Core Conceps

Components

Templates, directives, and data binding

Services and dependency injection

Angular is a platform and framework for building single-page client applications using HTML and TypeScript. Angular is written in TypeScript. It implements core and optional functionality as a set of TypeScript libraries that you import into your applications.

The architecture of an Angular application relies on certain fundamental concepts. The basic building blocks of the Angular framework are Angular components.

Components define views, which are sets of screen elements that Angular can choose among and modify according to your program logic and data

Components use services, which provide background functionality not directly related to views such as fetching data. Such services can be injected into components as dependencies, making your code modular, reusable, and efficient.

Components and services are classes marked with decorators. These decorators provide metadata that tells Angular how to use them.

The metadata for a component class associates it with a template that defines a view. A template combines ordinary HTML with Angular directives and binding markup that allow Angular to modify the HTML before rendering it for display.

The metadata for a service class provides the information Angular needs to make it available to components through dependency injection (DI)

An application's components typically define many views, arranged hierarchically. Angular provides the Router service to help you define navigation paths among views. The router provides sophisticated in-browser navigational capabilities.

See the Angular Glossary for basic definitions of important Angular terms and usage.

For the sample application that this page describes, see the live example / download example.

Components

Every Angular application has at least one component, the root component that connects a component hierarchy with the page document object model (DOM). Each component defines a class that contains application data and logic, and is associated with an HTML template that defines a view to be displayed in a target environment.

The @Component() decorator identifies the class immediately below it as a component, and provides the template and related component-specific metadata.

Decorators are functions that modify JavaScript classes. Angular defines a number of decorators that attach specific kinds of metadata to classes, so that the system knows what those classes mean and how they should work.

Angular is a framework that performs dynamic updating of DOM at/during execution.

Data Binding

index.html is loaded, scripts are dynamically injected into it and executed and app.component.ts injects app.component.html into root of index.html.

The cli creates the root components automatically.

app.component is the root component.

1. index.html is loaded
2. Scripts are dynamically inserted and executed starting Angular
3. main.ts is executed first and appModule is bootstraped.
4. appModule has a bootstrap array that contains app.component and all other imported modules that should be known when index.html is loaded.
5. app.component.ts contains selector of root element.
6. HTML template in then placed into root element of index.html.

A component is a TypeScript class that Angular instantiates.

@ Decorators are a TypeScript feature that enhances elements and classes. Used to denote that a TypeScript Class is a component.