Angular Basics

Introduction to Angular

* Angular is a platform and framework for building single-page client applications using HTML and Type
* Unlike some of the other frameworks that offer one-way data binding, the Angular framework provides two-way data binding. It seamlessly synchronizes the data between Model as well as View.

Note:

* + Initial release - 2.0 / 14 September 2016
  + Stable release - 13.1.1 / 15 December 2021
  + Preview release - 14.0.0-next.0 / 26 January 2022

Angular CLI

* A powerful to create, build, compile and serve Angular2+ App
* Used to generate new components, routes, services and pipes
* Installing Angular CLI
  + *npm install -g @angular/cli*
* Generating and serving Angular app
  + *ng new proj\_name --skip-install*
  + *cd proj\_name*
  + *npm install*
  + *ng serve*

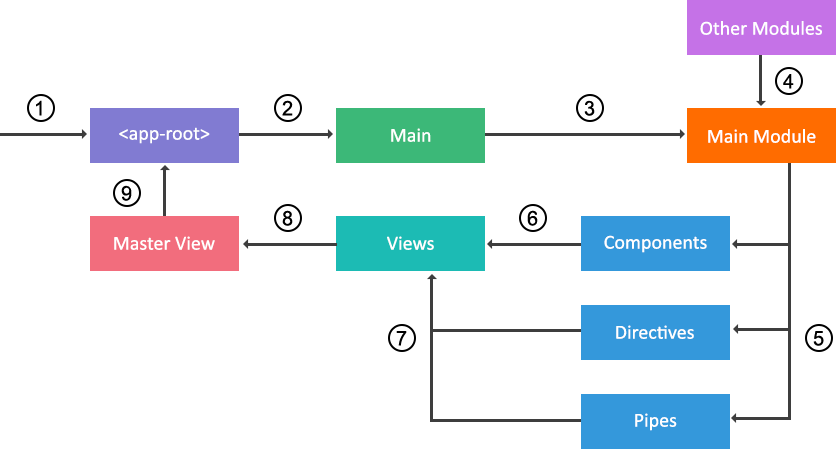
Angular CLI Options

|  |  |
| --- | --- |
| Options | Usage |
| Help | ng --help |
| Build | ng build *--env* |
| Build and Run | ng serve |
| Testing | ng test |
| End-End Testing | ng e2e |

Angular CLI Commands

|  |  |  |
| --- | --- | --- |
| Scaffold | Command | In Short |
| Module | ng generate module *my-module* | ng g m *my-module* |
| Component | ng generate component *my-component* | ng g c *my-component* |
| Directive | ng generate directive *my-directive* | ng g d *my-directive* |
| Pipe | ng generate pipe *my-pipe* | ng g p *my-pipe* |
| Service | ng generate service *my-service* | ng g s *my-service* |
| Guard | ng generate guard *my-guard* | ng g g *my-guard* |
| Class | ng generate class *my-class* | ng g cl *my-class* |
| Interface | ng generate interface *my-interface* | ng g i *my-interface* |
| Enum | ng generate enum *my-enum* | ng g e *my-enum* |

Angular Initialization Process

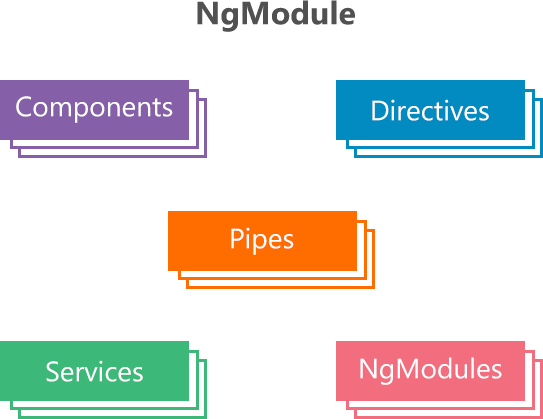


Angular Building Blocks

* Modules
* Components
* Templates
* Metadata
* Data binding
* Directives
* Pipes
* Routing
* Forms
* Services
* Dependency Injection

Modules

* + A module can organize an application into unified blocks of functionalities.
  + An Angular module is a class with an *@NgModule* decorator, Accepts a single metadata object whose properties describe the module
  + Each Angular app must have at least one module, known as root module



NgModule Metadata Main Properties

* + imports – Specify other dependent modules whose classes are required by the component templates declared in the module
  + declarations – Specify the components, directives, and pipes that belong to the module
  + bootstrap – Specify the main app view i.e root component. Only the root module can have this bootstrap property
  + exports – A subset of declarations that will be visible and usable in the other modules. A root module doesn’t have export option.
  + providers – Specify the services, accessible across the app

Built-In Modules

* + Angular has built-In library modules starting with the @angular as prefix



* + Built-In library & third part modules can be installed using npm manager
  + Built-In modules, components, services, directives etc. can be imported by using built-In library module

Component

* + A type of directives with template, styles and logic for user interaction
  + Exported as a custom HTML tag like as:
    - <my-component></my-component>
  + Initialized by Angular Dependency Injection engine



Angular Components Page View

ROOT COMPONENT

FOOTER COMPONENT

NEWS FEEDS COMPONENT

COMMENTS

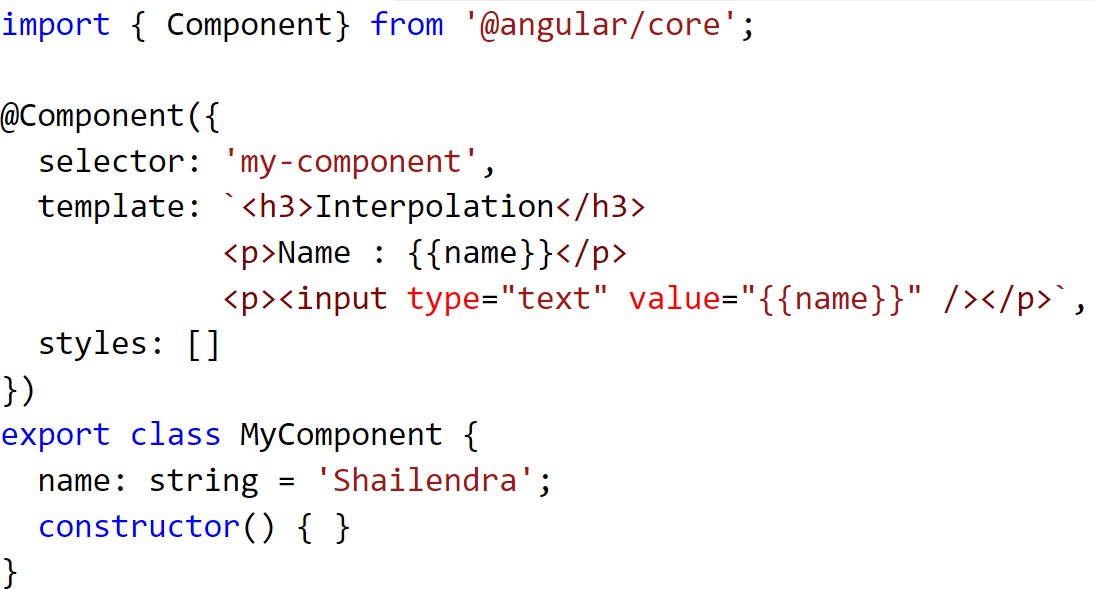
COMPONENT

CATEGORIES COMPONENT

ARTICLE COMPONENT

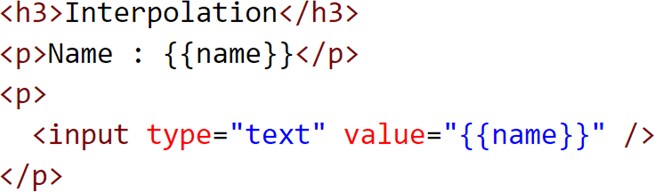
HEADER COMPONENT

Component Example



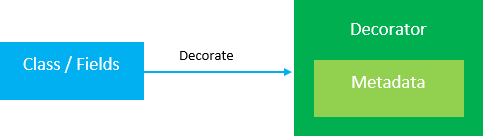
Template

* + Define the view of a component
  + Contains Html markup and angular directives, attributes etc.
  + Describe how a component is rendered on the page



Decorators

* + A function that adds metadata to a class, class members
  + These are prefix with @ symbol
  + Angular has built-In decorators like - @Component, @NgModule, @Directive, @Pipe etc.

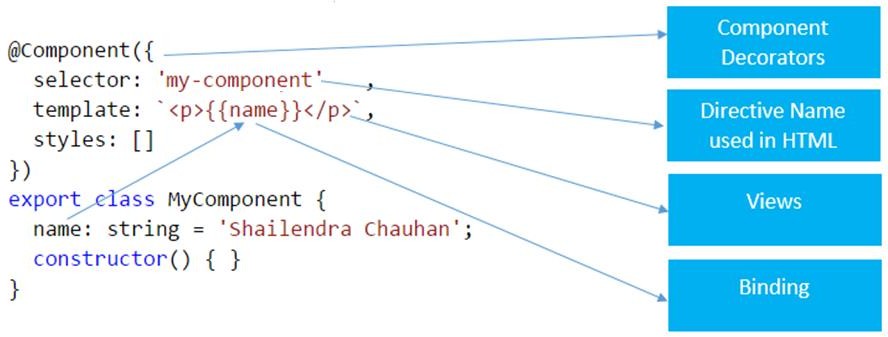


Types of Decorators

* + Class decorators
    - @NgModule – Used for defining a module
    - @Component – Used for defining a component
    - @Directive – Used for defining a directive
    - @Injectable – Used for injecting dependencies
    - @Pipe – Used for defining a pipe
  + Class field decorators
    - @Input – Used for receiving data (input) from parent to child component
    - @Output – Used for passing data (events) from child to parent component

Metadata

* + Tells Angular how to process a class
  + Decorators are used to attach metadata to a class



Angular Forms

* + HTML forms are an essential part of a web application
  + Angular provides two ways to create form – Template Driven and Model Driven

Forms

Template

Driven

Model Driven

Angular Form Building Blocks

Building

Blocks

FormGroup

FormControl

FormArray

Validations

Angular Form and Form Controls States

