# Chapter 2: Standalone Components

In Angular, **components** are the smallest unit of UI. Everything visible in an Angular app — buttons, forms, pages — is composed of one or more components.

Each component consists of:

* A **template** (HTML): What the user sees
* A **class** (TypeScript): How it behaves
* Optional **styles** (CSS or SCSS): How it looks

## Standalone Components (Angular 14+)

Since Angular v14, components can be created as **standalone**, which removes the need for declaring them in an NgModule. As of Angular 17+, standalone components are the **default** and **preferred** way to build Angular applications.

### Benefits of Standalone Components:

* No NgModule boilerplate
* Improved modularity and reusability
* Perfect for lazy loading
* Easier testing
* Simpler architecture for beginners

## Creating Standalone Components with Angular CLI

The Angular CLI helps generate components quickly with various options.

### Basic Command

ng generate component component-name --standalone

Or using the **shortcut**:

ng g c component-name --standalone

## CLI Options for Components

|  |  |
| --- | --- |
| Flag / Option | Description |
| --standalone | Generates a standalone component (no NgModule) |
| --inline-template | Inlines the HTML into the .ts file |
| --inline-style | Inlines CSS into the .ts file |
| --skip-tests | Skips generating the .spec.ts test file |
| --style=scss | Uses SCSS instead of CSS |

### 🔍 CLI Examples

#### ✅ 1. Standalone component (default layout):

ng g c hero --standalone

Creates:

src/app/hero/

├── hero.component.ts

├── hero.component.html

├── hero.component.css

├── hero.component.spec.ts

#### ✅ 2. Inline HTML and CSS:

ng g c alert --standalone --inline-template --inline-style

Or shorter:

ng g c alert -s -t --standalone

Creates just one file:

// alert.component.ts

@Component({

selector: 'app-alert',

standalone: true,

template: `<div>Alert message!</div>`,

styles: [`div { color: red; }`]

})

export class AlertComponent {}

#### ✅ 3. Skip the test file:

ng g c header --standalone --skip-tests

No shortcut for –skip-tests:

Result: No header.component.spec.ts will be generated.

#### ✅ 4. Combine all options:

ng g c toast --standalone -t -s –-skip-tests

This generates a **single file**, inline component with no test.

## Anatomy of a Standalone Component

### hero.component.ts

import { Component } from '@angular/core';

@Component({

selector: 'app-hero',

standalone: true,

templateUrl: './hero.component.html',

styleUrls: ['./hero.component.css']

})

export class HeroComponent {

name = 'Angular Hero';

}

## Bootstrapping Without NgModules

In a standalone app, bootstrapping is done via bootstrapApplication().

### main.ts

import { bootstrapApplication } from '@angular/platform-browser';

import { AppComponent } from './app/app.component';

import { appConfig } from './app/app.config';

bootstrapApplication(AppComponent, appConfig)

.catch(err => console.error(err));

### app.config.ts

import { provideRouter } from '@angular/router';

export const appConfig = {

providers: [

provideRouter([]) // Will configure routes later

]

};

## Passing Data with @Input()

### Parent Template

<app-hero [name]="'Angular Developer'"></app-hero>

### Component

import { Component, Input } from '@angular/core';

@Component({

selector: 'app-hero',

standalone: true,

template: `<h2>Welcome, {{ name }}!</h2>`

})

export class HeroComponent {

@Input() name = '';

}

## Sending Events with @Output()

### Component

import { Component, Output, EventEmitter } from '@angular/core';

@Component({

selector: 'app-button',

standalone: true,

template: `<button (click)="notify()">Click Me</button>`

})

export class ButtonComponent {

@Output() clicked = new EventEmitter<void>();

notify() {

this.clicked.emit();

}

}

### Parent

<app-button (clicked)="onClicked()"></app-button>

## Auto-Generated Test Files

By default, Angular creates a .spec.ts file for each component.

### hero.component.spec.ts

describe('HeroComponent', () => {

beforeEach(async () => {

await TestBed.configureTestingModule({

imports: [HeroComponent] // ✅ Standalone component

}).compileComponents();

});

it('should create', () => {

const fixture = TestBed.createComponent(HeroComponent);

const component = fixture.componentInstance;

expect(component).toBeTruthy();

});

});