with Shadows

1. Box Shadow

Gives a shadow around a box or element.

css

CopyEdit

.container {

box-shadow: 2px 2px 8px rgba(0,0,0,0.2); }

Format: box-shadow: x-offset y-offset blur color; More advanced:

css

CopyEdit

box-shadow: 0 4px 12px rgba(0,0,0,0.15);

2. Text Shadow

Adds a shadow to text: css

CopyEdit

h1 {

text-shadow: 1px 1px 3px rgba(0,0,0,0.3); }

Use shadows carefully — too many can make your design look heavy or blurry.

Real-Life Example: Styling a Card

HTML:

html

CopyEdit

 <h2>Welcome</h2> This box uses colors, borders, and shadows!

CSS: css

CopyEdit

.card {

background-color: #ffffff; border: 1px solid #ddd; border-radius: 8px; box-shadow: 0 2px 6px rgba(0,0,0,0.1); padding: 20px;

width: 300px;

margin: 20px auto; text-align: center; }

.card h2 {

color: #2c3e50;

}

Result: A clean, modern-looking content box.

✅ Key Takeaways

- CSS supports color with names, HEX, RGB, and HSL — use what works for you
- Backgrounds can be solid colors, images, or gradients
- Borders can be fully customized on all or individual sides
- border-radius and box-shadow add softness and depth to your layout
- Good use of these techniques can make your design look polished and professional

---

Coming Up Next...

In **Chapter 5**, we'll tackle **layouts** using the CSS **Box Model**, **Flexbox**, and **Grid** — powerful tools that let you control how your content is arranged on the page.

CHAPTER 5: LAYOUTS WITH BOX MODEL, FLEXBOX, AND GRID

What You’ll Learn in This Chapter:

- How the **CSS Box Model** works: content, padding, border, margin
- What **display types** are and when to use block, inline, and inline-block
- How to use **Flexbox** to align and distribute elements
- How to use **CSS Grid** to create advanced layouts easily

Web design isn’t just about colors and fonts — it’s also about **how things are arranged** on the page. In this chapter, you'll learn how to control layout in CSS so your content looks great on any screen.

The CSS Box Model

Every HTML element is a **box** — even text! The **box model** describes how space is structured around elements.

The 4 Layers of the Box Model

lua

CopyEdit

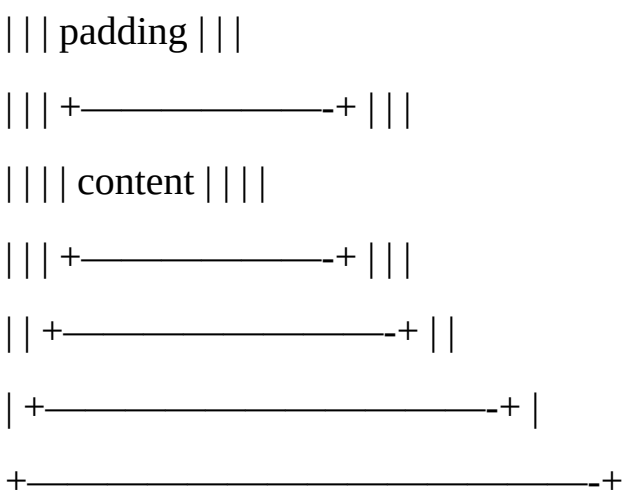
+———————————————————-+

| margin |

| +———————————————-+ |

| | border | |

| | +————————————-+ | |



Part	What it does
Content	The actual text, image, or element
Padding	Space **inside** the border, around the content
Border	A line around the padding (and content)
Margin	Space **outside** the border — separates elements

Example:

css

CopyEdit

.box {

margin: 20px;

padding: 10px;

border: 2px solid black; }

This creates 10px of space **inside** the border (padding), and 20px of space **outside** the border (margin).

Display Types: block, inline, and inline-block

By default, HTML elements behave differently on the page. CSS lets you control this using the display property.

---

1. block

- Takes up the **full width** of the container
- Starts on a **new line**

Examples: , <h1>, <p> css

CopyEdit

div {

display: block;

}

---

2. inline

- Only takes up as much **width** as necessary
- Sits **beside other elements**
- You **can’t set width/height/padding/margin** (only horizontally)

Examples: , <a>, css

CopyEdit

span {

display: inline;

}

---

3. inline-block

- Like inline (sits in line), but **you can set width, height, padding, etc.**
- Great for buttons and nav links

css

CopyEdit

button {

display: inline-block; padding: 10px 20px;

}

Flexbox: Layout Made Flexible

Flexbox is a modern CSS layout tool that makes it easy to align and space elements **in a row or column**.

How to Use Flexbox

First, turn a container into a **flex container**: css

CopyEdit

.container {

display: flex;

}

Then you can control the layout of the child elements (flex items).

Common Flexbox Properties

On the Container:

css

CopyEdit

display: flex;

flex-direction: row | column; justify-content: flex-start | center | space-between | space-around; align-items: stretch | center | flex-start | flex-end;

- flex-direction: row (default) — lays items horizontally

- flex-direction: column — stacks items vertically
- justify-content — controls **horizontal** alignment
- align-items — controls **vertical** alignment

On the Items:

css

CopyEdit

flex: 1;

align-self: center;

order: 2;

- flex — controls how much space the item takes
- order — changes the display order of items

Example: Horizontal Card Layout

HTML:

html

CopyEdit

 Card 1 Card 2 Card 3

CSS: css

CopyEdit

.cards {

display: flex;

justify-content: space-between; gap: 20px;

}

.card {

background: #eee;

padding: 20px;

flex: 1;

}

This creates **3 equal cards** spaced evenly across the row.

CSS Grid: Layout by Rows and Columns

CSS Grid is the most powerful layout system in CSS. It lets you create **two-dimensional** layouts — rows *and* columns.

Step 1: Make a Grid Container

css

CopyEdit

.grid {

display: grid;

grid-template-columns: repeat(3, 1fr); gap: 20px;

}

- display: grid — activates Grid
- grid-template-columns — creates 3 equal columns
- 1fr means “one fraction of the space”
- gap adds space between items

Step 2: Add Grid Items

HTML:

html

CopyEdit

 1 2 3 4

This will automatically fill the first row with 3 items, and place the 4th item on the next row.

Advanced Grid Options

You can control **exact placement**: css

CopyEdit

.item1 {

grid-column: 1 / 3; /* spans columns 1 and 2 */

grid-row: 1 / 2;

}

You can even make **asymmetric layouts** easily — like magazine pages or dashboards.

Practice: Simple 2-Column Layout

HTML:

html

CopyEdit

 Sidebar Main Content

CSS: css

CopyEdit

.grid {

display: grid;

grid-template-columns: 1fr 3fr; gap: 20px;

}

.sidebar {

background: #ccc;

}

.content {

background: #eee;

}

1fr 3fr means the main content is **3x wider** than the sidebar.

✅ Key Takeaways

- The **Box Model** explains how every element has content, padding, border, and margin
- Use display to control how elements behave (block vs inline)
- **Flexbox** is great for 1D layouts (rows or columns)
- **Grid** is perfect for complex 2D layouts with rows *and* columns
- Use gap, padding, and margin to space things cleanly

Coming Up Next...

In **Chapter 6**, we'll explore **Responsive Design Basics** — how to make your site look amazing on phones, tablets, and desktops. You'll learn about media queries, relative units, and mobile-first design.

CHAPTER 6: RESPONSIVE DESIGN BASICS

What You'll Learn in This Chapter:

- What **responsive design** is and why it's essential
- How to use **media queries** to adapt layouts for different screen sizes
- How to use **relative units** like %, em, rem, vh, and vw
- Tips for making your website **mobile-friendly**

Your website shouldn't just *look* good — it should look good **everywhere**.

What Is Responsive Design?

Responsive design means your website **adapts** to different screen sizes and devices — desktop, tablet, mobile, even smart TVs.

Without Responsive Design:

- Text is too small on phones
- Images go off-screen
- Users have to zoom and scroll awkwardly

With Responsive Design:

- Content adjusts to screen width
- Font sizes scale naturally
- Layouts stack and reflow beautifully

In today's world, where over **50% of web traffic** comes from mobile devices, responsive design is no longer optional — it's essential.

How Responsive Design Works

Responsive websites use a mix of:

1. **Flexible layouts** (using relative units like %, rem, etc.)
2. **Media queries** (to apply different styles at different screen widths)
3. **Mobile-first thinking** (designing for small screens first, then scaling up)

Let's break this down.

Relative Units in CSS

Instead of using fixed units like px (pixels), responsive design uses **relative units**. These adjust based on screen size, user preferences, and parent containers.

% (Percentage)

Used for widths and heights based on the **parent** element.

css

CopyEdit

.container {

width: 100%;

}

This means: "Take up all available space."

---

em and rem

Used mostly for **font sizing and spacing**.

Unit	Relative To
em	The font-size of the **parent** element
rem	The font-size of the **root (<html>)** element

Examples: css

CopyEdit

p {

font-size: 1.2rem; /* 120% of root font size */

padding: 2em; /* 2x the font size of the parent */

}

✅ rem is usually more predictable for consistent scaling.

---

vw and vh

Unit	What it means
vw	1% of the viewport's **width**
vh	1% of the viewport's **height**

css

CopyEdit

.hero {

height: 100vh; /* Full screen height */

font-size: 5vw; /* Scales based on screen width */

}

These units are great for full-screen sections and scalable text.

Media Queries: The Heart of Responsive Design

Media queries let you write CSS that only applies under certain conditions — like screen size.

Basic Syntax:

css

CopyEdit

```
@media (max-width: 768px) {
body {
background-color: lightblue; }
}
```

This code says:

“If the screen is **768 pixels wide or less**, apply this style.”

Common Breakpoints:

These are popular screen size cutoffs:

Device Type	Breakpoint
Mobile	max-width: 480px
Tablet	max-width: 768px
Desktop	min-width: 1024px

Example: Responsive Layout

HTML:

html

CopyEdit

```
<div class="wrapper"> <div class="sidebar">Sidebar</div> <div class="content">Main Content</div> </div>
```

CSS: css

CopyEdit

```
.wrapper {
```

```
display: flex;
}
.sidebar {
width: 25%;
}
.content {
width: 75%;
}
@media (max-width: 768px) {
.wrapper {
flex-direction: column; }
.sidebar,
.content {
width: 100%;
}
}
```

On larger screens, content is side-by-side.

On smaller screens, it **stacks vertically**.

Responsive Images

Large images on small screens can cause layout issues.

Use the following:

css

CopyEdit

```
img {
max-width: 100%;
height: auto;
}
```

✅ This keeps images within their container and scales them appropriately.

Mobile-Friendly Design Tips

1. **Use readable font sizes** (1rem or larger for body text)
2. **Avoid fixed widths** (width: 300px) — use % or vw instead
3. **Stack content** vertically on small screens
4. **Use media queries early** and test on real devices
5. **Leave room to tap!** Make buttons at least 44px high

Real-World Example

Responsive Card Grid

HTML: html

```
CopyEdit
<div class="grid"> <div class="card">Card 1</div> <div class="card">Card 2</div> <div class="card">Card 3</div> </div>
```

CSS: css

```
CopyEdit
.grid {
display: grid;
grid-template-columns: repeat(3, 1fr); gap: 20px;
}
```

```
@media (max-width: 768px) {
.grid {
grid-template-columns: 1fr; }
}
.card {
background: #fff;
padding: 20px;
border: 1px solid #ddd; }
```

✅ On desktop: 3 cards side-by-side On mobile: 1 card per row

✅ Key Takeaways

- **Responsive design** ensures your website works on all devices
- Use **relative units** (%, rem, vw) instead of fixed pixels
- **Media queries** allow CSS to adapt to screen sizes
- Start **mobile-first** — then scale up
- Test your site regularly on phones, tablets, and desktops

Coming Up Next...

In **Chapter 7**, we'll bring everything together in a **final mini project**: building and styling your own responsive portfolio page using everything you've learned — structure, color, layout, fonts, and media queries.

CHAPTER 7: BRINGING IT ALL TOGETHER

What You'll Learn in This Chapter:

- How to **style a complete webpage** step-by-step
- Common CSS **mistakes beginners make** and how to avoid them
- Handy **tools and resources** to keep learning
- A simple but powerful **final project**: your first **responsive portfolio page**

Let's wrap up everything you've learned and put it into practice!

Step-by-Step: Styling a Complete Web Page

Let's walk through the process of building and styling a real page from scratch — combining **HTML structure** and **CSS styling**.

---

1. Plan Your Page Structure

Before writing any code, sketch out your page layout. For a basic portfolio page, you might have:

- A **Header** with your name
- A **Navigation** bar
- An **About Me** section
- A **Projects** section

- A **Contact** section
- A **Footer**

---

2. Basic HTML Layout

html

CopyEdit

<!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> <link rel="stylesheet" href="style.css"> <link href="https://fonts.googleapis.com/css2?family=Poppins&display=swap" rel="stylesheet"> </head>

<body>

<header>

<h1>Steven Mcananey</h1> <nav>

About Projects Contact </nav>

</header>

<section id="about"> <h2>About Me</h2> I’m a beginner web developer learning CSS! </section>

<section id="projects"> <h2>My Projects</h2> Project 1 Project 2 Project 3

</section>

<section id="contact"> <h2>Contact Me</h2> Email: hello@example.com </section>

<footer>

© 2025 Steven Mcananey </footer>

</body>

</html>

---

3. Styling It with CSS

style.css

css

CopyEdit

/* Global Styles */

body {

font-family: 'Poppins', sans-serif; margin: 0;

```
padding: 0;
line-height: 1.6;
background-color: #f5f5f5; color: #333;
}
header {
background-color: #007acc; color: white;
padding: 20px;
text-align: center; }
nav a {
color: white;
text-decoration: none; margin: 0 15px;
font-weight: bold;
}
nav a:hover {
text-decoration: underline; }
/* Section Styling */
```

```
section {
padding: 40px 20px; max-width: 800px;
margin: auto;
}
h2 {
color: #007acc;
}
/* Projects Grid */
.project-grid {
display: grid;
grid-template-columns: repeat(auto-fit, minmax(220px, 1fr)); gap: 20px;
margin-top: 20px;
}
.project-card {
background: white;
padding: 20px;
```

```
border: 1px solid #ddd; border-radius: 8px; box-shadow: 0 2px 6px rgba(0,0,0,0.05); text-align: center; }

/* Footer */

footer {

background-color: #222; color: #aaa;

text-align: center; padding: 20px;

font-size: 14px;

}

/* Responsive Styling */

@media (max-width: 600px) {

header, nav {

text-align: center; }

nav a {

display: block;

margin: 10px 0;

}
```

}

Done! You now have a responsive, styled personal portfolio page.

✖ Common Mistakes Beginners Make (And How to Avoid Them)

1. Using Fixed Widths Everywhere

- ✖ width: 800px
- ✅ width: 80% or max-width: 800px

Why: Fixed widths break your layout on smaller screens.

---

2. Forgetting to Set box-sizing: border-box

css

CopyEdit

* {

box-sizing: border-box; }

Why: This makes padding and borders easier to manage in layouts.

---

3. Overusing IDs for Styling

- ✖ #main { ... }

- ✅ .main { ... }

Why: IDs are specific and hard to reuse; classes are flexible.

---

4. Not Testing on Mobile Devices

Tip: Use Chrome DevTools (F12 → Toggle Device Toolbar) to test screen sizes.

---

5. Using Too Many Fonts or Colors

Keep it simple and consistent:

- Max **2 fonts**
- A **main color**, **accent color**, and **neutral base**

Helpful Tools and Resources

Code & Design Tools

- CodePen.io — Test and share live HTML/CSS
- Google Fonts — Free, web-safe fonts
- Color Hunt — Beautiful color palettes
- Coolors — Generate color schemes

Learn More

- MDN Web Docs — In-depth reference
- CSS Tricks — Articles and examples
- Flexbox Froggy — Learn Flexbox by playing a game!
- Grid Garden — Game to learn CSS Grid

Final Project: Build Your Own Portfolio

Let's put it all together.

✅ **What to include:**

- Your name, bio, and skills
- 2–3 project cards with brief descriptions
- A working navigation that scrolls to each section
- A responsive layout that adapts to phone screens
- Use Flexbox or Grid for layout and Google Fonts for style

File structure:

bash

CopyEdit

/portfolio-project

│

├── index.html

├── style.css

└── /images (optional)

✅ Key Takeaways from the Entire Book

- CSS gives you full control over how a website **looks and behaves**
- Start with a clear **HTML structure**, then apply **consistent styling**
- Learn to use the **Box Model**, **Flexbox**, and **Grid** for layout
- Write **clean, mobile-friendly CSS** using relative units and media queries
- Use external resources and keep **practicing** small projects to improve

---

Congratulations!

You've reached the end of *The Beginner's Guide to CSS* — and you've gained the skills to **build and style real websites**.

This isn't the end of your journey — it's the beginning. From here, you can explore animations, transitions, SCSS, frameworks like Tailwind or Bootstrap, and even JavaScript to add interactivity.

Keep learning, keep experimenting, and most of all — keep building.