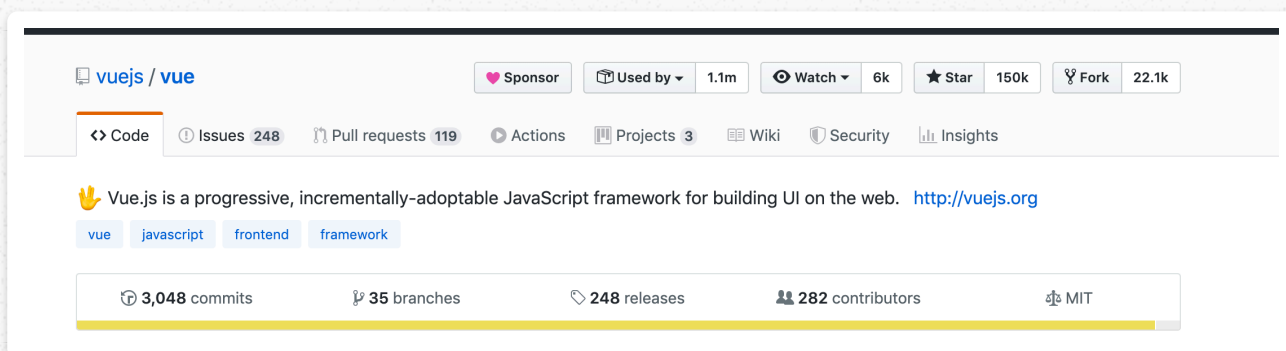


Vue培训文档

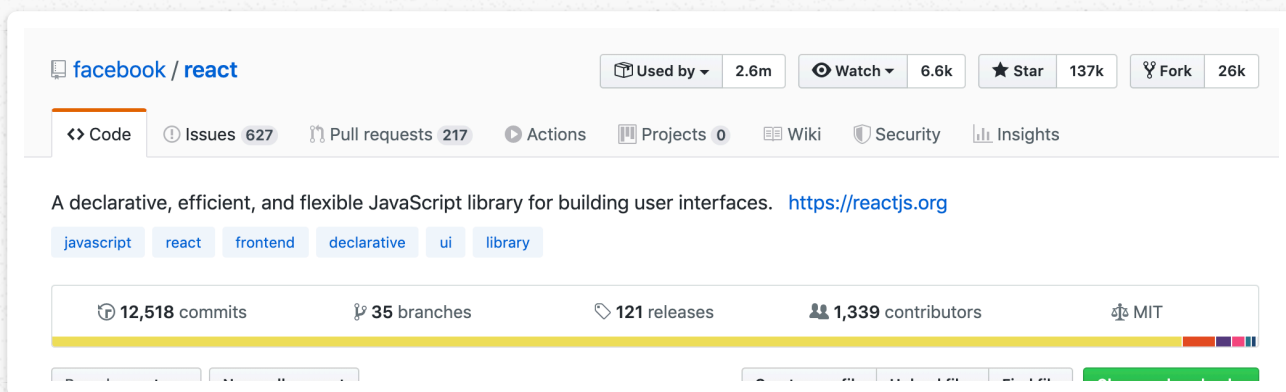
官方文档: `Vue.js` [渐进式JavaScript 框架](#)

近两年，在前端岗位的招聘需求中，对Vue开发工程师的需求越来越大。因为市场环境和需求的变化，前端技术逐渐从远古生物jQuery到现在的MVVM框架转变。之前传统的PC端开发模式，比如我们一个页面跳转，我们需要把资源重新加载一遍，需要不断的操作dom元素，而操作dom元素又是一个性能开销很大的操作。现在的MVVM框架，我们开发者逐渐摆脱了频繁操作dom元素，通过更改数据层，视图层也会相应变更，这种响应方式不仅更有助于开发，同时对性能也有较好的提升。以下4张图是我在github上截图的star数，通过对比，我们可以看到，在MVVM框架的3驾马车---Vue React Angular中，Vue独占鳌头。

- Vue



- React



- Angular

angular / angular

Watch 3.2k

Star 52.7k

Fork 14.6k

Code

Issues 2,748

Pull requests 462

Actions

Projects 5

Security

Insights

One framework. Mobile & desktop. <https://angular.io>

angular

typescript

web

javascript

pwa

web-framework

web-performance

15,576 commits

78 branches

0 packages

365 releases

1,013 contributors

MIT

• 上古生物jQuery

jquery / jquery

Used by 360k

Watch 3.5k

Star 52.3k

Fork 18.6k

Code

Issues 69

Pull requests 13

Projects 0

Wiki

Security

Insights

jQuery JavaScript Library <https://jquery.com/>

6,426 commits

4 branches

150 releases

278 contributors

MIT

那么，问题来了，为什么是Vue呢，它凭什么这么火？

一、WHY VUE.JS ?

• Vue的描述

Vue.js（读音 /vjuː/，类似于 view）是一套构建用户界面的**渐进式**框架，不仅易于上手，还便于与第三方库或既有项目整合。**个人看来，渐进式代表的含义是它的强制性少，你不需要一下子就使用它所有的东西，如果只使用Vue最基础的声明式渲染的功能，则完全可以把Vue当做一个模板引擎来使用；如果想以组件化开发方式进行开发，则可以进一步使用Vue里面的组件系统；如果要制作SPA(单页应用)，则使用Vue里面的客户端路由功能。如果组件越来越多，需要共享一些数据，则可以使用Vue里的状态管理。**

而比如说，Angular，它的主张性比较强，如果你用它，必须接受以下东西：必须使用它的模块机制、必须使用它的依赖注入。

• 什么是MVVM?

MVVM可以拆分成：View --- ViewModel --- Model三部分，看下面的视图：



那么，我们怎么理解MVVM呢？

上图中，左侧的View相当于我们的DOM内容，我们所看到的页面视图，右侧的Model相当于我们的数据对象，比如一个对象的信息：`{name: '张三', age: 18}`而中间的监控者就负责监控两侧的数据，并相对应地通知另一侧进行修改。比如：你在Model层中修改了name的值为：“李四”，那么View视图层显示的“张三”也会自动变成了“李四”，而这个过程就是有ViewModel来操作的，你不用再手动操作DOM了。

```
// 传统做法
<div id="app">张三</div>
<script>
  var appEl = document.getElementById('app');
  appEl.innerText = '李四';
</script>
```

```
// Vue做法
<div id="app">
  {{ message }}
</div>
<script>
  var app = new Vue({
    el: '#app',
    data: {
      message: '张三'
    },
    created() {
      this.changeMessage();
    },
    methods: {
      /**
       * @description 修改message的值
       */
      changeMessage() {
        this.message = '李四';
      }
    }
  });
```

```
    }  
  }  
})  
</script>
```

- 为什么是Vue而不是React?

Vue.js更轻量更快，文档写得很好，界面清爽，内容翔实。对于新手来说，比较好上手

生态完善，插件丰富

先入为主，个人喜好，不习惯JSX语法

```
<ul>  
  {items.map(item =>  
    <li key={item.id}>{item.name}</li>  
  )}  
</ul>
```

二、Vue常用知识点回顾

- 模版内容

Vue.js使用了基于HTML的模板语法，允许声明式地将DOM绑定至底层Vue实例的数据。接下来我们看下常用的模版内容语法。

1.文本插值

文本渲染最常见的形式是使用双大括号语法来进行文本插值，下面的message相当于一个变量或占位符，最终会表示为真正的文本内容。详见vue-demo案例

```
<div id="app">  
  {{ message }}  
</div>  
<script>  
new Vue({  
  el: '#app',  
  data:{  
    message: 测试内容 '  
  }  
})  
</script>
```


2.表达式插值

```
{{ number + 1 }}  
{{ ok ? 'YES' : 'NO' }}  
{{ message.split('').reverse().join('') }}
```

上面这些表达式会在所属Vue实例的数据作用域下作为JS被解析。有个限制就是，每个绑定都只能包含单个表达式，所以下面的例子都不会生效

```
<!-- 这是语句，不是表达式 -->  
{{ var a = 1 }}  
<!-- 流控制也不会生效，请使用三元表达式 -->  
{{ if (ok) { return message } }}
```

3.v-text

实现插值类似效果的另一种写法是使用v-text指令，该指令用于更新元素的innerText。详见vue-demo案例

4.v-html

如果要输出真正的HTML，需要使用v-html指令，该指令用于更新元素的innerHTML。详见vue-demo案例

5.class绑定

数据绑定一个常见需求是操作元素的class列表和它的内联样式。绑定class包括对象语法、数组语法和组件绑定，详见vue-demo案例。

对象语法：可以传给v-bind:class一个对象，以动态地切换class。

数组语法：可以把一个数组传给v-bind:class，以应用一个class列表

6.style绑定

style绑定也是一个非常常见的操作，它也是有对象语法和数组语法，详见vue-demo案例。

对象语法：对象语法十分直观——看着非常像CSS，其实它是一个JS对象。CSS属性名可以用驼峰式(camelCase)或(配合引号的)短横分隔命名(kebab-case)。

数组语法：v-bind:style的数组语法可以将多个样式对象应用到一个元素上。

7.过滤器

Vue.js允许自定义过滤器，可被用作一些常见的文本格式化。过滤器可以用在两个地

方：模板插值和v-bind表达式。过滤器应该被添加在JS表达式的尾部，由“管道”符指示。

```
{{ message | capitalize }}  
<div v-bind:id="rawId | formatId"></div>
```

具体详见demo案例。

过滤器有两种注册形式，一种是全局注册、一种是局部注册

全局注册：使用Vue.filter()方法注册

```
// 注册  
Vue.filter('my-filter', function (value) {  
  // 返回处理后的值  
})
```

局部注册：在组件中使用filters参数

```
filters: {  
  myFilter (value) {  
    return value.split('').reverse().join('')  
  },  
  filterLength (value) {  
    return value.length  
  },  
  filterFunc (value, arg1, arg2) {  
    return value + '-' + arg1 + '-' + arg2  
  }  
}
```

• 指令

指令有2种，一种是系统默认设置的核心指令，另外一种是自定义的指令。

核心指令：

1、v-if

2、v-show

两者的区别：v-if是根据表达式来生成或移除一个dom元素

v-show是根据表达式的值来显示或者隐藏元素，根据的是display的值，该元素是一直渲染的。v-if有更高的切换消耗，而v-show有更高的初始渲染消耗。如果频繁切换，则

用v-show

3、v-bind

```
<a v-bind:href="url">...</a>
// 通常在开发中会省略v-bind, 如
<a :href="url">...</a>
<p :class="classObject"></p>
```

4、v-on

它用于监听 DOM 事件

```
<a v-on:click="doSomething">...</a>
// 通常在开发中, 用@代替v-on:, 如
<a @click="doSomething">...</a>
```

5、v-for

我们可以用 v-for 指令基于一个数组来渲染一个列表。v-for 指令需要使用 item in items 形式的特殊语法, 其中 items 是源数据数组, 而 item 则是被迭代的数组元素的别名。详见demo

注意: 由于 JavaScript 的限制, Vue 不能检测以下数组的变动: 当你利用索引直接设置一个数组项时, 1) 例如: vm.items[indexOfItem] = newValue。2) 当你修改数组的长度时, 例如: vm.items.length = newLength

自定义指令:

在 Vue2.0 中, 代码复用和抽象的主要形式是组件。然而, 有的情况下, 你仍然需要对普通 DOM 元素进行底层操作, 这时候就会用到自定义指令。

自定义指令和过滤器类似, 也分全局和局部注册, 全局注册的例子如下:

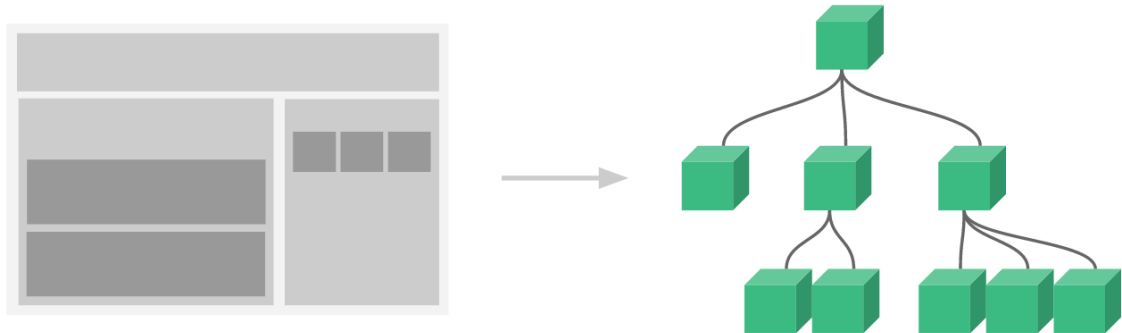
```
Vue.directive('focus', {
  // 当被绑定的元素插入到 DOM 中时.....
  inserted: function (el, binding) {
    el.focus()    // 聚焦元素
  }
})
```

局部注册是在组件中, 对directives对象增加指令属性。

详情见demo

三、Vue组件化、组件通信、代码复用

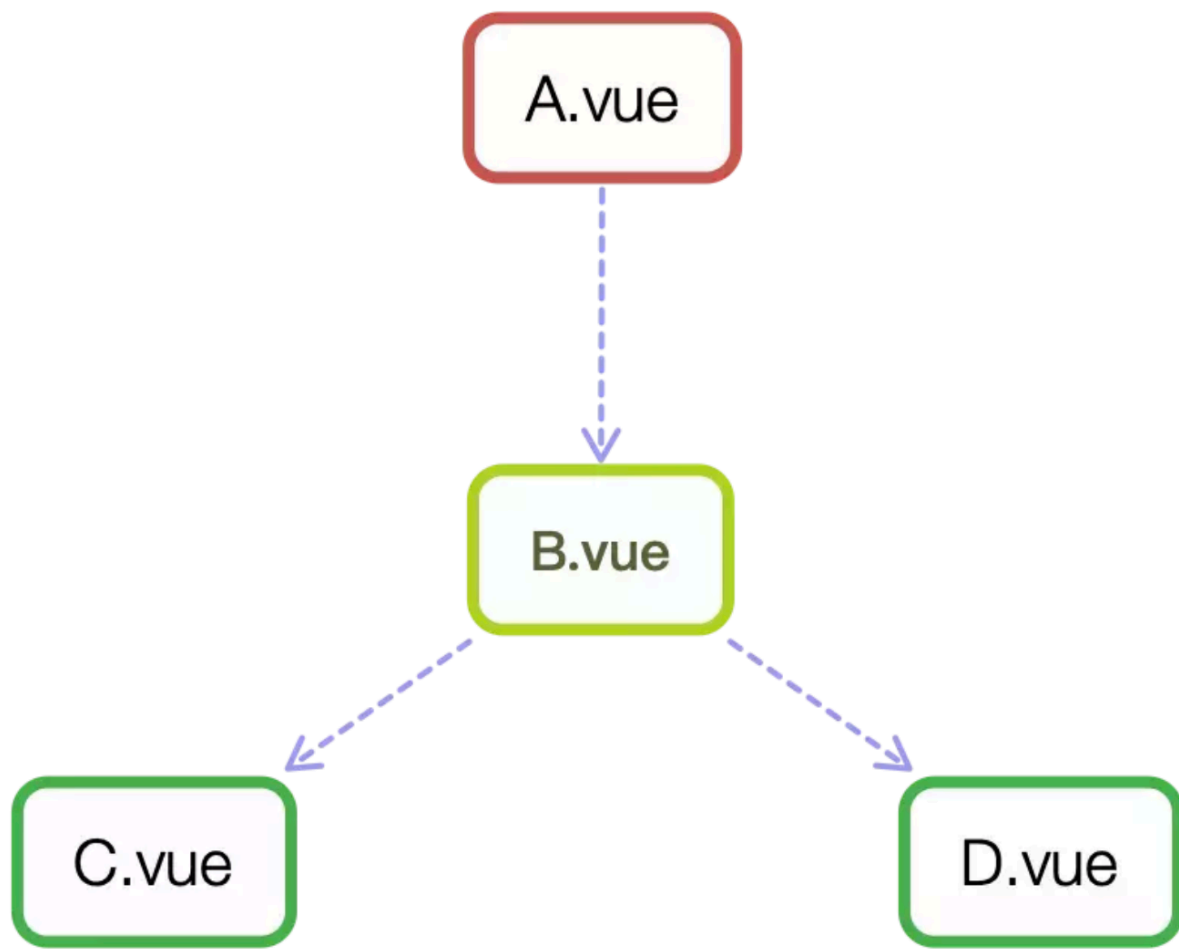
组件(Component)是Vue最强大的功能之一，Vue 主要思想之一就是组件式开发。组件可以扩展HTML元素，封装可重用的代码。根据项目需求，抽象出一些组件，每个组件里包含了展现、功能和样式。每个页面，根据自己所需，使用不同的组件来拼接页面。这种开发模式使前端页面易于扩展，且灵活性高，而且组件之间也实现了解耦。



犹如上图，一个页面，我们可以拆分为多个组件，头部组件、左侧内容组件、右侧内容组件。左侧内容中，又包含2个组件，右侧内容包含3个组件。

组件注册：

组件关系：



上图中，展示了组件之间的关系：

A 组件和 B 组件、B 组件和 C 组件、B 组件和 D 组件形成了父子关系；

C 组件和 D 组件形成了兄弟关系；

A 组件和 C 组件、A 组件和 D 组件形成了隔代关系（其中的层级可能是多级，即隔多代）

组件作为Vue的核心，它们的作用域又是相互独立的，这也意味着不同组件之间的数据是无法相互引用，所以才有了组件之间的各种通信方式。接下来重点介绍不同组件关系间的通信方式。

- 父子组件

1、通过prop 向子组件传递数据(详见demo案例)

```
// 父组件引入子组件
<a-component title="我是a组件" @show="handleShow"></a-component>
```

```

// 子组件代码
<!-- 组件a -->
<template>
  <div class="wrapper">
    <div class="title" @click="show">{{title}}</div>
  </div>
</template>

<script>
export default {
  props: ['title'],
  data () {
    return {
      flag: false
    }
  },
  methods: {
    show () {
      this.flag = !this.flag
      this.$emit('show', this.flag)
    }
  }
}
</script>

```

注意：prop 是单向数据流的，不要在子组件里直接修改props中的对象数据

2、*emit*和*on*

父组件通过v-on:fn="xx"来设置监听，子组件通过\$emit('fn') 来触发。通常开发中，一般用@代替v-on:

```

// 父组件
<a-component title="我是a组件" :color="color" @show="handleShow"></a-component>

// 子组件
this.$emit('show', this.flag)

// 详细案例见demo

```

3、ref

在父组件中，可以为子组件添加ref引用，然后根据ref引用值获取子组件的数据和方法

```

<b-component ref="componentB"></b-component>

```

```
showComponentB () {  
  console.log(this.$refs.componentB.text)  
  this.$refs.componentB.show()  
}
```

• 非父子

1、bus方式传递数据

该方式的原理是利用一个新的vue实例作为事件的总线

```
// 新建一个bus.js文件  
import Vue from 'vue'  
const bus = new Vue()  
export default bus  
  
// 在主入口main.js中引入并挂载到Vue的原型对象上  
import bus from './utils/bus'  
Vue.prototype.$bus = bus  
  
// 组件a中传递数据  
this.$bus.$emit('emitComponentC', 'hello c')  
  
// 组件c中监听传递事件，获取数据  
mounted () {  
  this.$bus.$on('emitComponentC', (msg) => {  
    this.text = msg  
  })  
}
```

2、vuex状态管理工具

Vuex 是状态管理工具，实现了项目状态的集中式管理，它的实现借鉴了React的Redux的设计和概念。Vuex 是专门为 Vue.js 设计的状态管理库，以利用 Vue.js 的细粒度数据响应机制来进行高效的状态更新。Vuex算是Vue的终极通信工具，凡是以上能实现的，它都能做到。

接下来，通过案例来看下vuex是怎么做的，详情见demo.

• 代码复用

- 1、封装公用组件
- 2、提取公用方法
- 3、provide（依赖）和inject（注入）的使用
- 4、mixin

详情演示见demo。

四、Vue项目工程搭建和配置

推荐使用vue-cli脚手架搭建Vue项目，@vue/cli 2.x 版本的脚手架搭建方式如下：

```
vue init webpack my-project
```

生成的项目结构图：

✓ WE-OFFICE

- > build
- > config
- > dist
- > node_modules
- > src
- > static
- > test

6 .babelrc

⚙️ .editorconfig

⚙️ .eslintignore

⚙️ .eslintrc.js

📁 .gitignore

JS .postcssrc.js

<> index.html

{ } package-lock.json

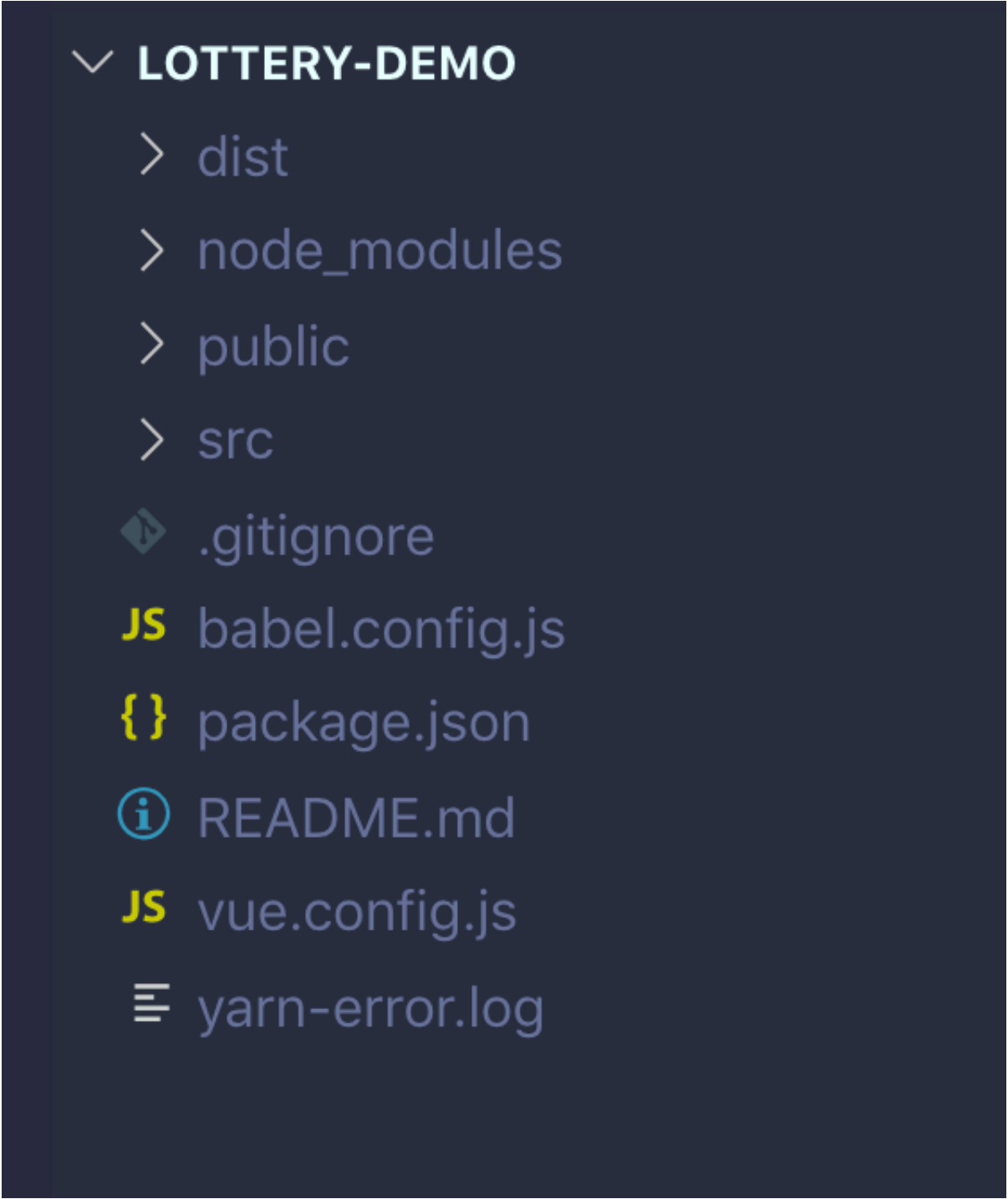
{ } package.json

📄 README.md

最新的@vue/cli 3.x 版本，官方文档：<https://cli.vuejs.org/zh>，搭建方式为：

```
vue create my-project
```

生成的项目结构图：



▼ **LOTTERY-DEMO**

- > dist
- > node_modules
- > public
- > src
- 📄 .gitignore
- JS babel.config.js
- { } package.json
- 📄 README.md
- JS vue.config.js
- ≡ yarn-error.log

本次讲解，以@vue/cli 2.x 版本的为基础

- 接口配置

方式1:

1) 本地开发环境

本地开发中，我们在config/index.js文件中，为dev模式的对象增加一个proxyTable对象，类似：

```
proxyTable: {  
  "/webchatgroup-server": {  
    target: "http://hzdsc.jiebai.com", // 拦截webchatgroup-server转发的  
地址  
    changeOrigin: true, // 是否跨域  
    /**  
     * @description webchatgroup-server重写  
     * 如http://hzdsc.jiebai.com/webchatgroup-server/get/activities  
     * 如果重写设置为空字符串""，则请求会变成http://hzdsc.jiebai.com/get/  
activities  
     */  
    pathRewrite: {  
      "^/webchatgroup-server": "/webchatgroup-server"  
    }  
  },  
},
```

通常Vue项目搭配axios使用，作为我们的请求插件，当请求地址拦截到/webchatgroup-server开头的api时，该请求会被转发到我们配置的target地址

2) 生产环境

如果前后端代码部署的域名不一致，则需要后端配置nginx转发，例如：

```
server {
    listen 80;
    server_name webchatgroup.data4truth.com;

    rewrite ^/api/webchat/wx/group/verification$ /webchatgroup-server/api/webchat/wx/group/verification;

    location / {
        root /home/front/hzds;
        try_files $uri $uri/ @router;
        index index.html index.htm;
    }

    location ~* ^/(webchatgroup-server).* {
        proxy_pass http://183.131.202.154:8006;
    }

    location ~* ^/(webchatmall-server).* {
        proxy_pass http://183.131.202.154:8006;
    }

    location ~* ^/(order-server).* {
        proxy_pass http://183.131.202.154:8006;
    }

    location ~* ^/(medical-server).* {
        proxy_pass http://183.131.202.154:8006;
    }
}
```



方式2:

对axios封装中, 为其设置baseUrl, 如:

```
import axios from 'axios';
axios.defaults.baseUrl = 'http://hzdsc.jiebai.com'
```

- 开发源代码src目录 src目录通常用于放置我们的源代码, 以下是个人常用的目录结构:


```
✓ src
  > api
  > assets
  > axios
  > components
  > env
  > pages
  > router
  > store
  > utils
  ▼ App.vue
  JS main.js
```

具体情况，我们在代码中一起看一下

五、Vue搭配UI组件推荐

六、Webpack搭建知识

七、Vue中的单元测试
