🧱 Angular 19 – Components 😍

✅ What is a Component in Angular?

A **Component** is like a small building block of your web page.

- Think of a webpage like a house.
- Each room in the house (like a kitchen, bathroom, living room) is a **component**.
- Components help divide the webpage into small, manageable, and reusable parts.

In Angular:

- A component **shows some HTML** (the design),
- **Runs some logic** (written in TypeScript),
- And **styles itself** with CSS.

---

1. What makes up a component?

Every component has 3 main parts:

Part	Purpose	Example
HTML file (`.html`)	Shows what it looks like (the structure)	A template like a form, button, card
TypeScript file (`.ts`)	Handles logic (what happens when you click, type, etc.)	Functions, variables
CSS file (`.css`)	Controls how it looks (colors, spacing, etc.)	Styling

Let's see a simple example:

```
@Component({
  selector: 'app-hello',
  templateUrl: './hello.component.html',
  styleUrls: ['./hello.component.css']
})
export class HelloComponent {
  message = 'Welcome!';
}
```

- `selector`: Like a tag you can use in other components (`<app-hello></app-hello>`)
- `templateUrl`: Points to the HTML
- `styleUrls`: Points to CSS

---

2. How does a component work?

- The component class holds data or Properties and functions or Methods.
- The HTML template uses that data to show on the screen.
- The CSS file styles the component.

```
Example:

export class HelloComponent {
  name = 'John';

  greet() {
    alert('Hello ' + this.name);
  }
}

And in HTML:
<h1>Hello {{ name }}</h1>
<button (click)="greet()">Say Hello</button>
```

- `{{ name }}` is called **interpolation** – it shows the value in the UI.
- `(click)="greet()"` is called **event binding** – it calls a function when clicked.

🛠️ Real-Time Use Cases

Use Case	What it Does
Navbar Component	Shows menu links on every page
Login Form Component	Takes user login input

Product List Component	Shows a list of items (like Amazon products)
Chat Message Component	Shows a single message in a chat app
Modal Component	A popup box to confirm delete or show info
Footer Component	Shows contact info at bottom of the page

These components are reusable—used in many places with different data.

🌟 Best Practices (Simple Tips)

- **One component = One responsibility** – keep it focused
- Reuse components where needed – don't copy-paste code
- Clean up using `ngOnDestroy()` if you're using timers or subscriptions
- Keep component names clear (like `LoginComponent`, `HeaderComponent`)
- Keep CSS inside the component unless it needs to be global

✅ How to Create a Component

📌 Syntax:

```
ng generate component <component-name>

OR the shorthand:
ng g c component-name
```

```
📘 Example:
ng g c header
```

- Create a folder named `header` inside `src/app/`
- Generate 4 files:
  - `header.component.ts` – TypeScript logic
  - `header.component.html` – HTML template
  - `header.component.css` – CSS styles
  - `header.component.spec.ts` – Unit test file
- Automatically **register the component** in `app.module.ts`

🛠️ Options You Can Use

Option	Description

`--skip-tests`	Don't generate the `.spec.ts` file

```
Example:
ng g c footer --skip-tests
```

❌ How to Delete a Component

Angular CLI does not have a direct `delete` command, so you must manually:

🪓 Steps to Delete a Component:

1. **Delete the component folder manually**
2. **Remove component import** from:
    i. `app.component.ts` (or any module where it's declared) and imports section
    ii. Any templates where it's being used (like `<app-header></app-header>`)