# AngularJs1.x与Angular2的主要区别

|  |  |  |
| --- | --- | --- |
|  | AngularJs 1.x | Angular 2 |
| 名字 | AngularJs | Angular，去掉了JS |
| 编程语言 | JavaScript，可以直接在浏览器中运行代码 | TypeScript，不能直接在浏览器中运行代码，必须经过编译  所以必须使用构建工具 |
| 作用域Scope | 作用域链（相当于视图模型） | 无 |
| 脏值检查 | $digest机制，多次检查整个作用域链 | 采用与VueJs类似的机制，取消主动脏值检查 |
| 控制器Controller | 非常重要，用于封装视图控制逻辑甚至业务逻辑 | 无，组件化开发 |
| 指令 | 指令数量较多；指令与组件区别较少 | 指令数量较少，但指令更强大了；指令与组件区别明显，指令使用\*符号，如\*ngFor、\*ngIf |
| 组件 | 通过组件定义对象CDO进行组件定义；组件比较鸡肋，大多数情况下都在使用指令 | 通过装饰器元数据和组件类进行定义，组件呈现和控制页面上的一块区域；大量、直接使用组件构建应用页面 |
| 过滤器 | Filter | 名字改为Pipe，增加asyc、slice、percent去掉filter、limit、orderby，Angular2认为筛选/分页/排序属于视图控制逻辑/业务逻辑 |
| 绑定 | 通过大量指令和模板插值实现，单向绑定双向绑定区别不明显 | 大量使用括号，  () 从视图到数据方向，指事件  [] 从数据到视图方向  [()] 双向绑定  {{}}，模板插值  严格区分2种单向绑定，区分单向绑定和双向绑定 |
| 父子组件/指令传值 | 作用域链继承  广播/发射事件  属性传值  服务共享 | 订阅事件  属性传值  服务共享 |
| 服务 | 按照Angular规定好的方式定义服务，服务属于模块级，定义时自动注册到模块中 | 普通类，需要在模块或组件级别的providers元数据中声明 |
| 模块 | 在应用层面明确地区分为根模块和特性模块（可以理解为普通模块），模块内部包括组件、指令、服务（包括服务类、值、函数） | 一般这样区分：官方模块、第三方模块、自定义模块。模块包含的东西比较多，比较容易混淆 |
| 路由 | 内置的路由机制不支持嵌套，通常复杂点的页面就要使用第三方路由ui-router | 支持平级多路由，支持基于组件的多级嵌套路由，支持路由守卫 |
| 元数据 | 受限于JavaScript语言而无明显的元数据概念 | 有明确的基于装饰器的元数据概念 |
| 组件生命周期 | 没有明确的生命周期机制 | 有明确的经过精心设计生命周期，如：  OnInit  OnDoCheck  AfterContentInit  AfterContentChecked  AfterViewInit  AfterViewChecked  OnChanges  OnDestroy |
| 启动 | 可以通过ng-app自动启动或代码启动 | 没有ng-app指令  一般通过代码启动根模块的方式启动 |
| 事件对象 | 在代码中随意使用事件对象 | 不建议在组件代码中使用事件对象，而推荐使用模版变量获取想要的值然后仅将值传到组件内部 |
| 事件过滤 | 无 | 提供了类似VueJs的机制 |
| 依赖注入 | 基于名字（字符串）的注入 | 基于类（要利用强类型机制）的注入 |
| 提供者 | 规范定义的可配置的提供者机制，比较复杂 | 简化为工厂函数+简单的提供者定义对象 |
| 命名规范 | 零散的规定 | 完整的规范 |

整体来看，Angular2比AngularJs1.x简单且功能强大，但因为Angular使用TypeScript导致纯前端的开发工程师不容易上手；再加上Angular2与AngularJs1.x差异较大，AngularJs1.x的理念有时反而成为学习Angular2的障碍。

使用Angular2的原因：

1. 希望使用TypeScript语言提高开发效率。TypeScript提供了比JavaScript更强大的面向对象编程能力，而且因为TypeScript是强类型的所以开发工具可以提供更强大的错误检查机制、代码重构机制，开发体验更好，效率更高。
2. Angular2简单，代表未来，更有生命力。