mvvm学习&vue实践小结

In Web开发 on 2015年06月27日 by TAT.lqlongli view: 897

1 mvvm 学习

1.1 实现原理

mvvm类框架的实现原理不复杂，大致如下：

- 模板分析得到依赖的属性
- 通过某种变动监测手段监测这些依赖的属性
- 当属性变动的时候，触发相应的directive的处理逻辑即可

实际上，directive的处理逻辑不一定是对view进行操作，比如上报。但是，在mv的思想下，建议对view的操作都集中在directive里实现

从最核心上看，mv思想仅仅是一个观察者模式的具体应用于延展而已

1.2 核心技术点

1.2.1 模板分析

模板分析是比较基础的，凡是和view相关的基本都会涉及模板，这是原始资料，这里的关键点是**模板来源**的问题，实际上，它应该可以是**任何字符串**

这里暗示了框架需要一个模板解析器，不管这个解析器复杂还是简单，它都处于一个模式：【输入 –> 模板引擎 –> 输出】

于是，mvvm的模板解析器特点如下：

- 输入：任何符合规则的字符串
- 输出：需要监听的data.attr，directive，filter

在设计一个框架的时候，如果想要有更好的可扩展性，则

输入应该足够灵活，从来源上来说，模板可以是someDomHere.html()，也可以是动态输入，那就更有可适用性；从内容上来说，如果引擎可以识别更高级的语法，那就更有功能性

输出应该足够收敛，收敛的意思是有限并规则，像mvvm框架，最后出来的只是directive和filter，具体的处理都集中在这两个概念中，仅扩展这两个概念，即可对系统进行扩展

1.2.2 变动监测

在众多mvvm类框架中，实现变动监测有3种：

1. 门面方法setter，getter：比如knockout，q。限定可以变动的入口，并且让入口使用权放给用户来决定。
2. 利用defineProperty：比如vue，avalon。本质上也是setter，getter，但是没有把入口使用权放给用户来决定。
3. dirty check：比如angular。对angular的研究够多了，这里也不赘述了。

```
<span class="comment">//方式1 vs. 方式2</span>
<span class="comment">//方式1：</span>
vm.<span class="variable">$set</span>(aaa, <span class="number">1</span>)
vm._data.aaa = <span class="number">2</span>;    <span class="comment">//
vm.<span class="variable">$get</span>(aaa);        <span class="comment">/
```

```
7  <span class="comment">//方式2: </span>
8  vm.aaa = <span class="number">1</span>;          <span class="comment">//
9  vm._data.aaa = <span class="number">2</span>;    <span class="comment">//
10 vm.aaa;              <span class="comment">//1</span>
11
```

1.2.3 小结与延伸

对一类复杂并且常见的问题进行分析，解耦，抽象，在实践的过程中获得广泛的认可，那就形成了一种模式，mvvm也是一种模式，它不一定叫mvvm模式，这也不是笔者能决定的

对于这个模式的核心，笔者理解如下：系统根据**配置**得到了对某些**数据源**的某些**处理规则**，当数据源变动时就会引发相应的处理规则。模式的扩展是双向性，这由系统实现来决定，当符合某些规则的时候，可以对数据源进行更新。

我们跳出view的概念禁锢，联想实现一个监控系统，其实这个模式非常适合用在监控系统上面。

一般的监控系统的处理逻辑是：由收集源对监控数据进行收集整理，然后存储到数据库中，监控系统实时监控数据源，绘制实时的图线（反馈），当数据源发生了符合某些规则的变动时，就会触发相应的动作，比如报警。

如何实现这个系统，让系统具有更高的扩展性？参考mvvm模式，可以这样：

收集系统独立于监控系统，各不相同，暂且不论。监控系统通过某些配置文件取得需要监控的数据源与相应的处理逻辑规则，当数据源发生变动时触发相应的处理。

按照mvvm模式，进行一些抽象。

- 数据源不一定限定在数据库中，他可以在任何地方，只需要系统可以通过某些可配置的规则获取得到
- 处理规则进行抽象，让它更容易被扩展，比如发邮件，发短信，发微信，发qq消息等等

对应前端的mvvm框架，模板就是配置文件，directive就是处理规则，data对应数据源。

- 当系统需要新增一个数据源的时候，只需要更新配置文件，让系统读取即可启动数据监控
- 当需要新增一个处理规则的时候，可以通过一个热插拔的处理规则插件系统，扩展一个新的处理规则，再更新配置文件，系统即可接受新的处理规则

2 vue实践

vue介绍就不用了，太多资源了。这里讲述一下vue实践过程中的一些收获

2.1 组织结构

```
+ src
    +-- common
          +-- vue
                +-- coms
                +-- directives
                +-- filters
                +-- vue.js
                +-- vue.ext.js
    +-- pages
          +-- index
                +-- index.js
                +-- vue.ext.js
                +-- xxx.mixin.js
```

2.2 Vue扩展

vue的扩展非常方便，与vue相关的资源都放置在src/common/vue/下面，比如coms（组件），directive，filter

src/common/vue/vue.ext.js是对vue进行全局公共的扩展，对于所有页面共有的扩展放在这个文件下面，内容如下：

AngularJS经验分享

一箩筐的预加载技术

最新评论

alan： 这个真心强文，把redux的核心原理解释清楚了

微笑的鱼： 开启DEBUG模式后，需要运行react-native

微笑的鱼： 远程加载jsbundle文件，需要开启DEBUG模式，h

efenghuo： 保存的时候501，跨域问题，赶紧解决下。。。

年骚轻狂： 请问你们这个工具可以做重力感应的效果吗

友情链接

90ITer

AlloyTeam on Github.com

Cson's Room

dmfeel

Errorera's Blog

GTDLife | 时间管理行动家

HTML5 梦工场

HTML5中文学习网

imatlas

imyukin

Infinity World

iOSCAR

IT 宅

i在云端 – Kinvix

Too simple to blog

V2EX

W3C Plus

W3Cshare

W3CTech

Web 前端开发

Yukin

中国青少年开发者大会

前端开发网

前端观察

叶落为重生

周良粥凉

大猫の意淫筆記

時計坂一刻館三号室

淘宝 UED

爱思资源网

牛大拿_前端设计导航

百度FEX

胡尛睿丶可惜我是程序猿

腾讯 CDC

腾讯 FlashTeam

```
module.exports.install = function(Vue, options) {
    console.log('common Vue api install')
    //global api
    Vue.noop = function() {};
    Vue.noopArray = [];
    Vue.DIV = document.createElement('div');
    Vue.noopVue = new Vue({});
    //copy from zepto
    function extend(target, source, deep) {...}
    Vue.util.extend = function (target) {...}
    Vue.util.scrollTop = function(el, value) {...};
    Vue.util.createDOM = function(html) {...};
    Vue.util.newCom = function(c) {...}
    Vue.util.manage = function(id, v) {...}

    //instance api
    Vue.prototype.$unmount = function() {...}

    //directives
    Vue.directive('lazyload', require('common/vue/directives/lazyload'));
    Vue.directive('report', require('common/vue/directives/report'));

    //filters
    Vue.filter('formatTime', function(s, type) {...});
}
```

腾讯 GDC

腾讯 ISUX

腾讯 QQ客户端团队博客

腾讯大讲堂

腾讯游戏 TGideas

蓝色理想

雪泥鸿爪

麦时

可以看到，扩展vue库本身有4个扩展点：

- 扩展Vue库的全局方法/属性，方式：Vue.xxx = …
- 扩展Vue实例的方法/属性，方式：Vue.prototype = …
- 扩展directive，方式：Vue.directive(‘directiveName’, options);
- 扩展filter，方式：Vue.filter(‘filterName’, function(){});

对于页面单独需要的扩展，集中在src/pages/pageName/vue.ext.js里面，形式与全局的vue.ext.js一样

在实例化Vue的过程中也有许多可以扩展与优化的地方，在实践过程中只是应用了mixin功能，其他的可以慢慢深入

mixin的作用是在实例化Vue的时候混入一些功能，它可以混入许多特性，格式与实例化Vue时用到的option格式一样，比如index页面的mixin.js的内容如下：

```
return {
    methods: {
        onTitleClick: function(opt, isMore) {...},
        onCourseClick: function(item, opt) {...}
    }
}
```

这个mixin混入了两个方法，多个Vue实例共享的options可以放置到mixin中，从而避免了代码重，比如在实例化Vue的时候这样使用mixin：

```
vueObj = new Vue({
    el: '#js-cateCourse',
    template: tpl(),
    data: {
        list: data,
        picHeight: Math.ceil((parseInt(document.body.clientWidth) - 30)/2*125/222),
        picHeight2: Math.ceil((parseInt(document.body.clientWidth) - 20)*16/60),
        defaultPicSrc: __uri('/img/default-img.png')
    },
    methods: {
        onBannerClick: function(item) {
            var src = T.html.decodeHtml(item.banner_url.link), rId = item.banner_url.data_report;
            if (!src) return;
            src += (src.indexOf('?')<0?'?':'&') + 'from=Homepage';
            T.report({
                local: true,
                module: 'Y-MQ-Homepage',
                action: 'category_banner_clk',
                obj1: 'N_'+rId,
                obj2: 'V_3'
            });
            T.jump(src);
        },
        onBannerLoad: function(succ, img, src) {
            if (succ) {
                img.style.opacity = 1;
            } else {
                img.parentNode.style.display = 'none';
            }
        }
    },
    filters: {
        getCateName: function(cid) {
            return cateMap[cid];
        }
    },
    mixins: [require('index/mixin')]
});
```

可以看到mixin是个数组，因此可以同时使用多个mixin

实际上这里的mixin主要不是为了避免代码重复（实践的时候只是这样用），mixin是一种模式，一个mixin内聚了实现一项功能的方法/属性集合，在定义/生成实例的时候，通过混入mixin就可以让该实例拥有某项功能，归根结底是组合vs继承问题的产物

2.3 vue组件插入问题

2.3.1 首屏

对于首屏的vue组件，直接把模板放在主页面中即可，初始化的时候只需要把el参数传入，Vue就会用el的html作为模板来初始化Vue实例:

```
!cateVue && (cateVue = new Vue({
    el: '#js-category',
    methods: {
        onCateClick: function(cid) {...}
    }
}));
```

这里需要注意的是在模板中不能使用{{}}，否则在还没初始化之前，页面会显示奇怪的东西，比如:

```
<p>hello, {{name}}</p>       <!--初始化前，页面会直接展示hello, {{name}}-->
<img src="{{imgSrc}}" />     <!--初始化前，会报错
<!--正确的写法：-->
<p v-text="'hello, '+name"></p>
<img v-attr="src: imgSrc" />
```

{{}} 只是一个语法糖，不建议使用

2.3.2 非首屏

对于非首屏的组件，使用vue的方式和原始方式差不多，先生成节点，然后append，譬如:

```
this.vueObj = new Vue({
    el: function() {
        return Vue.util.createDOM('<a class="goToTop" href="javascript:void(0)"
            'v-class="z-hide: hide, z-showBlock: showBlock, z-show: show"></a>')
    },
    data: {"hide": 0...},
    methods: {
        onClick: function() {...},
        onWebkitAnimationEnd: function(e) {...}
    }
});

this.vueObj.$appendTo(this.box, null, false);
```

el参数可以接收query string，也可以直接是一个dom节点，如果是dom节点则直接编译dom的内容。如果dom节点不在文档树中，则利用vueObj.$appendTo方法将vue实例的根节点插入到文档树中

上面这种方式是在页面中没有组件的【坑】的情况下使用的，如果页面为组件留了【坑】，比如：

```
1 <section <span class="keyword">class</span>=<span class="string">"hotReco
2
```

那么，我们可以这样初始化vue实例：

```
vueObj = new Vue({
    el: '#js-hotRecord',
    template: tpl(),
    data: {...},
    mixins: [require('index/mixin')]
});
```

利用template参数传入模板，并指定el，那么vue实例在初始化之后就会自动把内容插入到el中

通过vue实现组件的主要核心也就这些，更方便的组件写法也只是对这些进行封装

2.4 自定义 directive

在vue中自定义directive是非常简单明了的，要自定义一个directive，可以注册3个钩子函数：

- bind：仅调用一次，当指令第一次绑定元素的时候。
- update：第一次调用是在 bind之后，用的是初始值；以后每当绑定的值发生变化就会被调用，新值与旧值作为参数。
- unbind：仅调用一次，当指令解绑元素的时候。

下面简单介绍一个自定义directive——lazyload：

```
1  <span class="keyword">function</span> addSrc(){}
2  <span class="keyword">function</span> load(){}
3
4  module.exports = {
5      bind: <span class="keyword">function</span>() {
6          <span class="keyword">if</span> (!hasBind) { <span class="comment
7              hasBind = <span class="keyword">true</span>;
8              (document.querySelector(<span class="string">'.z-scroller'</s
9          }
10         <span class="comment">//这里也可以使用data属性来获取</span>
11         <span class="keyword">var</span> defaultSrc = <span class="keywo
12         <span class="keyword">if</span> (defaultSrc) addSrc(<span class="
13     },
14     update: <span class="keyword">function</span>(src) {
15         <span class="comment">//directive初始化时，会调用一次bind和update，bi
16         <span class="comment">//因此只能在update这里拿到需要lazyload的src</sp
17         <span class="comment">//lazyload不允许修改src，这里限制只会执行一次upd
18         <span class="comment">//注：接受src改变可以实现，只是需要一些复杂的处理，
19         <span class="keyword">if</span> (<span class="keyword">this</span>.par
20         <span class="keyword">this</span>.init = <span class="keyword">tr
21
22         <span class="comment">//如果图片已经加载了，就不需要注册了，这里也可以使用
23         <span class="keyword">var</span> isLoad = parseInt(<span class="k
24         <span class="keyword">if</span> (isLoad) <span class="keyword">re
25
26         <span class="comment">//注册需要lazyload的图片</span>
27         <span class="keyword">list</span>[index++] = <span class="keywor
28         <span class="keyword">list</span>[index++] = src;
29     }
30     <span class="comment">//这里有一个最大的问题：由于有local的存在，会创建两个一
```

```
31      <span class="comment">//按理说应该定义一个unbind，但是在unbind中找到并除掉l
32      <span class="comment">//因此在load函数里面做了一个处理：如果发现需要lazyloa
33      <span class="comment">//通过这个直接省掉了unbind函数</span>
34  };
35
```

自定义filter也很简单，只是定义一个处理函数而已，这里就不多介绍了

2.5 实践过程中的痛点与小技巧

2.5.1 没有事件代理

用习惯了事件代理，突然没有了会有点不习惯，但是回头想想，事件代理真的很重要吗？还是说我们只是习惯了事件代理而已？

通过vue注册相同的事件并不费事。另一个问题，只要事件不多，大约不超过50，100，也不至于耗掉很大的内存，因此有时候还真不需要事件代理。如果真的需要，也只是实现一个contain方法而已

2.5.2 没有if-else的奇怪

最初看到下面的代码真的会觉得很奇怪

```
1  <h3 v-<span class="keyword">if</span>=<span class="string">"hasTitle"</spa
2  <p v-<span class="keyword">if</span>=<span class="string">"!hasTitle"</spa
3
```

2.5.3 单值

虽然vue有语法解析器，可以在directive的值中使用表达式，但是当出现一个复杂的表达式时，会污染模板，让代码可读性变得很差，又或者，表达式完成不了这个任务的时候。

因此，在mvvm实践的过程中，深深地发现，利用单值（最多只用一个?:表达式）来写模板会让代码变得很清晰，更加可读，增加代码的可维护性，而且这也更符合mvvm的核心思想：f(state) = view

有些库连语法解析器都没有，比如q，但也能很好的工作。

那么，复杂的操作放在哪里呢？

- 对于不会变的值来说，也就是常量，要在初始化之前完成处理
- 对于会变的值来说，把复杂的操作放在filter里面，在filter里面不仅可以进行复杂处理，甚至可以同时应用到其他字段，这不完全等同于computed attribute

2.5.4 替代 $(document).on

用jquery/zepto的时候，习惯了用$(document).on来充当一个全局的事件代理，在使用vue的时候，需要抛弃zepto，因此需要解决这个问题

因为vue实例本身就有event功能，因此这里解决的办法是创建一个全局的空vue对象，把它作为全局的事件代理：

```
1  <span class="comment">//common/vue/vue.ext.js 回头看前面对该文件的介绍可以看到这
2  Vue.noopVue = <span class="keyword">new</span> Vue({});
3
4  <span class="comment">//a.js</span>
5  Vue.noopVue.<span class="variable">$on</span>(<span class="string">'someEv
6
7  <span class="comment">//b.js</span>
8  Vue.noopVue.<span class="variable">$emit</span>(<span class="string">'some
9
```

3 总结

虽然，最后在付出产出比权衡中放弃了对现有项目的vue改造，但是这并不妨碍我们研究mvvm类框架

mvvm模式还是值得我们去深入学习的，而在实践中，我们也能学习到许多

用一种不一样的思想和思维去开发的体验也会令我们在看待问题，处理问题的道路上有所收获

最后，期待q的发展，我已经整装待发了哟

原创文章转载请注明：

转载自AlloyTeam：http://www.alloyteam.com/2015/06/mvvm-xue-xi-vue-shi-jian-xiao-jie/

分享到: 0

喜欢

2 条评论 最新 最早 最热

tcdona

https://github.com/imweb/Q.js 用 set get 兼容的想法很贴心
再啰嗦一句，有空求把代码换成截图吧

7月2日 回复 顶 转发

tcdona

感谢手把手教写vue!
“页面为组件留了【坑】 ”这里的截图看不太懂，而且标签好像没闭合，而且弱弱说一句高亮好难受

7月2日 回复 顶 转发

iM丁丁 帐号管理

说点什么吧...

分享到: 发布

Web前端 腾讯AlloyTeam Blog | 愿景: 成为地球卓越的Web团队！正在使用多说

喜欢

2 条评论 最新 最早 最热

tcdona

https://github.com/imweb/Q.js 用 set get 兼容的想法很贴心
再啰嗦一句，有空求把代码换成截图吧

7月2日 回复 顶 转发

tcdona

感谢手把手教写vue!
“页面为组件留了【坑】 ”这里的截图看不太懂，而且标签好像没闭合，而且弱弱说一句高亮好难受

7月2日 回复 顶 转发

iM丁丁 帐号管理

说点什么吧...

分享到: 发布

Web前端 腾讯AlloyTeam Blog | 愿景: 成为地球卓越的Web团队！正在使用多说