# Introduction to Angular 4

Pei Chen

## Background

In March 2017, Google released Angular 4.0.0, passing over the version number 3. Angular 4 is a latest version of Angular, which is backward compatible with Angular 2.x, but not Angular 1. In this paper, we will talk about the feature of Angular 4. The large part covered is also applicable to Angular 2.x. Besides, some new features of Angular 4 will be addressed.

## Framework Architecture

The architecture of Angular 4 mainly contains eight building blocks, which form up a functional application: modules, components, templates, metadata, data binding, directives, services, dependency injection.

A module is a pack of code designed to execute a simple task. Angular apps should be modular and it has its own modularity system, that is NgModules. Every Angular app has at least one module as the root module, conventionally named AppModule. Angular module is a class with a decorator @NgModule, which is used to receive meta data. The meta-data is the words depicting the properties of this module.

A component is a class within a module. It is used for manipulating the view part of the app. You can define its application logic, that is, what it does to manipulate the view, by an API of properties and methods.

A template is very similar to a standard HTML file. Actually, its suffix is html, too. It defines how the component is rendered. It is always accompanied by the component.

Compared to React, the component in Angular 4 is similar to the component in React. React component has its props, and Angular also owns them within the component. However, React has a render function (or embedded in the return value) to define what kind of html structure it will render in the view level. Angular 4 use a separate file as the template. React has a restriction that the template it renders with a component should have only one root node in DOM. Such restriction majorly comes from the convenience of DOM Diff. Angular templates are not restricted in this aspect.

The metadata is used to tell Angular how to process a class. We should note that if we define a component, the syntax is very similar as you define a class which is not a component. How can the framework know it is indeed a component? React's solution is either by stating "extend React.component" or giving this task to a module bundler. In Angular 4 here, we use metadata and attach it to the component's definition. If Typescript is used, we can attach a decorator like "@Component({})".

Data binding is a mechanism for coordinating parts of a template with parts of a component. The direction of such binding could be “from DOM to component”, “from component to DOM”, “both direction”. Such idea is different from React’ philosophy of single direction dataflow.

A directive is used to transform the DOM according to the instructions given by the directives. Angular 4 can change the DOM according to the instructions given by directives. While in React, normally you should use JavaScript function to do such actions first and then insert it into the DOM structure that you want to render. Or in the rendering JSX, we have a curly braces, and you put your business logic into it as JavaScript code. One example of Angular 4 directive is: <li *ngFor=”let element of elements”></li>. Then, Angular 4 knows that it should render a <li> tag for each element in elements. Another example is: <the-name-of-component *ngIf=”isSelected”></the-name-of-component>. In this example, there is a component named the-name-of-component. It has a property named isSelected (though isSelected may come from other place), which can be true or false. When the app comes to render the template and see this line of HTML, it will firstly find out the value of isSelected. If true, then this component is rendered. If false, the component is skipped. Those are how directives helps Angular 4 to control the view.

A service is an independent module which encapsulates a set of functionalities. It can be injected and used by an Angular app. The definition of service is a broad category, including values, functions and other features.

Dependency injection is a method to provide an instance of class with the dependencies it needs, many of which are services. When Angular 4 creates a new component, it firstly makes a request to an injector for the services the component needs.

## New Features of Angular 4

In this part, we will talk about some awesome new features of Angular 4. For the lack of space, not all new features will be covered in this paper.

1. If-else syntax in component HTML templates

In the introduction above, we talk about how a directive *ngIf plays a role in <the-name-of-component *ngIf=”isSelected”></the-name-of-component>. However, we don’t have *ngElse. Previously, you can either use *ng-switch-when or maintain do another *ngIf with the another indicator. Each has their disadvantages and byproducts. Now, Angular 4 tries to make this logic more elegant in programming. Now you can use if-else syntax in component HTML templates.

2. Support of newer version TypeScript

As is recommended by Angular Development Team, TypeScript can be used in designing an Angular App. Angular 2 has some conflict issues in TypeScript 2.1 and 2.2. Angular 4 now supports them. To be specific, old version of TypeScript doesn’t not allow you to define a variable and tell TypeScript that this variable’s value could be null or undefined. In TypeScript 2.1 and 2.2, a change is made to allow such usage. However, Angular 2 is not ready to prepare for that change. Now, Angular 4 fixes it and this feature is compliant in Angular 4.

3. Stand alone animation module

We've talked about the modular feature of Angular. Angular provides a lot of built-in module in "@angular/core". And you can import them like this line of code: import {Component, OnInit, animate, state, style, transition, trigger} from '@angular/core'. Now, in angular 4, it is no longer available to find those animation members from core module. There are now in "@angular/animations". Therefore, the equivalent codes are now: import {Component, OnInit} from '@angular/core'; import {animate, state, style, transition, trigger} from '@angular/animations'.

We should note that in Angular 2, with animation in the core module, even if you don't use any animation, those exist in your production codes. Now, with this change, the core code is simpler. It provides a choice to the developer, that is, if you want to use such animation features, you should manually import it from outside.

4. Performance boost with FESM

FESM stands for Flattened ECMAScript Module. Quoting the changelog: This format should help tree-shaking, help reduce the size of your generated bundles, and speed up build, transpilation, and loading in the browser in certain scenarios.

## Conclusions

Angular 4 is a complete set of framework. It has plenty of tools, covering data binding, render, angular UI library, filter, directive, service, route, http, cookie, inject, factory, provider. Basically the tools you will be faced with in web development has a similar replacement within Angular.

The architecture of Angular is clear, user-friendly and extensible. The model, view and controller is clearly separated, letting programmer focus on business logic. Also, it has no strong invasion into html, thus facilitate the cooperation between front end engineer and designer. The dependency injection mechanism facilitates the switch of a specific object without affecting others.

A popular criticism on Angular is that the MVVM using data binding mechanism slows down the app while the virtual DOM and DOM diff technology. Also, with Redux's control of data flow, the structure of React is cleaner and easier to maintain. Though we should note that in some situations MVVM is better than Virtual DOM, it is obvious that Angular 1 is far slower than React in general. With such deficiencies and other issues, Angular 2 is released with breaking changes. It does improve it in speed but still keeps the philosophy of MVVM, using data binding and checking data rather than DOM change. In the new released version Angular 4, performance improvement is also made but data binding is definitely kept, too.

## References

[1] <Official Documentation – Architecture Overview>
https://angular.io/docs/ts/latest/guide/architecture.html

[2] <5 Features to Watch Out for in Angular v4> https://scotch.io/tutorials/5-features-to-watch-out-for-in-angular-4
[3] https://www.zhihu.com/question/31809713