**🧱 What Is a Component?**

A **component** is a self-contained piece of code that combines **HTML**, **CSS**, and optionally **JavaScript**, to build a **reusable UI element**.

Think of components like LEGO blocks — each block (component) serves a purpose, and you can reuse or rearrange them to build bigger things like websites or web apps.

**💡 Why Use Components?**

* **Reusability**: Write once, use everywhere.
* **Organization**: Break down big UIs into smaller, manageable pieces.
* **Maintainability**: Fix or update one component, and the change reflects across all uses.

**🔍 Components in HTML**

When designing components with HTML, ask:

*"What part of the UI am I trying to display repeatedly?"*

In this phase, we aren't pulling live data from a database — we're just focusing on **how the UI looks and behaves** using static HTML as our "data."

**📦 Example: A Button Group Component**

Here’s a simple example that shows **repetition** — a perfect opportunity for a component:

<div class="custom-buttons">

<button>Button 1</button>

<button>Button 2</button>

<button>Button 3</button>

<button>Button 4</button>

</div>

You could turn this into a **ButtonGroup component**, which would:

* Use **HTML** for structure,
* Use **CSS** for styling,
* Use **JavaScript** (if needed) to handle clicks or dynamically add/remove buttons.

**🔁 Component Mentality**

Instead of copying and pasting <button> tags every time, you can think:

* Can I use **JavaScript** to generate the buttons?
* Can I apply the same **CSS class** to style them consistently?
* Can I wrap this into a function or framework-specific component (like React/Vue)?

Great! Here's a **simple live coding demo** idea using **vanilla HTML, CSS, and JavaScript** to teach your students how to build a reusable button group component dynamically.

**✅ Goal: Create a ButtonGroup Component**

We’ll use:

* **HTML** for the base structure
* **CSS** for styling
* **JavaScript** to dynamically generate buttons from an array

**🔧 Final Output Preview**

A group of styled buttons created dynamically using JavaScript.

**🧩 1. HTML (Basic Structure)**

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8" />

<meta name="viewport" content="width=device-width, initial-scale=1.0" />

<title>Button Component Demo</title>

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

</head>

<body>

<h2>Reusable Button Group Component</h2>

<div id="button-group" class="custom-buttons"></div>

<script src="index.js"></script>

</body>

</html>

**🎨 2. CSS (style.css)**

.custom-buttons {

display: flex;

gap: 10px;

margin-top: 20px;

}

.custom-buttons button {

padding: 10px 20px;

border: none;

background-color: royalblue;

color: white;

border-radius: 5px;

cursor: pointer;

transition: background-color 0.3s;

}

.custom-buttons button:hover {

background-color: darkblue;

}

**🧠 3. JavaScript (index.js)**

// This is our dynamic data

const buttonLabels = ['Home', 'About', 'Services', 'Contact'];

function createButtonGroup(buttonTexts, targetId) {

const container = document.getElementById(targetId);

buttonTexts.forEach(text => {

const btn = document.createElement('button');

btn.textContent = text;

container.appendChild(btn);

});

}

// Call the component function

createButtonGroup(buttonLabels, 'button-group');

**🗣️ Teaching Tips:**

* Ask students: “What happens if we want 10 buttons? 100?”
* Show how easy it is to update buttonLabels to scale the UI.
* Later, expand this to use **event listeners**, **React**, or **Vue** as your course progresses.