Standalone Components

Starting Setup and Why We Want Standalone Components

- **Standalone** refers to components, directives, or pipes that can be used independently of **ngModule**
- Normally, we must declare **components** inside of **modules** to use them
- This **simplifies** writing Angular applications

Building a First Standalone Component

- In the **component**, add the **standalone: true** field to the **@Component** decorator's object
- **Reading** the standalone component **elsewhere**
  - Universally → Add the standalone component to the **imports** array of **AppModule**
  - In an existing **standalone component** holding your new **standalone component**, add the **imports: [<sc-name>]** field to the **@Component** decorator's object
- The **standalone component**'s **@Component** decorator also takes an **imports** field → Necessary for taking **normal directives**

A Standalone Root Component

- To make the **entire application standalone**, we must make **AppComponent** standalone as well
- Make **AppComponent** standalone as described before
- In **main.ts**, we now bootstrap code with **bootstrapApplication(AppComponent);**
- We can now get rid of **AppModule**

Services and Standalone Components

- Including **providedIn: 'root'** within the service's **Injectable** decorator will still allow services to work

- To have a global service, add an object to **main.ts**'s **bootstrapApplication** function
    - This should include a service's typical **providers** array
    - Still, the first route works best

Routing with Standalone Components

- Add **RouterModule** to **imports** within **AppComponent**
- Inside of **main.ts**'s **providers** array (inside **bootstrapApplication**), add **importProvidersFrom(AppRoutingModule)**

Lazy Loading

- Just **rewatch** the videos