**Angular.js作用域是如何工作的**

1. **作用域是什么**
2. 是一个存储应用数据模型的对象
3. 为 [表达式](https://www.angularjs.net.cn/tutorial/3.html) 提供了一个执行上下文
4. 作用域的层级结构对应于 DOM 树结构
5. 作用域可以监听 [表达式](https://www.angularjs.net.cn/tutorial/3.html) 的变化并传播事件
6. **作用域有什么**
7. 提供了 ([$watch](https://www.angularjs.net.cn/tutorial/api/ng.$rootScope.Scope#methods_$watch)) 方法监听数据模型的变化
8. 提供了 ([$apply](https://www.angularjs.net.cn/tutorial/api/ng.$rootScope.Scope#methods_$apply)) 方法，可以把不是由Angular触发的数据模型的改变引入Angular的控制范围内。
9. 提供了基于原型链继承其父作用域属性的机制。
10. 提供了表达式的执行环境。
11. **作用域是如何工作的**
12. **作用域作为数据模型使用**

作用域是Web应用的控制器和视图之间的粘结剂。在Angular中，最直观的表现是：在自定义指令中，处在模版的链接(linking)阶段时， 指令(directive)会设置一个$watch函数监听着作用域中各表达式（注：这个过程是隐式的）。这个 $watch 允许指令在作用域中的属性变化时收到通知，进而让指令能够根据这个改变来对DOM进行重新渲染，以便更新已改变的属性值。

链条关系：控制器 --> 作用域 --> 视图（DOM）

1. **作用域分层结构**

作用域的结构对应于DOM结构，最顶层和DOM树有根节点一样，每个Angular应用有且仅有一个 [root scope](https://www.angularjs.net.cn/tutorial/api/ng.$rootScope)，子级作用域也和DOM树的子节点一样，可以有多个的。

一般情况下，当新的作用域被创建时，它是以嵌入在父级作用域的子级的形式被创建的，这样就形成了与其所关联的DOM树相对应的一个作用域的树结构。

1. **基于作用域的事件传播**

1、broadcasted ：从父级作用域广播至子级作用域

2、emitted ：从子级作用域往上发射到父级作用域

然后使用scope.$on()监听事件即可。

1. **作用域的生命周期**
2. **创建期**

[root scope](https://www.angularjs.net.cn/tutorial/api/ng.$rootScope) 是在应用程序启动时创建的。另外，在指令的模版链接阶段（template linking），指令会创建一些新的子级 scope。

1. **注册$watch**

在模版链接阶段（template linking），指令会往作用域中注册 [监听器(watch)](https://www.angularjs.net.cn/tutorial/api/ng.$rootScope.Scope#methods_$watch)，而且不止一个。这些 $watch 用来监测数据模型的更新并将更新值传给DOM。

1. **数据模型变化**

要想让数据模型的变化能够很好的被Angular监测，需要让它们在 [scope.$apply()](https://www.angularjs.net.cn/tutorial/api/ng.$rootScope.Scope#methods_$apply) 里发生。 当然，对于Angular本身的API来讲，Angular已经隐式地为我们做了这一点。

1. **数据模型变化监测**

在把数据变化 $apply 进来之后，Angular开始进入 [$digest](https://www.angularjs.net.cn/tutorial/api/ng.$rootScope.Scope#methods_$digest) 轮循（就是调用 $digest() 方法），首先是 rootscope 进入 $digest ，然后由其把各个监听表达式或是函数的任务传播分配给所有的子级作用域，那样各个作用域就各司其职了，如果监听到自己负责的数据模型有变化，马上就调用 $watch 。（译注：这里所说的从根scope往下分发是译者自己的想法，如有错误，请纠正）

1. **销毁作用域**

当子级作用域不再需要的时候，这时候创建它们的就会负责把它们回收或是销毁（注：比如在指令中，创建是隐式的，销毁可以不但可以是隐式的，也可以是显式的，如 scope.$destroy()）。销毁是通过 [scope.$destroy()](https://www.angularjs.net.cn/tutorial/api/ng.$rootScope.Scope#methods_$destroy) 这个方法。销毁之后，$digest() 方法就不会继续往子级作用域传播了，这样也就可以让垃圾回收系统把这一个作用域上用来存放数据模型的内存给回收利用了。