谈谈AngularJS中的$watch - 专栏 - 前端乱炖

在文章任何区域双击击即可给文章添加【评注】！浮到评注点上可以查看详情。
隐藏标注

在使用AngulaJS编写应用时，我们经常需要做的一件事情就是对模型中的变量进行监视，并对其发生的变化做出相应的回应。AngularJS为我们提供了一个非常方便的$watch方法，它可以帮助我们在每个scope中监视其中的变量。下面是一个非常简单的例子：

```
< span>html<< span>head<< span>script src='./lib.angular.min.js'<< span>/script<< span>/head<< span>body ng-app='watch'<
< span>input ng-model='name' type='text'/<< span>div<change count:{{count}}< span>/div<< span>script<        angular.mod
ule('watch',[]).run(['$rootScope',function($rootScope){                          $rootScope.count =0;
          $rootScope.name ='Alfred';                              $rootScope.$watch('name',function(){
            $rootScope.count++;})}]);< span>/script<< span>/body<< span>/html<
```

上面的这段代码非常简单，它用$watch来对$rootScope中的name进行监视，并在它发生变化的时候将$rootScope中的count属性增加1。因此，每当我们对name进行一次修改时，下面显示的change count数字就会增加1。

在AngularJS内部，每当我们对ng-model绑定的name属性进行一次修改，AngularJS内部的$digest就会运行一次，并在运行结束之后检查我们使用$watch来监视的东西，如果和进行上一次$digest之前相比有了变化，则执行我们在其中绑定的处理函数。

然而，我们在实际运用中常常不只是对一个原始类型的属性进行监视，如果你还记得Javascript中的六种基本类型，你一定会记得原始类型（数字，字符串）和引用类型的区别。对于原始类型，如果我们使用了一个赋值操作，则这个原始类型变量会“真正的”被进行一次复制，然而对于引用类型，在进行赋值时，仅仅时将赋值的变量指向了这个引用类型。在AngularJS的$watch方法中，对两者的操作也有不同之处。原始类型，就像我们上面例子中提到的$rootScope，没有什么特别之处，然而如果要对一个引用类型，尤其是在实际运用中常见的对象数组进行监视时，情况就不一样了。我们来看下面的例子：

```
< span>html<< span>head<< span>script src='./lib.angular.min.js'<< span>/script<< span>/head<< span>body ng-app='watch'<
< span>div hg-repeat='item in items'<< span>input ng-model='item.a'/<< span>span<{{item.a}}< span>/span<< span>/div<< sp
an>div<change count:{{count}}< span>/div<< span>script<        angular.module('watch',[]).run(['$rootScope',function($ro
otScope){                          $rootScope.count =0;                          $rootScope.items =[{"a":1},{"a":2},
{"a":3},{"a":4}]                          $rootScope.$watch('items',function(){                                  $rootSc
ope.count++;})}]);< span>/script<< span>/body<< span>/html<
```

此时，如果我们对四个input中的a进行改变时，我们会发现，count的值依然是0。这是怎么回事？难道没有$watch失灵了吗？

正如我们前面所说的，$watch在对待原始类型和引用类型会有不同的处理方式，这就要首先说一说$watch函数的第三个参数。在前面的例子中，我们知道，$watch函数有接收两个参数，第一个参数是需要监视的对象，第二个参数是在监视对象发生变化时需要调用的函数，实际上$watch还有第三个参数，它在默认情况下是false。在默认情况下，即不显式指明第三个参数或者将其指明为false时，我们进行的监视叫做“引用监视”。引用监视的原词的“reference watch”，它的意思是只要监视的对象引用没有发生变化，就不算它发生了变化。具体来说，在上面的例子中，只要是items的引用没有发生变化，就算items中的一些属性发生了变化，$watch也会当做没有看见。那么在什么时候算是引用发生了变化呢？比如说将一个新的数组newItems赋值给items，此时$watch才会站出来说：“你变了！你再也不是以前我认识的那个和我一起看星星看月亮聊人生理想并且教会我什么叫做爱的人了！”

相反，如果我们将$watch的第三个变量设置为true，那么此时我们进行的监视叫做“全等监视”，原词是“equality watch”。此时，$watch就像是一个醋意十足的恋人，只要看他的对象有一点风吹草动，马上就跳出来，大喊大叫：“你冷酷你无情，你无理取闹。就算我冷酷，我无情，我无理取闹，也没有你冷酷你无情，你无理取闹！尔康。。。。。”。

因此，有同学就会问了，既然全等监视这么好，那么我们为什么不直接用全等监视呢？当然，任何事情都有好的坏的两个方面，全等监视固然是好，但是它在运行时需要先遍历整个监视对象，然后在每次$digest之前使用angular.copy()将整个对象深拷贝一遍然后在运行之后用angular.equal()将前后的对象进行对比，上面的例子中因为items比较简单，因此可能性能上不会有什么差别，但是到了实际生产时，我们要面对的数据千千万万，可能因为全等监视这一个设置就会消耗大量的资源，让应用停滞不前。因此这就需要我们在使用时进行权衡，究竟应该使用哪一种监视方式。

除了上面提到的两种方式之外，在angular 1.1.4版本之后，添加了一个$watchCollection()方法来针对数组（也就是集合）进行监视，它的性能介于全等监视和引用监视二者之间，即它并不会对数组中每一项的属性进行监视，但是可以对数组的项目的增减做出反应。比如还是上面的例子：

```
$rootScope.items =[{"a":1},{"a":2},{"a":3},{"a":4}]                    $rootScope.$watchCollection('items',funct
ion(){                          $rootScope.count++;})}]);
```

如果改变了items[0]的a属性值，$watch并不会做出反应，但是如果我们在items上push或者pop了一个项目，$watch就会开始行动了。

关于AngularJS中$watch部分就讲这么多，还是那句话“最好的学习方式就是实践”。多coding，多犯错，一定能学好AngularJS。

如果你觉得本文对你有帮助，请点击为我提供赞助