

搜索.....

首页 笔记首页 ANDROID ES6 教程 排序算法 程序员人生 程序员笑话 编程技术 极客时间

Vue2.0 新手入门 — 从环境搭建到发布

分类 **编程技术**



Jinkey原创

感谢 [showonne](#)、[yubang](#) 技术指导

Demo 地址:

<http://demo.jinkey.io/vue2>

源码:

<https://github.com/Jinkeycode/vue2-example>

Vue2 教程: <https://www.runoob.com/vue2/vue-tutorial.html>

什么是 Vue

Vue 是一个前端框架，特点是

数据绑定

比如你改变一个输入框 Input 标签的值，会**自动同步**更新到页面上其他绑定该输入框的组件的值



教程列表

ADO 教程 Ajax 教程 Android 教

Angular2 AngularJS AppML 教

ASP 教程 ASP.NET Bootstrap

Bootstrap4 C 教程 C# 教程

C++ 教程 CSS 参考 CSS 教程

CSS3 教程 Django 教 Docker 教

DTD 教程 Eclipse 教 Firebug 教

Font Foundation Git 教程

Go 语言教 Google 地 Highcharts

HTML HTML 参考 HTML 字符

HTML 教程 HTTP 教程 ionic 教程

iOS 教程 Java 教程 JavaScript

Javascript jQuery jQuery

jQuery UI jQuery 教 JSON 教程

JSP 教程 Kotlin 教程 Linux 教程

Lua 教程 Markdown Maven 教

Memcache MongoDB MySQL 教

Node.js 教 NumPy 教 Perl 教程

PHP 教程 PostgreSQL Python 3

Python 基 RDF 教程 React 教程

Redis 教程 RSS 教程 Ruby 教程

Scala 教程 Servlet 教 SOAP 教程

SQL 教程 SQLite 教 SVG 教程

SVN 教程 Swift 教程 TCP/IP 教

TypeScript VBScript Vue.js 教程

W3C 教程 Web WSDL 教

XLink 教程 XML DOM XML

XML 教程 XPath 教程 XQuery 教

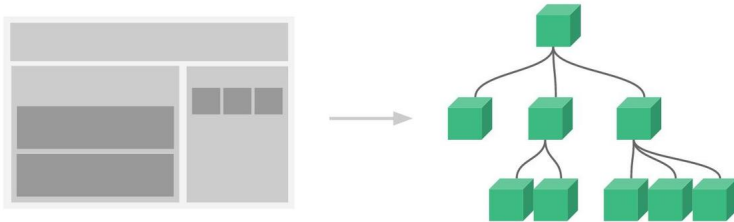
XSLFO 教 XSLT 教程 正则表达式

测验 浏览器 网站品质

网站建设指 网站服务器 设计模式

组件化

页面上小到一个按钮都可以是一个单独的文件.vue, 这些小组件直接可以像乐高积木一样通过互相引用而组装起来



Vue2.0 推荐开发环境

Homebrew

Mac 系统下的包管理器
类似于 Linux 的 apt-get , Windows 的控制面板-安装删除应用程序

Node.js

Javascript运行环境(runtime), 不同系统直接不能直接运行各种编程语言
Runtime 类似于各种国际会议上的 同声传译

npm

Nodejs 下的包管理器
类似于 Mac 下的 Homebrew

微信公众号 jinkey-love
Github JinkeyCode

webpack

Vue 的组件都是通过 .vue 或者 像微信小程序的 .wxml 和 .wxss 等自定义的组件都无法被用户端的各种浏览器解析, 需要被翻译和打包成 .js 文件

vue-cli

用来生成模板的 Vue 工程, 相当于按照设计好的图来盖房子
微信小程序一开始也会以可视化界面的方式问你是否创建示例工程, 其实就是封装了类似的脚手架

Homebrew 1.0.6(Mac)、Node.js 6.7.0、npm 3.10.3、webpack 1.13.2、vue-cli 2.4.0、Atom 1.10.2

环境安装

Mac OS系统安装 brew

打开终端运行以下命令:

```
/usr/bin/ruby -e "$(curl -fsSL https://raw.githubusercontent.com/Homebrew/install/master/install)"
```

Mac OS系统安装 nodejs

```
brew install nodejs
```

用 npm install npm@3.10.3 更新 npm 版本报错:

(node:42) fs: re-evaluating native module sources is not supported.



解决办法:

在官网下载6.7.0的安装包再安装一次(刚刚相当于帮你配置好环境变量, 现在再安装一次升级到最新的 npm)

好像以前官网的安装包不会自动配置环境变量的, 由于我电脑上之前安装过 nodejs 所以环境变量已经配置好了, 不知道现在的安装包会不会自动配置环境变量。

Windows 下直接下载安装包即可

linux 下可以使用 apt-get (ubuntu) 或 yum (centos) 命令安装。

具体可参考: <http://www.runoob.com/nodejs/nodejs-install-setup.html>

获取nodejs模块安装目录访问权限

```
sudo chmod -R 777 /usr/local/lib/node_modules/
```

安装淘宝镜像

大家都知道国内直接使用 npm 的官方镜像是非常慢的, 这里推荐使用淘宝 NPM 镜像。

```
npm install -g cnpm --registry=https://registry.npm.taobao.org
```

这样就可以使用 cnpm 命令来安装模块了:

```
cnpm install [name]
```

安装webpack

```
cnpm install webpack -g
```

安装vue脚手架

```
npm install vue-cli -g
```

在硬盘上找一个文件夹放工程用的, 在终端中进入该目录

```
cd 目录路径
```

根据模板创建项目

```
vue init webpack-simple 工程名字<工程名字不能用中文>
或者创建 vue1.0 的项目
vue init webpack-simple#1.0 工程名字<工程名字不能用中文>
```

会有一些初始化的设置, 如下输入:

Target directory exists. Continue? (Y/n)直接回车默认(然后会下载 vu



e2.0模板，这里可能需要连代理)

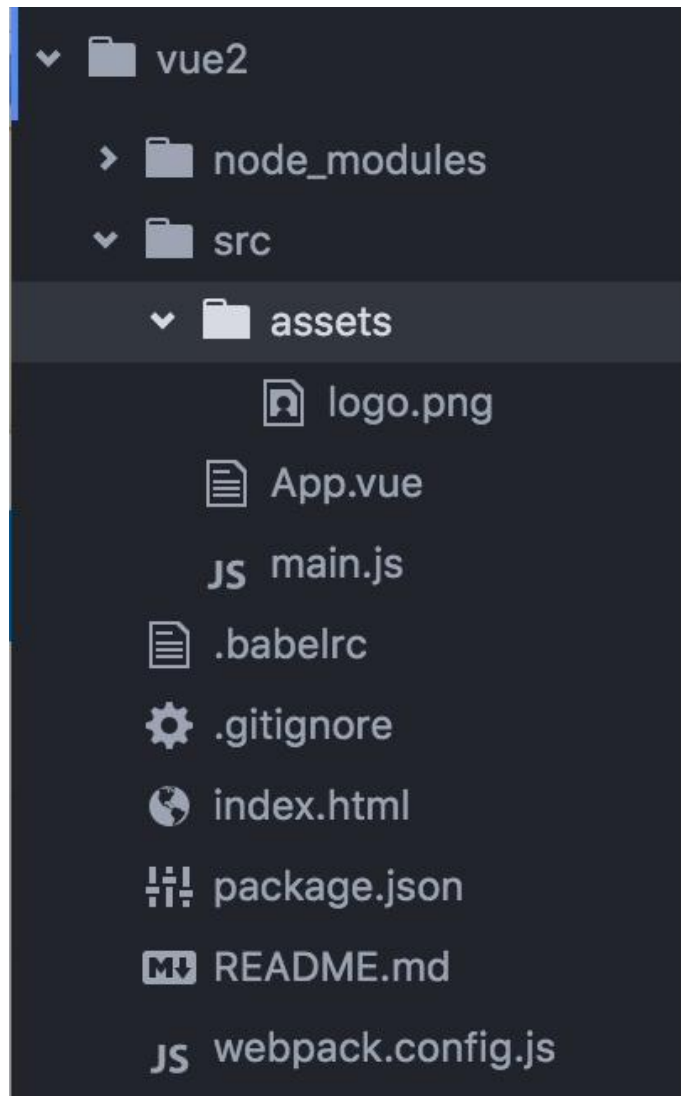
Project name (vue-test)直接回车默认

Project description (A Vue.js project) 直接回车默认

Author 写你自己的名字

cd 命令进入创建的工程目录

工程目录如图所示:



安装项目依赖

一定要从官方仓库安装，npm 服务器在国外所以这一步安装速度会很慢。

```
npm install
```

不要从国内镜像cnpm安装(会导致后面缺了很多依赖库)

```
cnpm install
```



安装 vue 路由模块vue-router和网络请求模块vue-resource

```
cnpm install vue-router vue-resource --save
```

启动项目

```
npm run dev
```

填坑(以下坑可能由于 vue2.0 导致其他相关编译打包工具没更新导致的)

【重点】 后来发现这些坑是由于 npm 不是最新的版本3.10.2, 用 npm 3.9.5就会出现以下坑

解决办法: 请运行以下命令

npm update -g

报错

```
Error: Cannot find module 'opn'  
Error: Cannot find module 'webpack-dev-middleware'  
Error: Cannot find module 'express'  
Error: Cannot find module 'compression'  
Error: Cannot find module 'sockjs'  
Error: Cannot find module 'spdy'  
Error: Cannot find module 'http-proxy-middleware'  
Error: Cannot find module 'serve-index'
```

如果你用的是老版本的 vue-cli 还可能报其他错误, 需要更新一下 vue-cli

```
npm update vue-cli
```

然后可以查看一下当前全局 vue-cli 的版本

```
npm view vue-cli
```

安装一下这个依赖到工程开发环境

```
cnpm install opn --save-dev  
cnpm install webpack-dev-middleware --save-dev  
cnpm install express --save-dev  
cnpm install compression --save-dev  
cnpm install sockjs --save-dev  
cnpm install spdy --save-dev  
cnpm install http-proxy-middleware --save-dev  
cnpm install serve-index --save-dev  
cnpm install connect-history-api-fallback --save-dev
```

再启动项目, 报错



```
ERROR in ./src/main.js
Module build failed: Error: Cannot find module 'babel-runtime/helper
s/typeof'
at Function.Module._resolveFilename (module.js:440:15)
at Function.Module._load (module.js:388:25)
at Module.require (module.js:468:17)
at require (internal/module.js:20:19)
at Object.<anonymous> (/Volumes/MacStorage/Coding/Web/vue-test/node_m
odules/.6.17.0@babel-core/lib/transformation/file/index.js:6:16)
at Module._compile (module.js:541:32)
at Object.Module._extensions..js (module.js:550:10)
at Module.load (module.js:458:32)
at tryModuleLoad (module.js:417:12)
at Function.Module._load (module.js:409:3)
@ multi main
ERROR in ./~/./2.1.0-beta.8@webpack-dev-server/client/socket.js
Module not found: Error: Can't resolve 'sockjs-client' in '/Volumes/M
acStorage/Coding/Web/vue-test/node_modules/.2.1.0-beta.8@webpack-dev-
server/client'
@ ./~/./2.1.0-beta.8@webpack-dev-server/client/socket.js 1:13-37
@ ./~/./2.1.0-beta.8@webpack-dev-server/client?http://localhost:8080
```

安装一下 babel-runtime

```
cnpm install babel-helpers --save-dev
```

启动项目，再次报错

```
Module build failed: Error: Cannot find module 'babel-helpers'
Module build failed: Error: Cannot find module 'babel-traverse'
Module build failed: Error: Cannot find module 'json5'
Module build failed: Error: Cannot find module 'babel-generator'
Module build failed: Error: Cannot find module 'detect-indent'
Module build failed: Error: Cannot find module 'jsesc'
```

找不到依赖那就再安装一下

```
cnpm install babel-helpers --save-dev
cnpm install babel-traverse --save-dev
cnpm install json5 --save-dev
...不写了，请把全部把旧的环境全部清除，更新到最新版本
```

解决办法概述

遇到

```
Module build failed: Error: Cannot find module '模块名'
```

那就安装

```
cnpm install 模块名 --save-dev(关于环境的，表现为npm run dev 启动不了)
cnpm install 模块名 --save(关于项目的，比如main.js，表现为npm run dev 成
```



功之后控制台报错)

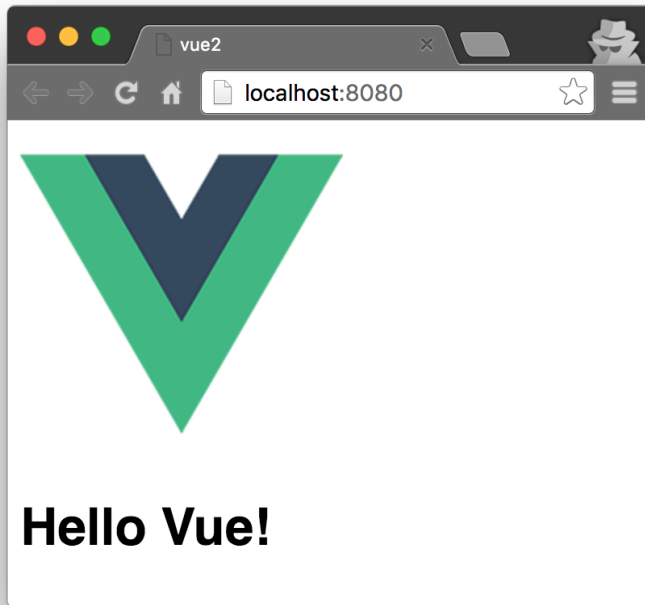
比如`escape-string-regexp`、`strip-ansi`、`has-ansi`、`is-finite`、`emojis-list`

终于可以启动项目了

输入完命令会自动启动浏览器，如果默认打开 IE 不行

```
npm run dev
```

自动启动浏览器就会看到这个界面了。

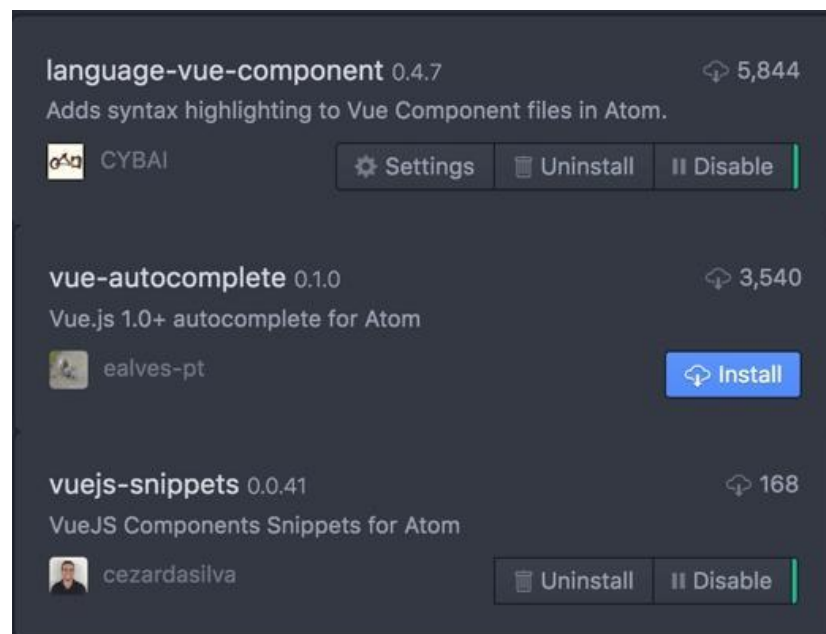


开始 Vue 之旅

打开 IDE

推荐 Atom 打开项目，需要安装 Vue 语法高亮的插件





使用官网文档学习基础

我们来看官网的一个例子，(中文文档请自行上网搜索)

Note in the method we simply update the state of our app without touching the DOM - all DOM manipulations are handled by Vue, and the code you write is focused on the underlying logic.

Vue also provides the `v-model` directive that makes two-way binding between form input and app state a breeze:

```
<div id="app-6">
  <p>{{ message }}</p>
  <input v-model="message">
</div>
```

```
var app6 = new Vue({
  el: '#app-6',
  data: {
    message: 'Hello Vue!'
  }
})
```

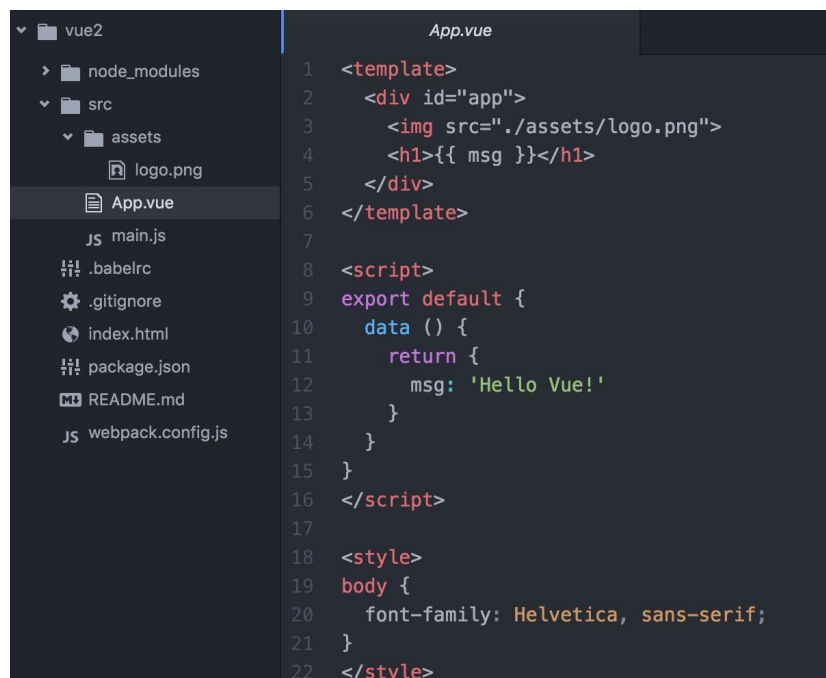
Hello Vue!

Hello Vue!

打开 工程目录下的 App.vue

template 写 html, script写 js, style写样式





为了方便叙述，我们把官网例子写在同一个组件内

这里有两个坑：

第一。一个组件下只能有一个并列的 div，可以这么写，所以复制官网示例的时候只要复制 div 里面的内容就好。



但是不能这样写：❌



```
<template>
  <div id="app">
    something here
  </div>
  <div id="app-in">
    something here
  </div>
</template>
```

第二。数据要写在 return 里面而不是像文档这样子写

```
<script>
export default {
  data () {
    return {
      msg: 'Hello Vue!'
    }
  }
}
</script>
```

错误的写法:

```
data: {
  message: 'Hello Vue!'
}
```

这样子可以自己啃完官网文档组件之前的部分了。



中文

起步
概述
Vue 实例
数据绑定语法
计算属性
Class 与 Style 绑定
条件渲染
列表渲染
方法与事件处理器
表单控件绑定
过渡

英文

Introduction
The Vue Instance
Template Syntax
Computed Properties and Watchers
Class and Style Bindings
Conditional Rendering
List Rendering
Event Handling
Form Input Bindings

来玩玩组件

前面讲得基本上都是各种常用组件的数据绑定，下面还得说说的是 Vue 的组件的使用。

在工程目录/src下创建component文件夹，并在component文件夹下创建一个 firstcomponent.vue并写仿照 App.vue 的格式和前面学到的知识写一个组件。

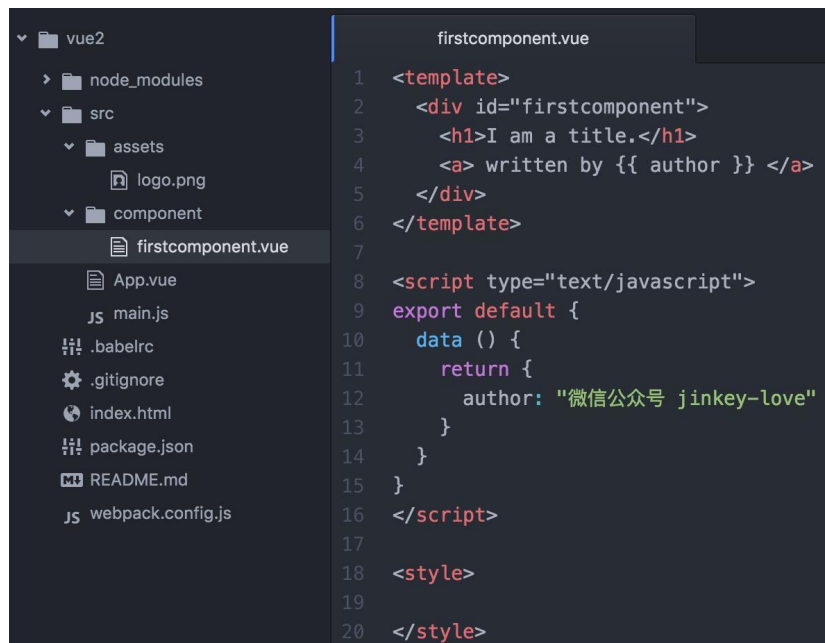
```
<template>
  <div id="firstcomponent">
    <h1>I am a title.</h1>
    <a> written by {{ author }} </a>
  </div>
</template>

<script type="text/javascript">
export default {
  data () {
    return {
      author: "微信公众号 jinkey-love"
    }
  }
}
</script>

<style>
</style>
```

duang... 不能按官网文档那样子叫我做就做，我得先试试再告诉你，我做完效果是这样子的，希望观众做完也是这样子。(迷之微笑)





然后在 App.vue 使用组件 (因为在 index.html 里面定义了 <div id="app"></div> 所以就以这个组件作为主入口, 方便)

第一步, 引入。 在 <script></script> 标签内的第一行写

```
import firstcomponent from './component/firstcomponent.vue'
```

第二步, 注册。 在 <script></script> 标签内的 data 代码块后面加上 components: { firstcomponent }。记得中间加英文逗号!!!

```
export default {
  data () {
    return {
      msg: 'Hello Vue!'
    }
  },
  components: { firstcomponent }
}
```

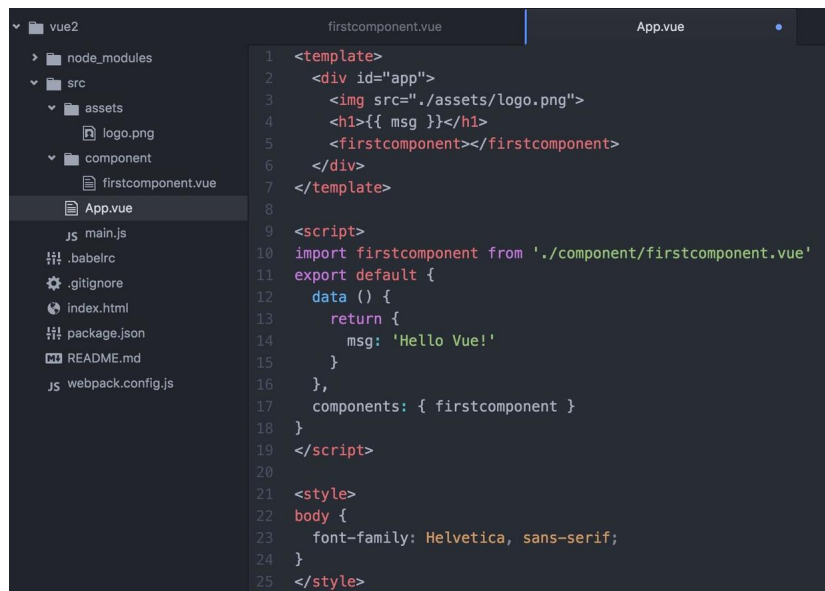
第三步, 使用。

在 <template></template> 内加上 <firstcomponent></firstcomponent>

```
<template>
  <div id="app">
    
    <h1>{{ msg }}</h1>
    <firstcomponent></firstcomponent>
  </div>
</template>
```

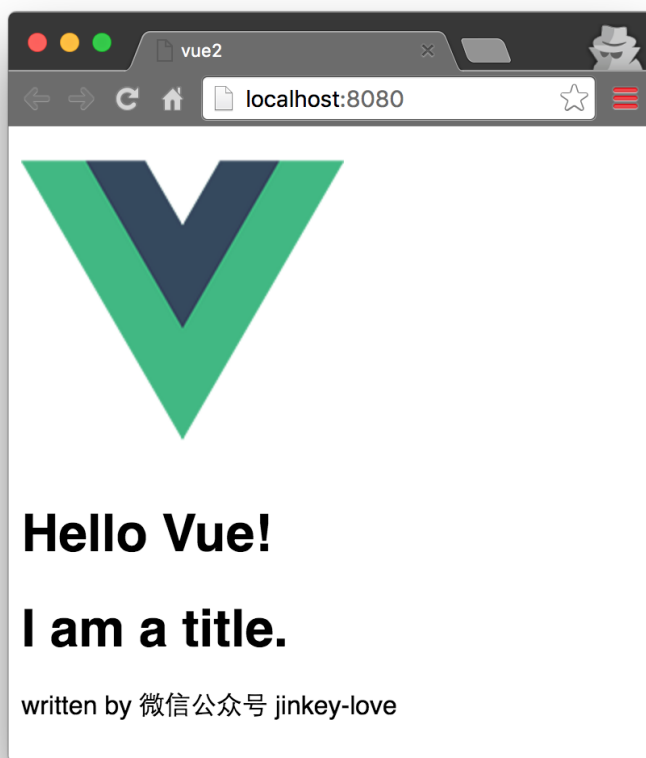
完成后的代码:





```
1 <template>
2   <div id="app">
3     
4     <h1>{{ msg }}</h1>
5     <firstcomponent></firstcomponent>
6   </div>
7 </template>
8
9 <script>
10  import firstcomponent from './component/firstcomponent.vue'
11  export default {
12    data () {
13      return {
14        msg: 'Hello Vue!'
15      }
16    },
17    components: { firstcomponent }
18  }
19 </script>
20
21 <style>
22  body {
23    font-family: Helvetica, sans-serif;
24  }
25 </style>
```

这时候看看浏览器上的 <http://localhost:8080/> 页面(之前打开过就会自动刷新), 如果你没看到效果是因为你没有对 `App.vue` 和 `firstcomponent.vue` 进行保存操作, 保存后页面会自动刷新。



使用路由搭建单页应用

之前已经通过命令安装了vue-router

```
cnpm install vue-router --save
```



在webpack.config.js加入别名

```
resolve: {  
  alias: {vue: 'vue/dist/vue.js'}  
}
```

为什么要加 alias 配置项？其作用可以在文档中有相应的描述：

独立构建 vs 运行时构建

有两种构建方式，独立构建和运行构建。

- 独立构建包括编译和支持 `template` 选项。它也依赖于浏览器的接口的存在，所以你不能使用它来为服务器端渲染。
- 运行时构建不包括模板编译，不支持 `template` 选项。运行时构建，可以用 `render` 选项，但它只在单文件组件中起作用，因为单文件组件的模板是在构建时预编译到 `render` 函数中，运行时构建只有独立构建大小的30%，只有 16Kb min+gzip大小。

默认 NPM 包导出的是 运行时 构建。为了使用独立构建，在 webpack 配置中添加下面的别名：

```
resolve: {  
  alias: {  
    vue: 'vue/dist/vue.js'  
  }  
}
```

对于Browserify，可以用 `aliasify`

❗ 不要用 `import Vue from 'vue/dist/vue.js'` - 用一些工具或第三方库引入 Vue，这可能会导致应用程序在同一时间加载运行时和独立构建并造成错误。

修改完之后的webpack.config.js是这样子的：

```
var path = require('path')  
var webpack = require('webpack')  
  
module.exports = {  
  entry: './src/main.js',  
  output: {  
    path: path.resolve(__dirname, './dist'),  
    publicPath: '/dist/',  
    filename: 'build.js'  
  },  
  resolveLoader: {  
    root: path.join(__dirname, 'node_modules'),  
  },  
  module: {  
    loaders: [  
      {  
        test: /\\\\\\\\\\\\\\\\\\\\.vue$/,  
        loader: 'vue'  
      },  
      {  
        test: /\\\\\\\\\\\\\\\\\\\\.js$/,  
        loader: 'babel',  
        exclude: /node_modules/  
      },  
      {  
        test: /\\\\\\\\\\\\\\\\\\\\.(png|jpg|gif|svg)$/,  
        loader: 'file',  
        query: {  
          name: '[name].[ext]?[hash]'  
        }  
      }  
    ]  
  }  
}
```



```

    }
  ]
},
resolve: {
  alias: {vue: 'vue/dist/vue.js'}
},
devServer: {
  historyApiFallback: true,
  noInfo: true
},
devtool: '#eval-source-map'
}

if (process.env.NODE_ENV === 'production') {
  module.exports.devtool = '#source-map'
  // http://vue-loader.vuejs.org/en/workflow/production.html
  module.exports.plugins = (module.exports.plugins || []).concat([
    new webpack.DefinePlugin({
      'process.env': {
        NODE_ENV: '"production"'
      }
    }),
    new webpack.optimize.UglifyJsPlugin({
      compress: {
        warnings: false
      }
    })
  ])
}
}

```

再按之前的方法写一个组件 secondcomponent.vue

```

<template>
  <div id="secondcomponent">
    <h1>I am another page</h1>
    <a> written by {{ author }} </a>
    <p> 感谢 <a href="https://github.com/showonne">showonne
  </a>大神的技术指导</p>
  </div>
</template>

<script>
export default {
  data() {
    return {
      author: "微信公众号 jinkey-love",
      articles: [],
    }
  }
}
</script>

<style>
</style>

```



这时候修改 main.js，引入并注册 vue-router

```
import VueRouter from "vue-router";
Vue.use(VueRouter);
```

并且配置路由规则和 app 启动配置项加上 router，旧版的 router.map 方法在 vue-router 2.0 已经不能用了。修改后的main.js如下：

```
import Vue from 'vue'
import App from './App.vue'
import VueRouter from "vue-router";
import VueResource from 'vue-resource'

//开启debug模式
Vue.config.debug = true;

Vue.use(VueRouter);
Vue.use(VueResource);

// 定义组件，也可以像教程之前教的方法从别的文件引入
const First = { template: '<div><h2>我是第 1 个子页面</h2></div>' }
import secondcomponent from './component/secondcomponent.vue'

// 创建一个路由器实例
// 并且配置路由规则
const router = new VueRouter({
  mode: 'history',
  base: __dirname,
  routes: [
    {
      path: '/first',
      component: First
    },
    {
      path: '/second',
      component: secondcomponent
    }
  ]
})

// 现在我们可以启动应用了！
// 路由器会创建一个 App 实例，并且挂载到选择符 #app 匹配的元素上。
const app = new Vue({
  router: router,
  render: h => h(App)
}).$mount('#app')
```

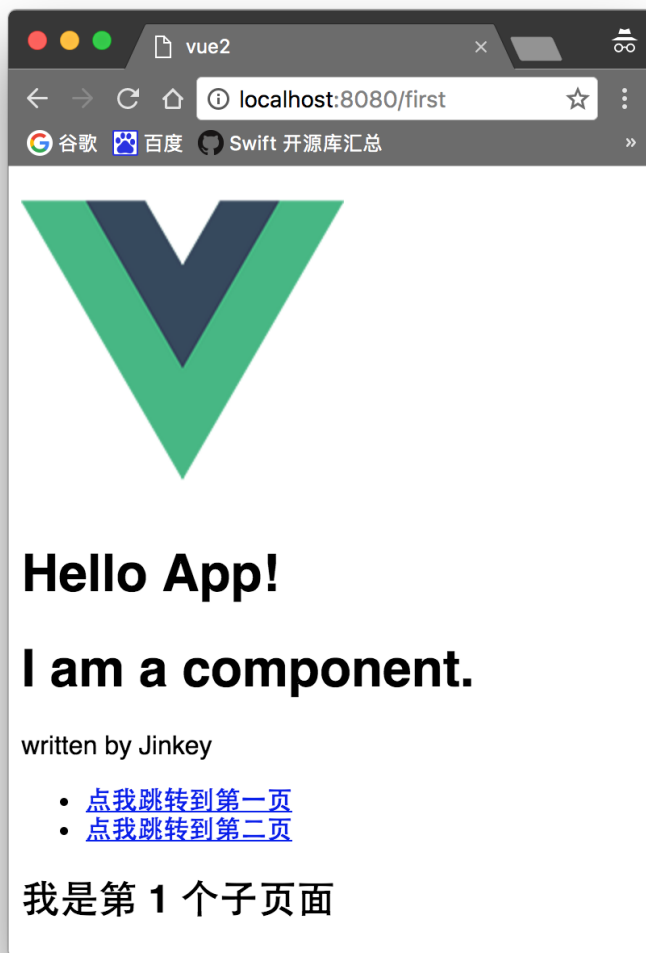
这样子改完再打开浏览器看看。





点击那两个链接试试，会发现`<router-view class="view"></router-view>`的内容已经展示出来，同时注意浏览器地址已经变更。





另外，也可以把 App.vue 的内容写在 main.js 也是可以的不过不建议这么做

```
const app = new Vue({
  router: router,
  // render: h => h(App),
  template: `
    <div>
      
      <h1>Hello App!</h1>
      <secondcomponent></secondcomponent>
      <firstcomponent></firstcomponent>
      <ul>
        <li><router-link to="/first">点我跳转到第一页</router-link></li>
        <li><router-link to="/second">点我跳转到第二页</router-link></li>
      </ul>
      <router-view class="view"></router-view>
    </div>
  `,
  components: { firstcomponent, secondcomponent }
}).$mount('#app')
```

如果你使用 vue1.0和0.7版本的 vue-router，请参照下面这个教程，他整个系列都不错的，当然仅限于 vue1.0：

<http://guowenfz.github.io/2016/03/28/vue-webpack-06-router/>



给页面加点动态数据

这时候的页面都是静态的(数据在写程序的时候已经固定了不能修改)，而每个应用基本上都会请求外部数据以动态改变页面内容。对应有一个库叫 `vue-resource` 帮我们解决这个问题。

使用命令行安装

```
cnpm install vue-resource --save
```

在 `main.js` 引入并注册 `vue-resource`:

```
import VueResource from 'vue-resource'
Vue.use(VueResource);
```

我们在 `secondcomponent.vue` 上来动态加载数据

添加一个列表:

```
<ul>
  <li v-for="article in articles">
    {{article.title}}
  </li>
</ul>
```

在 `data` 里面加入数组 `articles` 并赋值为`[]`

然后在 `data` 后面加入加入钩子函数 `mounted`(详细请参照官方文档关于 `vue` 生命周期的解析)，**`data` 和 `mount` 中间记得记得加逗号**

```
mounted: function() {
  this.$http.jsonp('https://api.douban.com/v2/movie/top25
0?count=10', {}, {
    headers: {

    },
    emulateJSON: true
  }).then(function(response) {
    // 这里是处理正确的回调

    this.articles = response.data.subjects
    // this.articles = response.data["subjects"] 也可以

  }, function(response) {
    // 这里是处理错误的回调
    console.log(response)
  });
}
```

这里使用的是豆瓣的公开 `GET` 接口，如果接口是跨域的 `POST` 请求，则需要在服务器端配置:

```
Access-Control-Allow-Origin: *
```



这时候运行看看。等一会接口返回数据，咦，数据加载出来了，棒棒哒！



更多 vue-router 的使用方法可以看



vue-router 2.0

<http://router.vuejs.org/zh-cn/index.html>

来拯救如此难看的界面

组件、双向绑定、路由、数据请求等基本特性都能用了，写到这里一个单页应用基本上成型了。但是，这界面也太 TM 难看了吧。自己写 UI 框架太费劲？那就上网找一个吧。

本来想给大家介绍 Vux 的，因为他用的是微信的 WeUI 设计规范，对于开发微信小程序或者微信内的网页非常和谐，但由于写这篇文章的时候 Vux 还不支持 vue2.0，只能用别的框架了。

命令行安装 ElementUI (此处某公司的人应该发红包了...)

```
cnpm install element-ui@next -S
```

然后在 main.js 引入并注册

```
import Element from 'element-ui'
import 'element-ui/lib/theme-default/index.css'
Vue.use(Element)
```

保存，这时候程序报错

```
Uncaught Error: Module parse failed: /Users/**/Desktop/vue2/node_modules/.1.0.0-rc.5@element-ui/lib/theme-default/index.css Unexpected character '@' (1:0)
You may need an appropriate loader to handle this file type.
```

官网文档又有坑了，安装教程也不跟我们说这一步，当我们都是高手了...

报错是由于我们引入了 index.css 这个 CSS 文件，但是 webpack 打包的时候无法识别并转换成 js，所以需要配置才能读取 css 和字体文件，运行命令安装下面三个东西(如果之前安装过就不需要了)

```
cnpm install style-loader --save-dev
cnpm install css-loader --save-dev
cnpm install file-loader --save-dev
```

在 webpack.config.js 中的 loaders 数组加入以下配置，记得该加逗号的地方加逗号!

```
{
  test: /\\\\\\\\\\\\\\\\.css$/,
  loader: "style!css"
},
{
  test: /\\\\\\\\\\\\\\\\\\\\.(eot|woff|woff2|ttf)([\\\\\\\\\\\\\\\\\\\\?]?\\.*)$/,
  loader: "file"
}
```



修改完的 webpack.config.js 如下

```
var path = require('path')
var webpack = require('webpack')

module.exports = {
  entry: './src/main.js',
  output: {
    path: path.resolve(__dirname, './dist'),
    publicPath: '/dist/',
    filename: 'build.js'
  },
  resolveLoader: {
    root: path.join(__dirname, 'node_modules'),
  },
  module: {
    loaders: [
      {
        test: /\.vue$/,
        loader: 'vue'
      },
      {
        test: /\.js$/,
        loader: 'babel',
        exclude: /node_modules/
      },
      {
        test: /\.css$/,
        loader: "style!css"
      },
      {
        test: /\.?(eot|woff|woff2|ttf)([\.\?]*$|
*)$/i,
        loader: "file"
      },
      {
        test: /\.(png|jpg|gif|svg)$/i,
        loader: 'file',
        query: {
          name: '[name].[ext]?[hash]'
        }
      }
    ]
  },
  resolve: {
    alias: {vue: 'vue/dist/vue.js'}
  },
  devServer: {
    historyApiFallback: true,
    noInfo: true
  },
  devtool: '#eval-source-map'
}

if (process.env.NODE_ENV === 'production') {
  module.exports.devtool = '#source-map'
  // http://vue-loader.vuejs.org/en/workflow/production.html
}
```

1



```
module.exports.plugins = (module.exports.plugins || []).concat([
  new webpack.DefinePlugin({
    'process.env': {
      NODE_ENV: '"production"'
    }
  }),
  new webpack.optimize.UglifyJsPlugin({
    compress: {
      warnings: false
    }
  })
])
}
```

给豆瓣的电影列表套个衣服(样式):

```
<el-card class="box-card">
  <div slot="header" class="clearfix">
    <h2 style="line-height: 36px; color: #20A0FF">豆瓣电影排行榜</h2>
  </div>
  <div v-for="article in articles" class="text item">
    {{article.title}}
  </div>
</el-card>
```

打开浏览器,输入网址:<http://localhost:8080/second>





列表比之前漂亮多了，你还可以参照 ElementUI 的文档使用更多组件样式

<http://element.eleme.io/#/component/layout>



h3>编译

```
npm run build
```

又报错了...orz

```
ERROR in build.js from UglifyJs
SyntaxError: Unexpected token punc «(», expected punc «:» [build.js:3
2001,6]
把node_modules/.bin/cross-env里的
require('..../dist')(process.argv.slice(2));
```

后来发现直接运行 webpack 命令就可以打包了

```
webpack --color --progress
```

接着把 index.html 和整个 dist 目录丢到服务器就可以了。

 点我分享笔记

视频通话SDK

声网Agora.io，API接口，4行代码接入。每月1万分钟免费。 www.agora.io

打开

|

