

Vue + webpack 项目实践

最近在内部项目中做了一些基于 vue + webpack 的尝试，在小范围和同事们探讨之后，还是蛮多同学认可和喜欢的，所以通过 blog 分享给更多人。

首先，我会先简单介绍一下 vue 和 webpack：

(当然如果你已经比较熟悉它们的话前两个部分可以直接跳过)

介绍 vue



[Vue.js](#) 是一款极简的 mvvm 框架，如果让我用一个词来形容它，就是“轻·巧”。如果用一句话来描述它，它能够集众多优秀逐流的前端框架之大成，但同时保持简单易用。废话不多说，来看几个例子：

```
<script src="vue.js"></script>
```

```
<div id="demo">
  {{message}}
```

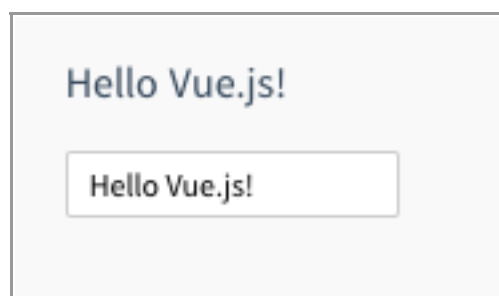
```

<input v-model="message">
</div>

<script>
  var vm = new Vue({
    el: '#demo',
    data: {
      message: 'Hello Vue.js!'
    }
  })
</script>

```

首先，代码分两部分，一部分是 html，同时也是视图模板，里面包含一个值为 `message` 的文本和一个相同值的输入框；另一部分是 `script`，它创建了一个 `vm` 对象，其中绑定的 `dom` 结点是 `#demo`，绑定的数据是 `{message: 'Hello Vue.js!'}`，最终页面的显示效果就是一段 `Hello Vue.js` 文本加一个含相同文字的输入框，更关键的是，由于数据是双向绑定的，所以我们修改文本框内文本的同时，第一段文本和被绑定的数据的 `message` 字段的值都会同步更新——而这底层的复杂逻辑，`Vue.js` 已经全部帮你做好了。



再多介绍一点

我们还可以加入更多的 `directive`，比如：

```

<script src="vue.js"></script>

<div id="demo2">
  <img title="{{name}}" alt="{{name}}" v-attr="src: url">
  <input v-model="name">
  <input v-model="url">
</div>

<script>
  var vm = new Vue({
    el: '#demo2',
    data: {
      name: 'taobao',
      url: 'https://www.taobao.com/favicon.ico'
    }
  })
</script>

```

这里的视图模板加入了一个 `` 标签，同时我们看到了 2 个特性的值都写作了 `{{name}}`。这样的话，图片的 `title` 和 `alt` 特性值就都会被绑定为字符串 `'taobao'`。

如果想绑定的特性是像 `img[src]` 这样的不能在 html 中随意初始化的（可能默认会产生预期外的网络请求），没关系，有 `v-attr="src: url"` 这样的写法，把被绑定的数据里的 `url` 同步过来。

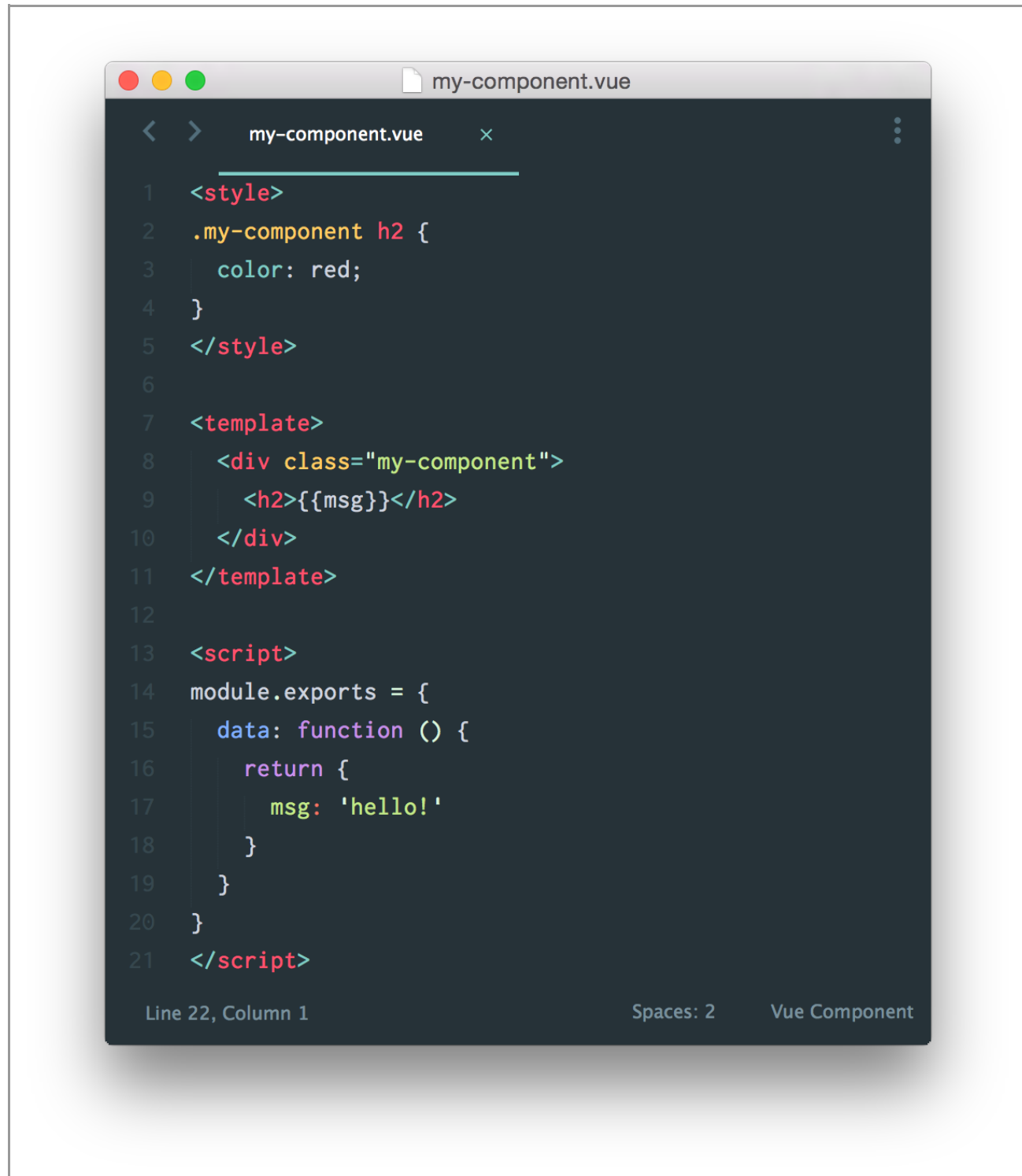
没有介绍到的功能还有很多，推荐大家来我(发起并)翻译的[Vue.js 中文文档](#)

web 组件化

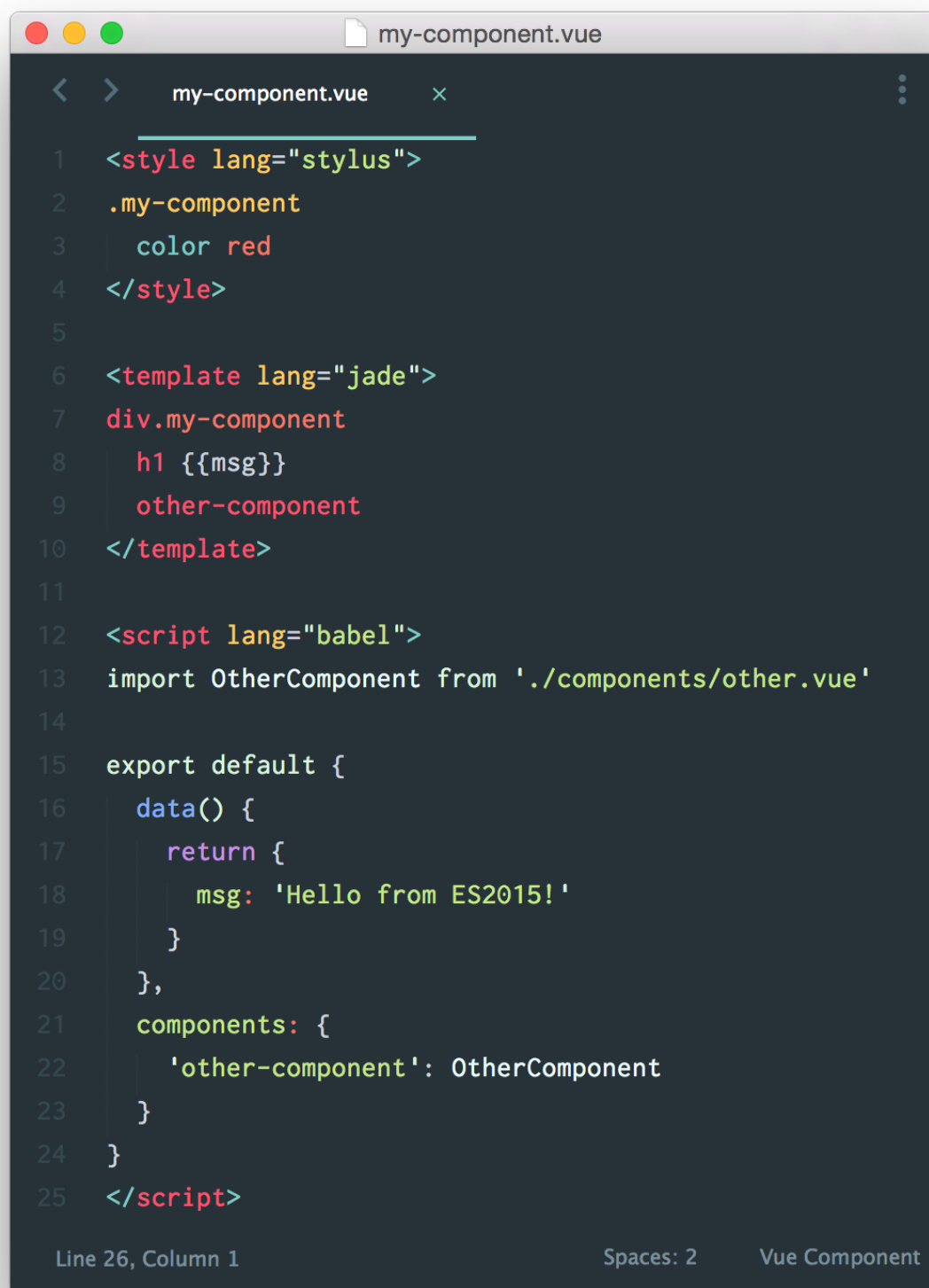
最后要介绍 Vue.js 对于 web 组件化开发的思考和设计

如果我们要开发更大型的网页或 web 应用，web 组件化的思维是非常重要的，这也是今天整个前端社区长久之不衰的话题。

Vue.js 设计了一个 *.vue 格式的文件，令每一个组件的样式、模板和脚本集成了一整个文件，每个文件就是一个组件，同时还包含了组件之间的依赖关系，麻雀虽小五脏俱全，整个组件从外观到结构到特性再到依赖关系都一览无余：



并且支持预编译各种方言：



```
1 <style lang="stylus">
2   .my-component
3     color red
4 </style>
5
6 <template lang="jade">
7   div.my-component
8     h1 {{msg}}
9     other-component
10 </template>
11
12 <script lang="babel">
13   import OtherComponent from './components/other.vue'
14
15   export default {
16     data() {
17       return {
18         msg: 'Hello from ES2015!'
19       }
20     },
21     components: {
22       'other-component': OtherComponent
23     }
24   }
25 </script>
```

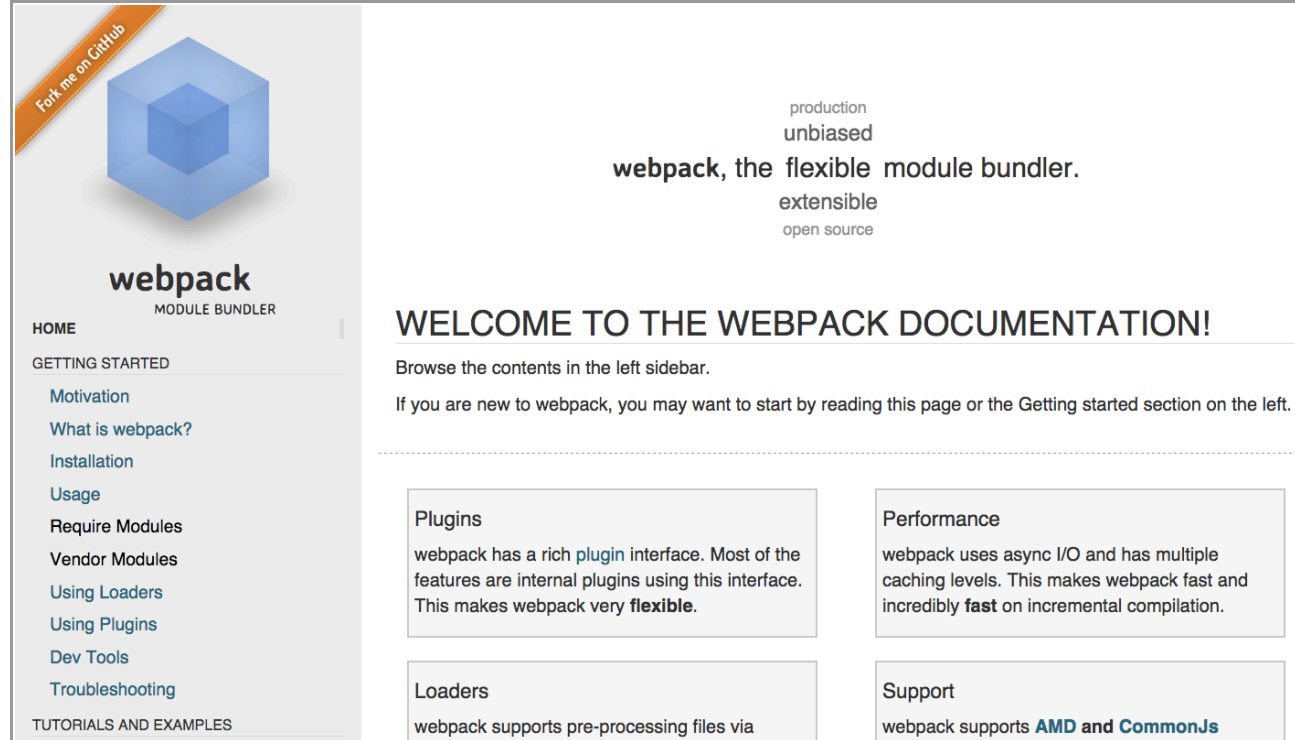
Line 26, Column 1 Spaces: 2 Vue Component

这样再大的系统、在复杂的界面，也可以用这样的方式庖丁解牛。当然这种组件的写法是需要编译工具才能最终在浏览器端工作的，下面会提到一个基于 webpack 的具体方案。

小结

从功能角度，`template`, `directive`, `data-binding`, `components` 各种实用功能都齐全，而 `filter`, `computed var`, `var watcher`, `custom event` 这样的高级功能也都洋溢着作者的巧思；从开发体验角度，这些设计几乎是完全自然的，没有刻意设计过或欠考虑的感觉，只有个别不得已的地方带了自己框架专属的 `v-` 前缀。从性能、体积角度评估，Vue.js 也非常有竞争力！

介绍 webpack



[webpack](#) 是另一个近期发现的好东西。它主要的用途是通过 CommonJS 的语法把所有浏览器端需要发布的静态资源做相应的准备，比如资源的合并和打包。

举个例子，现在有个脚本主文件 `app.js` 依赖了另一个脚本 `module.js`

```
// app.js
var module = require('./module.js')
... module.x ...

// module.js
exports.x = ...
```

则通过 `webpack app.js bundle.js` 命令，可以把 `app.js` 和 `module.js` 打包在一起并保存到 `bundle.js`

同时 webpack 提供了强大的 loader 机制和 plugin 机制，loader 机制支持载入各种各样的静态资源，不只是 js 脚本、连 html, css, images 等各种资源都有相应的 loader 来做依赖管理和打包；而 plugin 则可以对整个 webpack 的流程进行一定的控制。

比如在安装并配置了 `css-loader` 和 `style-loader` 之后，就可以通过 `require('./bootstrap.css')` 这样的方式给网页载入一份样式表。非常方便。

webpack 背后的原理其实就是把所有的非 js 资源都转换成 js (如把一个 css 文件转换成“创建一个 style 标签并把它插入 document”的脚本、把图片转换成一个图片地址的 js 变量或 base64 编码等)，然后用 CommonJS 的机制管理起来。一开始对于这种技术形态我个人还是不太喜欢的，不过随着不断的实践和体验，也逐渐习惯并认同了。

最后，对于之前提到的 Vue.js，作者也提供了一个叫做 `vue-loader` 的 [npm 包](#)，可以把 `*.vue` 文件转换成 webpack 包，和整个打包过程融合起来。所以有了 Vue.js、webpack 和 `vue-loader`，我们自然就可以把它们组合在一起试试看！

项目实践流程

回到正题。今天要分享的是，是基于上面两个东西：Vue.js 和 webpack，以及把它们串联起来的 `vue-loader`

Vue.js 的作者以及提供了一个基于它们三者的[项目示例](#)。而我们的例子会更贴近实际

工作的场景，同时和团队之前总结出来的项目特点和项目流程相吻合。

目录结构设计

- <components> 组件目录，一个组件一个 .vue 文件
 - a.vue
 - b.vue
- <lib> 如果实在有不能算组件，但也不来自外部 (tnpm) 的代码，可以放在这里
 - foo.css
 - bar.js
- <src> 主应用/页面相关文件
 - app.html 主 html
 - app.vue 主 vue
 - app.js 通常做的事情只是 `var Vue = require('vue'); new Vue(require('./app.vue'))`
- <dist> (ignored)
- <node_modules> (ignored)
- gulpfile.js 设计项目打包/监听等任务
- package.json 记录项目基本信息，包括模块依赖关系
- README.md 项目基本介绍

打包

通过 gulpfile.js 我们可以设计整套基于 webpack 的打包/监听/调试的任务

在 [gulp-webpack](#) 包的官方文档里推荐的写法是这样的：

```
var gulp = require('gulp');
var webpack = require('gulp-webpack');
var named = require('vinyl-named');
gulp.task('default', function() {
  return gulp.src(['src/app.js', 'test/test.js'])
    .pipe(named())
    .pipe(webpack())
    .pipe(gulp.dest('dist/'));
});
```

我们对这个文件稍加修改，首先加入 vue-loader

```
tnpm install vue-loader --save

.pipe(webpack({
  module: {
    loaders: [
      { test: /\.vue$/, loader: 'vue' }
    ]
  }
}))
```

其次，把要打包的文件列表从 gulp.src(...) 中抽出来，方便将来维护，也有机会把

这个信息共享到别的任务

```
var appList = ['main', 'sub1', 'sub2']

gulp.task('default', function() {
  return gulp.src(mapFiles(appList, 'js'))
    ...
})

/**
 * @private
 */
function mapFiles(list, extname) {
  return list.map(function (app) {return 'src/' + app + '.' + extname})
}
}
```

现在运行 gulp 命令，相应的文件应该就打包好并生成在了 dist 目录下。然后我们在 src/*.html 中加入对这些生成好的 js 文件的引入：

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="utf-8">
  <title>Main</title>
</head>
<body>
  <div id="app"></div>
  <script src="../../dist/main.js"></script>
</body>
</html>
```

用浏览器打开 src/main.html 这时页面已经可以正常工作了

加入监听

监听更加简单，只要在刚才 webpack(opt) 的参数中加入 watch: true 就可以了。

```
.pipe(webpack({
  module: {
    loaders: [
      { test: /\.vue$/, loader: 'vue' }
    ]
  },
  watch: true
}))
```

当然最好把打包和监听设计成两个任务，分别起名为 bundle 和 watch：

```
gulp.task('bundle', function() {
  return gulp.src(mapFiles(appList, 'js'))
    .pipe(named())
    .pipe(webpack(getConfig()))
    .pipe(gulp.dest('dist/'))
})

gulp.task('watch', function() {
  return gulp.src(mapFiles(appList, 'js'))
```



```

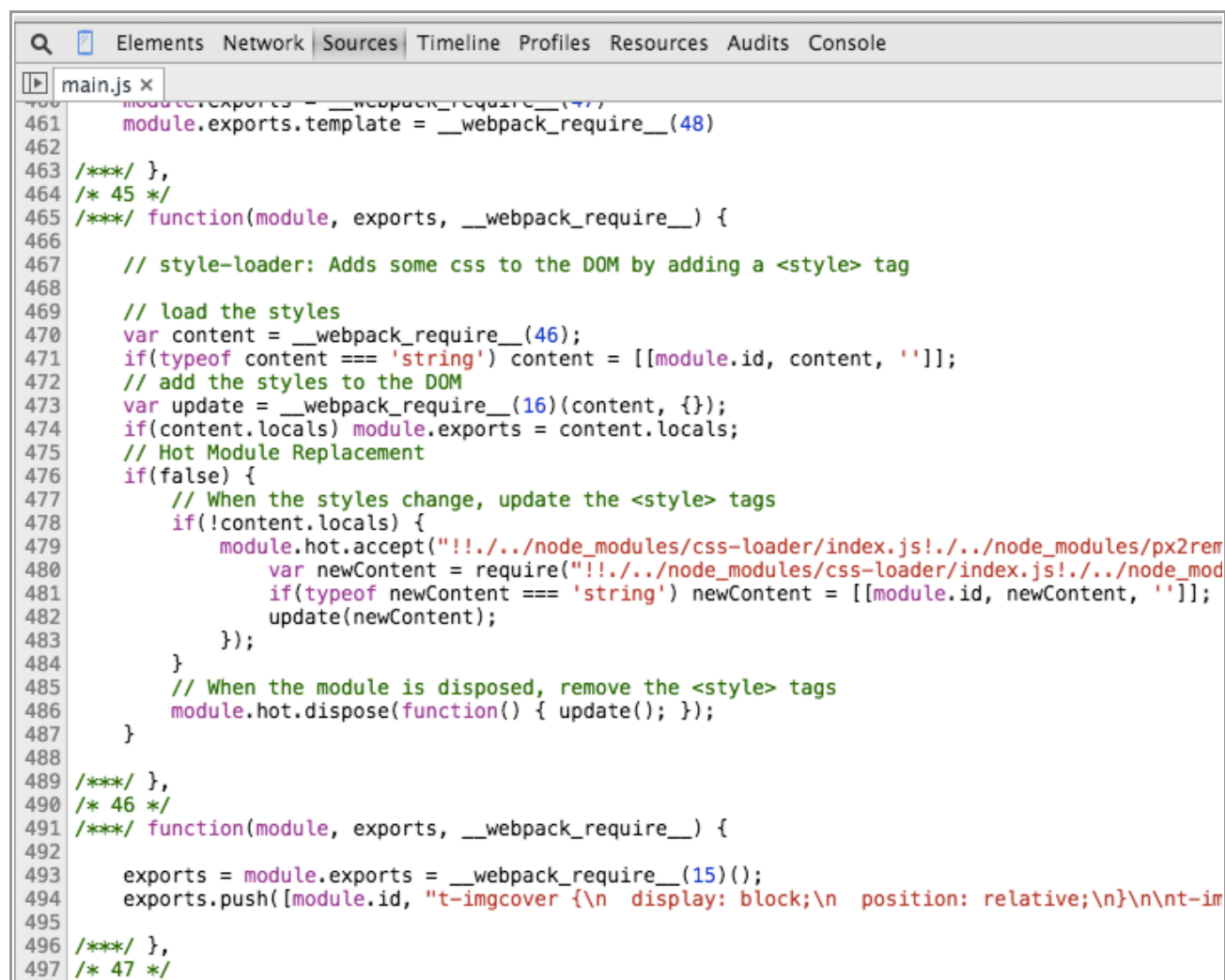
    .pipe(named())
    .pipe(webpack(getConfig({watch: true})))
    .pipe(gulp.dest('dist/'))
  })

  /**
   * @private
   */
  function getConfig(opt) {
    var config = {
      module: {
        loaders: [
          { test: /\.vue$/, loader: 'vue' }
        ]
      }
    }
    if (!opt) {
      return config
    }
    for (var i in opt) {
      config[i] = opt[i]
    }
    return config
  }
}

```

现在你可以不必每次修改文件之后都运行 `gulp bundle` 才能看到最新的效果，每次改动之后直接刷新浏览器即可。

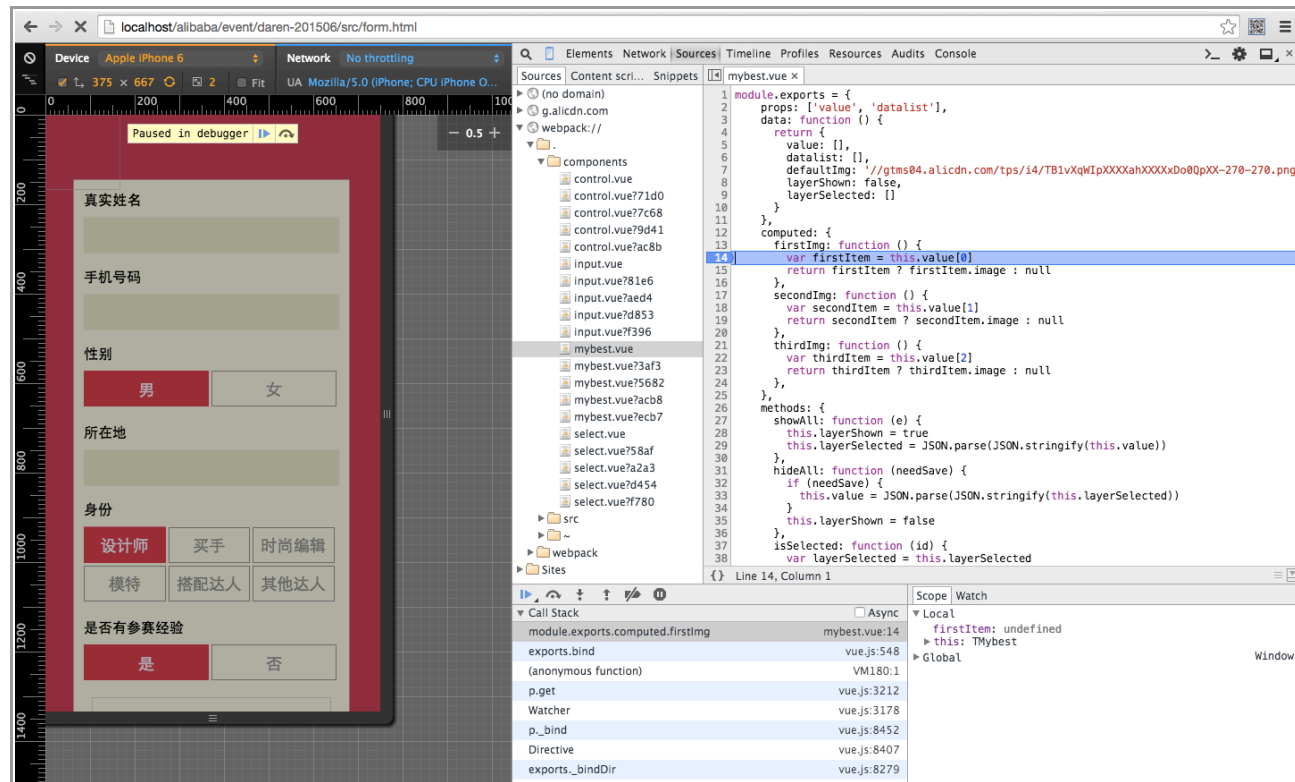
调试



打包好的代码已经不那么易读了，直接在这样的代码上调试还是不那么方便的。这个时候，webpack + vue 有另外一个现成的东西：source map 支持。为 webpack 加入这个配置字段 `devtool: 'source-map'`：


```
var config = { module: { loaders: [ { test: /\.vue$/, loader: 'vue' } ] }, devtool: 'source-map' }
```

再次运行 `gulp bundle` 或 `gulp watch` 试试看，是不是开发者工具里 debug 的时候，可以追踪断点到源代码了呢：)



完整的 javascript 代码如下：

```
var gulp = require('gulp')
var webpack = require('gulp-webpack')
var named = require('vinyl-named')

var appList = ['main']

gulp.task('default', ['bundle'], function() {
  console.log('done')
})

gulp.task('bundle', function() {
  return gulp.src(mapFiles(appList, 'js'))
    .pipe(named())
    .pipe(webpack(getConfig()))
    .pipe(gulp.dest('dist/'))
})

gulp.task('watch', function() {
  return gulp.src(mapFiles(appList, 'js'))
    .pipe(named())
    .pipe(webpack(getConfig({watch: true})))
    .pipe(gulp.dest('dist/'))
})

/**
 * @private
 */
function getConfig(opt) {
  var config = {
```

```

    module: {
      loaders: [
        { test: /\.vue$/, loader: 'vue'}
      ]
    },
    devtool: 'source-map'
  }
  if (!opt) {
    return config
  }
  for (var i in opt) {
    config[i] = opt[i]
  }
  return config
}

/**
 * @private
 */
function mapFiles(list, extname) {
  return list.map(function (app) {return 'src/' + app + '.' + extname})
}
}

```

最后，杜拉拉不如紫罗兰

做出一个 *vue + webpack* 的 *generator*，把这样的项目体验分享给更多的人。目前我基于团队内部在使用的轻量级脚手架工具写了一份名叫 *just-vue* 的 *generator*，目前这个 *generator* 还在小范围试用当中，待比较成熟之后，再分享出来

总结

其实上面提到的 *just-vue* 脚手架已经远不止文章中介绍的东西了，我们在业务落地的“最后一公里”做了更多的沉淀和积累，比如自动图片上传与画质处理、**rem**单位自动换算、服务端/客户端/数据埋点接口的梳理与整合、自动化 **htmlone** 打包与 **awp** 发布等等。它们为支持业务的开发者提供了更简单高效的工作体验。篇幅有限，更多内容我也希望将来有机会再多分享出来。

最后再次希望大家如果有兴趣的话可以来玩一下，无线前端组内的同学我都愿意提供一对一入门指导：)

Just Vue!

已有 16 条评论 »

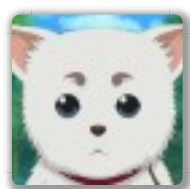


hiloooooo

[June 25th, 2015 at 03:18 pm](#)

是真爱

[回复](#)



szmtcjm

[June 26th, 2015 at 01:08 pm](#)

为什么我们内部项目这么少！

[回复](#)



傅小黑

[June 27th, 2015 at 09:04 am](#)

还是不太喜欢require(css)哈哈



囧克斯

[June 27th, 2015 at 12:00 pm](#)

你看我也纠结过这个问题，现在也逐渐接受了 <http://weibo.com/1712131295/CIWOLjrho>



Randy

[November 9th, 2015 at 11:55 am](#)

用多了之后你会发现 require css 是很方便的东西，因为用 style-loader 和其它 loader 可以让你免于写 gulp 去 process 你的 stylesheet。

[回复](#)

[回复](#)

[回复](#)



dazhenhan

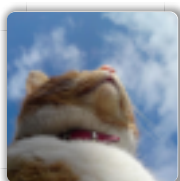
[July 15th, 2015 at 05:05 pm](#)

发现一个小错误，在getConfig方法中，for 中取值漏了i，应为

```
for (var i in opt) {  
  config[i] = opt[i]  
}
```

是吧？

[回复](#)



think2011

[July 23rd, 2015 at 12:17 pm](#)

好大的版面文字

[回复](#)

今天使用Vue+webpack对项目进行了重构 | Raito_MH的世界

[August 5th, 2015 at 12:12 am](#)

[...]在这几个月接触的项目中，都使用vue.js这个框架来进行开发，可以说是各种便利，然后今天无意间在微博上看到了勾股大大的这篇博文《Vue + webpack 项目实践》，可以说是受益匪浅，于是对目前正在做的web版本进行了重构。[...]

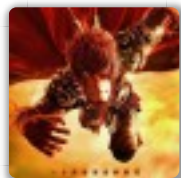
[回复](#)



tcdona

[August 5th, 2015 at 04:38 pm](#)

报错了，请问谁成功了吗



tcdona

[August 5th, 2015 at 05:10 pm](#)

发现是我项目文件的问题，config 错误确实存在。

最后希望能直接提供zip 包demo，避免一些错误~

文章赞!

[回复](#)

[回复](#)

[淘宝无线前端图片处理流程 | 潮流前端](#)

[August 10th, 2015 at 04:38 pm](#)

[...]我在这个过程中，融入了之前一段时间集中实践的 vue 和 webpack 的工程体系，在 vue 的基础上进行组件化开发，在 webpack 的基础上管理资源打包、集成和发布，最终合并在了最新的 just-vue 的 adam template 里面。[...]

[回复](#)



naux

[August 14th, 2015 at 11:50 am](#)

.vue 文件里的css经过webpack都是inline style，如何把vue组件样式抽取出来，合并到一个css文件中

[回复](#)

[手机淘宝前端的图片相关 workflows 梳理! | 加速会](#)

[August 23rd, 2015 at 04:54 pm](#)

[...]我在这个过程中，融入了之前一段时间集中实践的 vue 和 webpa

ck 的工程体系，在 vue 的基础上进行组件化开发，在 webpack 的基础上管理资源打包、集成和发布，最终合并在了最新的 just-vue 的 adam template 里面。[...]

[回复](#)



梁晓晨

[September 2nd, 2015 at 03:26 pm](#)

有了webpack还配合gulp。我也是醉了

[回复](#)

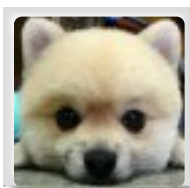


测试用户

[September 7th, 2015 at 10:25 am](#)

“里面包含一个值为 message 的文本何一个相同值的输入框”
“何”to“和”。

[回复](#)



Aaron

[September 19th, 2015 at 09:55 pm](#)

我个人也尝试了vue-gulp-webpack的方案
为什么用gulp
配合browser-sync做代码注入与浏览器刷新
gulp watch监控文件变动
webpack提供监控，但是很慢，消耗比较大
提供一个简单的配置：
<https://github.com/JsAaron/vue-gulp-webpack-example>

[回复](#)

添加新评论 »

称呼

电子邮件

网站

提交评论

(请至少包含一个汉字，且汉字不能比日本字少)

问卷

- 暂无

广告

我发起的开源项目

[H5Slides](#)

[Zorro](#)

欢迎了解

最新文章

- [Vue.js 1.0.0 发布了!](#)
- [如何成为一名卓越的前端工程师](#)
- [手机淘宝前端的图片相关工作流程梳理](#)
- [如何让办公室政治最小化](#)
- [Vue.js 源码学习笔记](#)
- [从原型到发布——“团队时间线” 1.0 开发心得](#)
- [Vue + webpack 项目实践](#)
- [用 Koa 写服务体验](#)
- [webcomponents 笔记 之 配置管理](#)
- [14](#), [15](#)

最近回复

- [tty228](#): 一直安装不成功-- 不知道是不是百度统计换了新版的原因, 审视元素能看到h.gif但是统计一直提示检...
- [tty228](#): 谢谢博主, 拿去用了~~
- [cxczy](#): 总结的很好, Vue如此小巧却又这么多优点
- [Randy](#): 用多了之后你会发现 require css 是很方便的东西, 因为用 style-loader 和其它...
- [elevensky](#): 大爱
- [云库网](#): 字体有点大, css命名还是标准加习惯为好
- [影乐](#): 名字好有意思
- [常某某](#): = =我总觉得你的网站设计很奇怪。或许是相邻两块之间并无色差。而且宽度很小。总感觉两边空出很多来。特...
- [小四](#): 棒棒哒
- [SuperZhang](#): 你的字确实很大....我也是用手动添加的. 我是来保持队形的

归档

- [October 2015](#)
- [August 2015](#)
- [July 2015](#)
- [June 2015](#)
- [March 2015](#)
- [January 2015](#)
- [October 2014](#)
- [September 2014](#)
- [January 2014](#)
- [December 2013](#)

- [October 2013](#)
- [September 2013](#)
- [July 2013](#)
- [June 2013](#)
- [May 2013](#)
- [March 2013](#)
- [February 2013](#)
- [January 2013](#)
- [December 2012](#)
- [November 2012](#)
- [September 2012](#)
- [August 2012](#)
- [July 2012](#)
- [June 2012](#)
- [May 2012](#)
- [April 2012](#)
- [March 2012](#)

其它

- [登录](#)
- [Valid XHTML](#)
- [Typecho](#)

链接

- [傲游浏览器](#)
- [我的Github](#)

[囧克斯](#) is powered by [Typecho](#))))

文章 [RSS](#) and 评论 [RSS](#) 我是百度统计: 

{"theme": "我被拍平了", "designer": "@勾三股四"}