Angular4 Cheat Sheet

by DevXperts

AngularJS is a structural framework for dynamic web apps. It lets you use HTML as your template language and lets you extend HTML's syntax to express your application's components clearly and succinctly. AngularJS's data binding and dependency injection eliminate much of the code you would otherwise have to write. And it handles all of the DOM and AJAX code you once wrote by hand and puts it in a well-defined structure.

Directives

Directives are the classes that can change the behaviour or appearance of the components by using CSS Classes, CSS Styles and events.
There are three kinds of directives in Angular:
1. Components – directives with a template.
2. Structural directives – change the DOM layout by adding and removing elements. (ngIF, ngFor, ngSwitch)
3. Attribute directives – change the appearance or behavior of an element.

Data binding

Interpolation use the {{expression}} to render the bound value to the component template, double curly braces {{ }} contain the template expression which allow us to read primitive or objects values from component properties. From Angular1.X we are using formally {{ }} for interpolation.

Data direction	Syntax	Binding type
One-way from data source to view target	**{{expression}}** [target] = "expression" bind-target = "expression"	Interpolation Property Attribute Class Style
One-way from view target to data source	(target) = "statement" on-target = "statement"	Event
Two-way	[(target)] = "expression" bindon-target = "expression"	Two-way

Components

• Components are the most basic building block of an UI in an Angular application.
• An Angular application is a tree of Angular components. Angular components are a subset of directives.
• Unlike directives, components always have a template and only one component can be instantiated per an element in a template.
• A component must belong to an NgModule in order for it to be usable by another component or application. To specify that a component is a member of an NgModule, you should list it in the declarations field of that NgModule.
• In addition to the metadata configuration specified via the Component decorator, components can control their runtime behaviour by implementing various Life-Cycle hooks.
• Angular 4 components are simply classes that are designated as a component with the help of a component decorator.

Modules

An NgModule class is adorned with @NgModule decorator function this will tell the angular application how to compile and run the module code. Every angular application will have at least one NgModule and every angular app has one root module class. The root module class is standardly termed as AppModule we can name it with any name, it exists in app.module.ts file.

@NgModule decorator identifies AppModule class as Angular Module class and it takes the metadata object that tells how to compile and launch the application. Some of the mostly used modules are HttpModule which contains Http Services, RouterModule which has router, BrowserModule which is needed to execute the application in browser.

The important properties are:
declarations - the view classes that belong to this module. Angular has three kinds of view classes: components, directives, and pipes.
exports - the subset of declarations that should be visible and usable in the component templates of other modules. The root module need not to be exported because no other component in angular needs to import the root module.
imports - other modules whose exported classes are needed by component templates declared in this module.
providers - creators of services that this module contributes to the global collection of services; they become accessible in all parts of the app.
bootstrap - the main application view, called the root component, that hosts all other app views. Only the root module should set this bootstrap property.

Bootstrap in main.ts
Bootstrapping an angular application can be done in many ways, the variations depends upon where we compile the application and where we run it. One of the way is by using JIT compiler and its recommended place is in src/main.ts.

Pipes

In Angular 1.X we used filters to modify (or) format (or) to choose the value within the template. From Angular >=4.X it is modernized as pipes
Pipes are used to transform the data
We can convert the value of one type to another type
We can use them as a function
Pipes makes your code clean and more structured and can execute functions within templates in effective way

Syntax:
With single pipe
Value | pipe1
Multiple pipes
Value | pipe1 | pipe2 | pipe3 ..
Parameterized Pipes
Value | pipe1: param1: param2 | pipe2:param1:param2

Here are few Built-in pipes from Angular.
1. DatePipe: Formats a date according to locale rules
2. UpperCasePipe: Transforms string to uppercase
3. LowerCasePipe: Transforms string to lowercase
4. DecimalPipe: Formats a number according to locale rules.
5. CurrencyPipe: Formats a number as currency using locale rules.
6. PercentPipe: Formats a number as percentage.
7. JsonPipe: Converts value into string using JSON.stringify. Useful for debugging.
8. SlicePipe: Creates a new List or String containing a subset (slice) of the elements

Services

In angular services are reusable classes which can be injected in components when it's needed. Using a separate service keeps components lean and focused on supporting the view, and makes it easy to unit-test components with a mock service.

Dependency Injection

Dependency Injection (DI) is the software design pattern and a framework which allows us to inject dependencies in different components across our application.
DI is used in an application when a module A needs to access the data from module B then module B is named as Independent Module and module A is dependent.

To make this tightly coupled access to loosely coupled we use DI pattern.
DI in angular has mainly three things
1. Injector: Injector object is used to expose API's and create instances of dependencies.
2. Provider: Provider tells the injector how to create instance for a dependency. A provider takes token and maps to a factory to create instances.
3. Dependency: It's a type of which an object should be created.

本速查表已收录到「前端大全」推荐的 Web 开发学习资源

关注「前端大全」，发送 资源 两个字，获取更多推荐资源