



理解\$watch, \$apply 和 \$digest --- 理解数据绑定过程

<http://zheteng.me/javas...> 08-09 12:30 08-04 19:41 73972 15022 27 AngularJS scope

原文地址: <http://angular-tips.com/blog/2013/08/watch-how-the-apply-runs-a-digest/>

注

这篇博文主要是写给新手的, 是给那些刚刚开始接触Angular, 并且想了解数据绑定是如何工作的人。如果你已经对Angular比较了解了, 那强烈建议你直接去阅读源代码。

Angular用户都想知道数据绑定是怎么实现的。你可能会看到各种各样的词汇: `$watch`, `$apply`, `$digest`, `dirty-checking` ... 它们是什么? 它们是如何工作的呢? 这里我想回答这些问题, 其实它们在官方的文档里都已经回答了, 但是我还是想把它们结合在一起来讲, 但是我只是用一种简单的方法来讲解, 如果要想了解技术细节, 查看源代码。

让我们从头开始吧。

浏览器事件循环和Angular.js扩展

我们的浏览器一直在等待事件, 比如用户交互。假如你点击一个按钮或者在输入框里输入东西, 事件的回调函数就会在javascript解释器里执行, 然后你就可以做任何DOM操作, 等回调函数执行完毕时, 浏览器就会相应地对DOM做出变化。Angular拓展了这个事件循环, 生成一个有时成为 `angular context` 的执行环境 (记住, 这是个重要的概念), 为了解释什么是 `context` 以及它如何工作, 我们还需要解释更多的概念。

\$watch 队列 (\$watch list)

每次你绑定一些东西到你的UI上时你就会往\$watch队列里插入一条 `$watch`。想象一下 `$watch` 就是那个可以检测它监视的model里时候有变化的东西。例如你有如下的代码

index.html

User: <input type="text" ng-model="user" />

Password: <input type="password" ng-model="pass" />

在这里我们有个 `$scope.user`，他被绑定在了第一个输入框上，还有一个 `$scope.pass`，它被绑定在了第二个输入框上，然后我们在 `$watch list` 里面加入两个 `$watch`：

controllers.js

```
app.controller('MainCtrl', function($scope) {
  $scope.foo = "Foo";
  $scope.world = "World";
});
```

index.html

Hello, {{ world }}

这里，即便我们在 `$scope` 上添加了两个东西，但是只有一个绑定在了UI上，因此在这里只生成了一个 `$watch`。再看下面的例子：`controllers.js`

```
app.controller('MainCtrl', function($scope) {
  $scope.people = [...];
});
```

index.html

```
<ul>
  <li ng-repeat="person in people">
    {{person.name}} - {{person.age}}
  </li>
</ul>
```

这里又生成了多少个 `$watch` 呢？每个person有两个（一个name，一个age），然后ng-repeat又有一个，因此10个人一共是 $(2 * 10) + 1$ ，也就是说有21个 `$watch`。因此，每一个绑定到了UI上的数据都会生成一个 `$watch`。

对，那这写 `$watch` 是什么时候生成的呢？当我们的模版加载完毕时，也就是在 `linking` 阶段（Angular分为`compile`阶段和`linking`阶段---译者注），Angular解释器会寻找每个`directive`，然后生成每个需要的 `$watch`。听起来不错哈，但是，然后呢？

`$digest` 循环（这个`digest`不知道怎么翻译）

还记得我前面提到的扩展的事件循环吗？当浏览器接收到可以被 `angular context` 处理的事件时，`$digest` 循环就会触发。这个循环是由两个更小的循环组合起来的。一个处理 `evalAsync` 队列，另一个处理 `$watch` 队列，这个也是本篇博文的主题。这个是处理什么的呢？`$digest` 将会遍历我们的 `$watch`，然后询问：

- 嘿，`$watch`，你的值是什么？
 - 是9。
- 好的，它改变过吗？
 - 没有，先生。
- （这个变量没变过，那下一个）
- 你呢，你的值是多少？
 - 报告，是 `Foo`。
- 刚才改变过没？
 - 改变过，刚才是 `Bar`。
- （很好，我们有DOM需要更新了）
- 继续询问知道 `$watch` 队列都检查过。

这就是所谓的 `dirty-checking`。既然所有的 `$watch` 都检查完了，那就要问了：有没有 `$watch` 更新过？如果有至少一个更新过，这个循环就会再次触发，直到所有的 `$watch` 都没有变化。这样就能够保证每个`model`都已经不会再变化。记住如果循环超过10次的话，它将会抛出一个异常，防止无限循环。当 `$digest` 循环结束时，DOM相应地变化。

例如： `controllers.js`

```
app.controller('MainCtrl', function() {  
    $scope.name = "Foo";  
  
    $scope.changeFoo = function() {  
        $scope.name = "Bar";  
    }  
});
```

```

    }
  });

```

index.html

```

{{ name }}
<button ng-click="changeFoo()">Change the name</button>

```

这里我们有一个 `$watch` 因为ng-click不生成 `$watch`（函数是不会变的）。

- 我们按下按钮
- 浏览器接收到一个事件，进入 `angular context`（后面会解释为什么）。
- `$digest` 循环开始执行，查询每个 `$watch` 是否变化。
- 由于监视 `$scope.name` 的 `$watch` 报告了变化，它会强制再执行一次 `$digest` 循环。
- 新的 `$digest` 循环没有检测到变化。
- 浏览器拿回控制权，更新与 `$scope.name` 新值相应部分的DOM。

这里很重要的（也是许多人的很蛋疼的地方）是每一个进入 `angular context` 的事件都会执行一个 `$digest` 循环，也就是说每次我们输入一个字母循环都会检查整个页面的所有 `$watch`。

通过 `$apply` 来进入 `angular context`

谁决定什么事件进入 `angular context`，而哪些又不进入呢？ `$apply`！

如果当事件触发时，你调用 `$apply`，它会进入 `angular context`，如果没有调用就不会进入。现在你可能会问：刚才的例子里我并没有调用 `$apply` 啊，为什么？Angular为了做了！因此你点击带有ng-click的元素时，时间就会被封装到一个 `$apply` 调用。如果你有一个 `ng-model="foo"` 的输入框，然后你敲一个 `f`，事件就会这样调用 `$apply("foo = 'f';")`。

Angular什么时候不会自动为我们 `$apply` 呢？

这是Angular新手共同的痛处。为什么我的jQuery不会更新我绑定的东西呢？因为jQuery没有调用 `$apply`，事件没有进入 `angular context`，`$digest` 循环永远没有执行。

我们来看一个有趣的例子：

假设我们有下面这个directive和controller

`app.js`

```
app.directive('clickable', function() {

  return {
    restrict: "E",
    scope: {
      foo: '=',
      bar: '='
    },
    template: '<ul style="background-color: lightblue"><li>
{{foo}}</li><li>{{bar}}</li></ul>',
    link: function(scope, element, attrs) {
      element.bind('click', function() {
        scope.foo++;
        scope.bar++;
      });
    }
  };

});

app.controller('MainCtrl', function($scope) {
  $scope.foo = 0;
  $scope.bar = 0;
});
```

它将 `foo` 和 `bar` 从controller里绑定到一个list里面，每次点击这个元素的时候，`foo` 和 `bar` 都会自增1。

那我们点击元素的时候会发生什么呢？我们能看到更新吗？答案是否定的。因为点击事件是一个没有封装到 `$apply` 里面的常见的事件，这意味着我们会失去我们的计数吗？不会

真正的结果是：`$scope` 确实改变了，但是没有强制 `$digest` 循环，监视 `foo` 和 `bar` 的 `$watch` 没有执行。也就是说如果我们自己执行一次 `$apply` 那么这些 `$watch` 就会看见这些变化，然后根据需要更新DOM。

试试看吧：<http://jsbin.com/opimat/2/>

如果我们点击这个directive（蓝色区域），我们看不到任何变化，但是我们点击按钮时，点击数就更新了。如刚才说的，在这个directive上点击时我们不会触发

`$digest` 循环，但是当按钮被点击时，`ng-click`会调用 `$apply`，然后就会执行 `$digest` 循环，于是所有的 `$watch` 都会被检查，当然就包括我们的 `foo` 和 `bar` 的 `$watch` 了。

现在你在想那并不是你想要的，你想要的是点击蓝色区域的时候就更新点击数。很简单，执行一下 `$apply` 就可以了：

```
element.bind('click', function() {
  scope.foo++;
  scope.bar++;

  scope.$apply();
});
```

`$apply` 是我们的 `$scope`（或者是`directive`里的 `link` 函数中的 `scope`）的一个函数，调用它会强制一次 `$digest` 循环（除非当前正在执行循环，这种情况下会抛出一个异常，这是我们不需要在那里执行 `$apply` 的标志）。

试试看：<http://jsbin.com/opimat/3/edit>

有用啦！但是有一种更好的使用 `$apply` 的方法：

```
element.bind('click', function() {
  scope.$apply(function() {
    scope.foo++;
    scope.bar++;
  });
});
```

有什么不一样的？差别就是在第一个版本中，我们是在 `angular context` 的外面更新的数据，如果有发生错误，Angular永远不知道。很明显在这个像个小玩具的例子里面不会出什么大错，但是想象一下我们如果有个`alert`框显示错误给用户，然后我们有个第三方的库进行一个网络调用然后失败了，如果我们不把它封装进 `$apply` 里面，Angular永远不会知道失败了，`alert`框就永远不会弹出来了。

因此，如果你想使用一个jQuery插件，并且要执行 `$digest` 循环来更新你的DOM的话，要确保你调用了 `$apply`。

有时候我想多说一句的是有些人在不得不调用 `$apply` 时会“感觉不妙”，因为他们会觉得他们做错了什么。其实不是这样的，Angular不是什么魔术师，他也不知道第三方库想要更新绑定的数据。

使用 `$watch` 来监视你自己的东西

你已经知道了我们设置的任何绑定都有一个它自己的 `$watch`，当需要时更新DOM，但是我们如果要自定义自己的watches呢？简单

来看个例子：

`app.js`

```
app.controller('MainCtrl', function($scope) {
  $scope.name = "Angular";

  $scope.updated = -1;

  $scope.$watch('name', function() {
    $scope.updated++;
  });
});
```

`index.html`

```
<body ng-controller="MainCtrl">
  <input ng-model="name" />
  Name updated: {{updated}} times.
</body>
```

这就是我们创造一个新的 `$watch` 的方法。第一个参数是一个字符串或者函数，在这里是只是一个字符串，就是我们要监视的变量的名字，在这里，`$scope.name` (注意我们只需要用 `name`)。第二个参数是当 `$watch` 说我监视的表达式发生变化后要执行的。我们要知道的第一件事就是当controller执行到这个 `$watch` 时，它会立即执行一次，因此我们设置updated为-1。

试试看：<http://jsbin.com/ucaxan/1/edit>

例子2：

`app.js`

```
app.controller('MainCtrl', function($scope) {
  $scope.name = "Angular";
```



```

$scope.name = 'foo';

$scope.$watch('name', function() {
    // AKA first run
    $scope.updated++;
});

```

index.html

```

<body ng-controller="MainCtrl">
  <input ng-model="name" />
  Name updated: {{updated}} times.
</body>

```

watch的第二个参数接受两个参数，新值和旧值。我们可以用他们来略过第一次的执行。通常你不需要略过第一次执行，但在这个例子里面你是需要的。灵活点嘛少年。

例子3:

app.js

```

app.controller('MainCtrl', function($scope) {
    $scope.user = { name: "Fox" };

    $scope.updated = 0;

    $scope.$watch('user', function(newVal, oldVal) {
        if (newVal === oldVal) { return; }
        $scope.updated++;
    });
});

```

index.html

```

<body ng-controller="MainCtrl">
  <input ng-model="user.name" />
  Name updated: {{updated}} times.

```


</body>

我们想要监视 `$scope.user` 对象里的任何变化，和以前一样这里只是用一个对象来代替前面的字符串。

试试看: <http://jsbin.com/ucaxan/3/edit>

呃？没用，为啥？因为 `$watch` 默认是比较两个对象所引用的是否相同，在例子1和2里面，每次更改 `$scope.name` 都会创建一个新的基本变量，因此 `$watch` 会执行，因为对这个变量的引用已经改变了。在上面的例子里，我们在监视 `$scope.user`，当我们改变 `$scope.user.name` 时，对 `$scope.user` 的引用是不会改变的，我们只是每次创建了一个新的 `$scope.user.name`，但是 `$scope.user` 永远是一样的。

例子4:

app.js

```
app.controller('MainCtrl', function($scope) {
  $scope.user = { name: "Fox" };

  $scope.updated = 0;

  $scope.$watch('user', function(newValue, oldValue) {
    if (newValue === oldValue) { return; }
    $scope.updated++;
  }, true);
});
```

index.html

```
<body ng-controller="MainCtrl">
  <input ng-model="user.name" />
  Name updated: {{updated}} times.
</body>
```

试试看: <http://jsbin.com/ucaxan/4/edit>

现在有用了吧！因为我们对 `$watch` 加入了第三个参数，它是一个bool类型的参数，表示的是我们比较的是对象的值而不是引用。由于当我们更新 `$scope.user.name` 时 `$scope.user` 也会改变，所以能够正确触发。

关于 `$watch` 还有很多tips&tricks, 但是这些都是基础。

总结

好吧, 我希望你们已经学会了在Angular中数据绑定是如何工作的。我猜想你的第一印象是 `dirty-checking` 很慢, 好吧, 其实是不对的。它像闪电般快。但是, 是的, 如果你在一个模版里有2000-3000个 `watch`, 它会开始变慢。但是我觉得如果你达到这个数量级, 就可以找个用户体验专家咨询一下了

无论如何, 随着ECMAScript6的到来, 在Angular未来的版本里我们将会 有 `Object.observe` 那样会极大改善 `$digest` 循环的速度。同时未来的文章也会涉及一些tips&tricks。

另一方面, 这个主题并不容易, 如果你发现我落下了什么重要的东西或者有什么东西完全错了, 请指正 (原文是在GITHUB上PR 或报告issue)

□ 16 □ 23 □ 0 发表评论□

评论：理解\$watch, \$apply 和 \$digest --- 理解数据绑定过程

您还没有登录, 不能发表评论哦

[使用Markdown语法, 24 到 20480 字节, 当前0字节]

编辑 / 预览 提交



评论：理解\$watch, \$apply 和 \$digest --- 理解数据绑定过程

看了之后对这几个函数理解了。

shadowprompt 08-04 19:41发表

□0 □0 □0 回复□



评论：黄网：<http://www.301se.com>

DEMO地址：<http://www.91wzl.com>

ifrancis 06-04 11:35发表

□0 □0 □0 回复□



评论：理解\$watch, \$apply 和 \$digest --- 理解数据绑定过程

理解了，赞一个。。。。。。。

noney 02-19 11:46发表

0 0 0 回复



评论：理解\$watch, \$apply 和 \$digest --- 理解数据绑定过程

不错的文章。。。。。

moonwater 01-19 16:18发表

0 0 0 回复



评论：理解\$watch, \$apply 和 \$digest --- 理解数据绑定过程

全讯网

qxwys2 11-19 12:21发表

0 0 0 回复



评论：理解\$watch, \$apply 和 \$digest --- 理解数据绑定过程

新2

qxwys2 11-18 11:52发表

0 0 0 回复



评论：理解\$watch, \$apply 和 \$digest --- 理解数据绑定过程

新2网址

qxwys2 11-17 13:20发表

0 0 0 回复



评论：理解\$watch, \$apply 和 \$digest --- 理解数据绑定过程

很赞的一篇文章~顿时对angularjs理解了不少~

sheting 10-22 17:32发表

0 0 0 回复



评论：理解\$watch, \$apply 和 \$digest --- 理解数据绑定过程

angularjs基础教程 angularjs+ionic 移动app开发教程 免费视频教程来袭 百度网盘下载地址：<http://pan.baidu.com/s/1sjlwRt7> 密码：fy99

<http://www.ionic.wang/start-index.html>

论坛交流下载地址：<http://bbs.phonegap100.com/thread-2047-1-1.html>

htzhanglong 08-30 22:11发表

0 0 0 回复



评论：博主你好，假如我有2000~3000的数组，需要repeat，如何提升...
{{::user.name}}

geasscai 08-14 17:47发表

0 0 0 回复

« 1 2 3 »

作者信息



junyuecao

关注

会员 积分：2399 粉丝：28 关注：1 文章/评论：4

作者文章

08-09 12:30 [理解\\$watch, \\$apply 和 \\$digest --- 理解数据绑定过程](#)