What Makes a Standalone Component Different from an Angular Module?

1. Declaration vs Self-contained

NgModule approach:

A component must be declared inside a module's declarations array.

```
@NgModule({
  declarations: [HomeComponent, AboutComponent],
  imports: [CommonModule],
  exports: [HomeComponent]
})
```

Standalone approach:

A component declares itself as standalone, no need for

```
@Component({
  selector: 'app-home',
  standalone: true,
  template: `<h1>Home</h1>`,
  imports: [CommonModule]   //=> component directly manages dependencies
})
```

1. Declaration vs Self-contained

NgModule approach:

A component must be declared inside a module's declarations array.

```
@NgModule({
  declarations: [HomeComponent, AboutComponent],
  imports: [CommonModule],
  exports: [HomeComponent]
})
```

Standalone approach:

A component declares itself as standalone, no need for

```
@Component({
  selector: 'app-home',
  standalone: true,
  template: `<h1>Home</h1>`,
  imports: [CommonModule]   //=> component directly manages dependencies
})
```

2.
Bootstrapping

NgModule: App starts by bootstrapping AppModule.

```
platformBrowserDynamic().bootstrapModule(AppModule);
```

Standalone: App starts by bootstrapping a component directly.

```
bootstrapApplication(AppComponent);
```

Difference: No AppModule needed the entry point is a component.

3. Lazy Loading

NgModule: You lazy load a module.

```
{ path: 'about', loadChildren: () => import('./about/about.module').then(m => m.AboutModule) }
```

Standalone: You lazy load a component directly.

```
{ path: 'about', loadComponent: () => import('./about/about.component').then(c => c.AboutComponent) }
```

Difference: Standalone eliminates “module shells” for lazy loading.

4.
Imports

In NgModule world

- When you import a module (like CommonModule, FormsModule, or MaterialButtonModule) into an NgModule,
- all components declared inside that module get access to those imports.

In Standalone world

- Each component declares its own imports explicitly.
- No automatic sharing → every component has only what it needs.

Analogy:

- NgModule: Like a buffet once you pay (import), everything is available to everyone.
- Standalone: Like an à la carte menu each component orders only the dishes (imports) it wants.

5.
Testing

- **NgModule:** Requires setting up a TestBed with the module.
- **Standalone:** You can import the component directly.

```
await TestBed.configureTestingModule({
    imports: [HomeComponent]
}).compileComponents();
```