**🎨 CSS : Introduction to CSS**

🧠 **By TNT Ethiopia** — *Teaching tech the easy way* 🇪🇹

**📘 1.1 What is CSS (Cascading Style Sheets)?**

CSS stands for **Cascading Style Sheets**, and it’s the language we use to **style and design websites**.

When you build a webpage using HTML, you're creating the **structure** (like the bones of a body). But without CSS, that structure looks **plain, boring, and lifeless**.

CSS brings your HTML to life! It lets you add:

* 🖍️ **Colors** (text, background, borders, etc.)
* 🔠 **Fonts** (typeface, size, bold/italic styles)
* 📏 **Spacing** (margins, padding, line height)
* 📐 **Layout and Positioning** (where things appear on the page)
* 🎨 **Backgrounds** (images, gradients, colors)
* 📱 **Responsive designs** for different devices (like mobile or tablet)

In simple words:

**CSS is a list of styling instructions that tells the browser how to display each HTML element.**

For example, with CSS, you can tell the browser:

* What **color** a button should be (background-color: blue;)
* How **big** the heading should be (font-size: 36px;)
* How much **space** should be between two sections (margin-top: 40px;)
* Whether a box should be aligned **left**, **right**, or **center**

CSS is **not** something that works alone — it **needs HTML**. HTML creates the elements like buttons, headings, paragraphs, and images. CSS is applied **on top** of these elements to make them look great.

📝 **Example:**

<h1>Hello, World!</h1>

h1 {

color: green;

font-size: 40px;

}

The above CSS tells the browser:

"Hey, when you display that <h1> heading, make the text green and make it 40 pixels big."

**🌊 What Does "Cascading" Mean in CSS?**

The word **"cascading"** in CSS comes from the idea of a **waterfall** — styles **flow down** from a parent element to its children.

Let’s take this simple example:

body {

color: blue;

}

This rule says:

"Make all text inside the body tag blue."

Now, if the body tag contains headings, paragraphs, and lists — **they all become blue by default**, unless we specifically change their color.

This is what **cascading** means:

* Styles are **inherited** by child elements.
* The browser starts at the top (parent), and flows styles downward (children).
* You can **override** inherited styles by writing more specific rules.

This behavior helps us avoid repeating the same style many times — which makes CSS more powerful and efficient.

**🧠 1.2 Understanding CSS (Versions + Styling Instructions)**

**📅 CSS Versions (CSS1, CSS2, CSS3)**

CSS has gone through **several versions**, just like any other technology. Here's a quick overview:

| **Version** | **Summary** |
| --- | --- |
| **CSS1** | The very first version. Introduced basic text styles. |
| **CSS2** | Added better layout options like positioning and z-index. |
| **CSS3** | The current and most powerful version. Adds cool things like transitions, animations, media queries, shadows, gradients, flexbox, grid, and more. |

➡️ **You should focus only on CSS3**. It includes everything from CSS1 and CSS2, and is what modern websites use.

**🛠️ How Does CSS Work with the Browser?**

Your browser (like Chrome, Firefox, or Safari) is like a **constructor or builder**.  
HTML gives the **structure**, and CSS gives the **instructions** to decorate that structure.

But the browser is not a mind reader! You need to send instructions using **valid CSS rules** — just like giving directions in a language the builder understands.

You’re sending written styling rules, and the browser reads and **renders** (displays) the design.

🔑 Key Points:

* CSS is the **language of design** for the browser.
* It tells the browser:  
  → What color to use  
  → Where things go  
  → How things behave
* The browser uses this info to display the site the way you want.

**🧪 Real-World Analogy:**

Think of HTML as a **house under construction** — just bricks and rooms.  
CSS is like the **interior designer** who comes in and says:

* "Let’s paint the walls cream-white."
* "The sofa should be centered."
* "Add spacing between the furniture."
* "Change the font on the wall posters."

Without the interior design (CSS), the house (HTML) works but feels **empty and dull**.

**✅ Summary – Key Takeaways**

* **CSS = Styling language for web pages**
* It works **with HTML** to make things look nice
* You can change colors, fonts, spacing, layouts, and much more
* **"Cascading"** means styles flow from parent to child
* The browser is the **builder**, CSS is your **instruction list**
* **CSS3** is the latest and best version — focus on learning it!

**How to Add CSS to Your HTML**

💡 *By TNT Ethiopia – Making tech simple* 🇪🇹

**🧩 Introduction**

To make your webpage look good using CSS, you need to **connect CSS with HTML** — because HTML gives the structure, and CSS gives the style.

But how exactly do we “add” CSS to our HTML?

👉 There are **three ways** to apply CSS styles to HTML pages:

1. Internal CSS
2. Inline CSS
3. External CSS

Let’s go through each one step-by-step 👇

**🧠 1. Internal CSS**

**📍 What is Internal CSS?**

Internal CSS means writing your CSS **inside the HTML file**, specifically in the <head> section using a <style> tag.

This way, the styles are written in the same file as your HTML content, but separated from the actual page content.

**🛠️ Syntax Example:**

<!DOCTYPE html>

<html>

<head>

<style>

body {

background-color: red;

}

h1 {

color: white;

}

</style>

</head>

<body>

<h1>Hello, CSS!</h1>

</body>

</html>

🎯 In this example:

* We used <style> inside <head>
* We changed the background color of the page to **red**
* The heading color is set to **white**

**✅ When to Use Internal CSS:**

* For **small projects** or **single-page** websites
* When you want to **test styles quickly**
* When you don’t want to create multiple files

**🧠 2. Inline CSS**

**📍 What is Inline CSS?**

Inline CSS means writing CSS **directly inside the HTML tag** using the style attribute.

You apply styles to **just one specific element**, not the whole page.

**🛠️ Syntax Example:**

<body style="background-color: blue;">

<h1 style="color: white;">Welcome to Inline CSS</h1>

</body>

🎯 In this example:

* The body has a background color of **blue**
* The h1 text color is **white**
* All styles are inside the style="" attribute of the tag

**⚠️ Drawbacks of Inline CSS:**

* Hard to manage for large websites
* Messy and repetitive
* Not reusable
* Breaks the separation between content and design

**✅ When to Use Inline CSS:**

* For **quick testing**
* When changing style dynamically (e.g., using JavaScript)
* For **small adjustments** on a single element

**🧠 3. External CSS**

**📍 What is External CSS?**

External CSS means creating a **separate file** with all your CSS code and **linking it** to your HTML file using the <link> tag.

This is the **recommended** and most professional way to use CSS.

**📄 Step 1: Create a CSS File**

Create a file named style.css and write your styles there:

/\* style.css \*/

body {

background-color: purple;

}

h1 {

color: yellow;

}

**🧩 Step 2: Link CSS File to HTML**

In your HTML file, use the <link> tag inside the <head>:

<!DOCTYPE html>

<html>

<head>

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

</head>

<body>

<h1>This is External CSS</h1>

</body>

</html>

🎯 In this example:

* CSS code is separated into style.css
* We use <link> to connect it to the HTML file
* All styles are clean, reusable, and organized

**✅ Why External CSS is the Best:**

* Keeps your code **clean and readable**
* Makes styles **reusable across multiple pages**
* Helps with **website performance**
* Easier to **maintain and update**

**🔄 Quick Comparison Table**

| **Method** | **Where it's Written** | **Reusability** | **Best For** |
| --- | --- | --- | --- |
| Inline | Inside the HTML tag | ❌ No | Small tweaks/testing |
| Internal | Inside <style> in <head> | ⚠️ Limited | Single-page websites |
| External | Separate .css file | ✅ Yes | All real projects |

**✅ Final Thoughts for This Section**

* There are **3 ways** to use CSS: Internal, Inline, and External
* **External CSS** is the best and cleanest method
* You should **avoid inline CSS** for large projects
* Always link your CSS file properly using the <link> tag

**1.4 CSS Units**

👉 Having a clear understanding of **measurement units** is the foundation for any design and styling job in CSS.

There are a couple of commonly used units:

* **📏 Pixel (px)**
  + The most common unit in CSS.
  + **1px = 1/96th of an inch (≈ 0.26mm)**
  + You should train your brain to think in terms of pixels from now on.
* **📐 Percentage (%)**
  + Defines sizes **relative to other elements** (usually the parent container).
  + Example: width: 50% makes the element take up half of its parent container’s width.
* **🔠 em and rem (relative units for fonts)**
  + These are usually used for **font sizes** and are **scalable**.

**👉 em (relative to parent element’s font size):**

.parent { font-size: 18px; }

.child { font-size: 2em; }

Here, the child will be **36px** (2 × 18px).

**👉 rem (relative to the root element <html>):**

* + Always based on the **root <html> element’s font-size**.
  + If not set, browsers default to **16px**.

html { font-size: 16px; }

.parent { font-size: 15px; }

.child-rem { font-size: 2rem; }

The child will be **32px** (2 × 16px), ignoring the parent.

**1.5 Thinking in Terms of Containers 🗂️**

The concept of **wrapping elements in containers** is key to mastering CSS.

* Containers are simply **HTML elements** that group related content together.
* Common container tags:
  + <header>
  + <section>
  + <div>
  + <footer>

✨ **Why think in containers?**

* Styling is often applied to containers, not individual elements.
* Organizing your HTML makes it easier to style and maintain.

💡 **Before writing CSS:**

* Organize your HTML elements.
* Group related elements together inside containers.

🔨 **Example Project: “The car Lover’s Page”**

* Web development usually starts with a **design provided by designers**.
* From the design:
  + Think about **conceptual containers** before writing HTML.
  + Build your HTML to match those containers.
* Check the **demo video** in this section for guidance.

**1.6 Naming Containers and Elements with ID & Class 🏷️**

To style specific elements in CSS, we **name HTML elements** using **class** and **id**.

⚡ **General rules when naming:**

* Every container should have a **unique and meaningful name**.
* Names are written in the **opening HTML tag**.

**🧑‍🤝‍🧑 Classes**

* Used when you want to style **multiple elements**.
* Think of them like **family names**.
* An element can have **more than one class**.

✅ Example:

<div class="car-container">Hello</div>

**🙋 IDs**

* Used to style **one specific element**.
* Think of them like a **unique first name**.
* Rules for IDs:
  + No two elements should share the same ID.
  + One element cannot have more than one ID.

✅ Example:

<div id="car-wrapper">Hello</div

**🎨 1.7 Selecting Elements Using ID, Class, and Element Type**

👉 As mentioned earlier, **CSS is just a set of instructions** that the browser follows to render HTML elements.

To write CSS, we must first **select (target)** the element we want to style. This process is called **selecting**.

There are multiple ways to select elements in CSS.  
📚 For a deeper dive: [30 CSS Selectors You Must Memorize](https://code.tutsplus.com/tutorials/the-30-css-selectors-you-must-memorize--net-16048)

**🧑‍🤝‍🧑 Class Selector**

* Used to select elements by their **class name**.
* **Symbol:** . (dot)
* Great for styling **multiple elements** that share the same class.

✅ Example:

.className {

/\* CSS instructions go here \*/

}

.bold-title {

color: red;

font-weight: bold;

}

**🙋 ID Selector**

* Used to select elements by their **unique ID**.
* **Symbol:** # (hash / hashtag)
* IDs are meant for **one specific element only**.

✅ Example:

#id {

/\* CSS instructions go here \*/

}

#header-wrapper {

background-color: black;

}

**🏷️ Type Selector (Element Type Selector)**

* Targets elements by their **HTML tag name** (p, span, div, a, etc.).
* Best for making **broad changes** across the whole site.

✅ Example:

a {

color: red;

}

This makes **all links (<a>) on the site red**.

✨ With these three selectors — **class (.)**, **id (#)**, and **type (tag name)** — you already have the **core tools** to target and style elements effectively! 🚀

**🎯 1.8 Selecting Elements Using Star, Hover, and Descendant Selectors**

In addition to class (.), ID (#), and type (tag) selectors, CSS also provides some **special selectors** that make styling more powerful.

**✨ Star / Wild / Universal Selector**

* Symbol: \*
* Selects **all elements** on the page.
* Useful when you want to apply a style to **every HTML element**.

✅ Example:

\* {

margin: 0;

}

This removes the margin (the space **outside** of an element’s border) from **all elements**.

**🧑‍👩‍👦 Descendant Selector**

* Sometimes, just using a class or element type is **not specific enough**.
* The descendant selector lets you **target elements inside (children) of another element (parent)**.
* Think of it like a **full name**: if two people share the same first name, you add their last name to be specific.

🔑 **Parent & Child concept**

* **Parent** = the outer wrapper element.
* **Child** = the element inside the parent.

✅ Example:

a {

color: red; /\* All links are red \*/

}

div a {

color: white; /\* But links inside a <div> are white \*/

}

Here:

* All links (<a>) = red.
* Links inside a <div> = white.

**🖱️ Hover Selector**

* An **action selector** that activates when you **hover your mouse cursor** over an element.
* Great for interactive effects like changing link colors, adding underlines, or animations.

✅ Example:

a:hover {

color: green;

text-decoration: underline;

}

Now, whenever someone hovers over a link, it turns **green** and becomes **underlined**.