

前端开发框架介绍

作者：徐达

目录

CONTENTS

前端发展历程

前端近些年来发展的主要情况各个阶段的特点

01

前端主流框架介绍

vue、react、angular三大框架的对比，介绍为什么要用vue

02

Vue、Element UI介绍


具体介绍vue、elementui的原理及使用方法

03

前端框架开发实战

实际开发案例代码及编写，前端工具使用介绍

04



01

第一部分

前端发展阶段介绍

前端近些年来发展的主要情况各个阶段的特点

前端发展阶段

Ajax阶段

AJAX 技术的整个过程中，后端只是负责提供数据，其他事情都由前端处理。前端不再是后端的模板，而是实现了从“获取数据 --》 处理数据 --》 展示数据”的完整业务逻辑。

SPA阶段

前端可以做到读写数据、切换视图、用户交互，这意味着，网页其实是一个应用程序，而不是信息的纯展示。这种单张网页的应用程序称为 SPA。

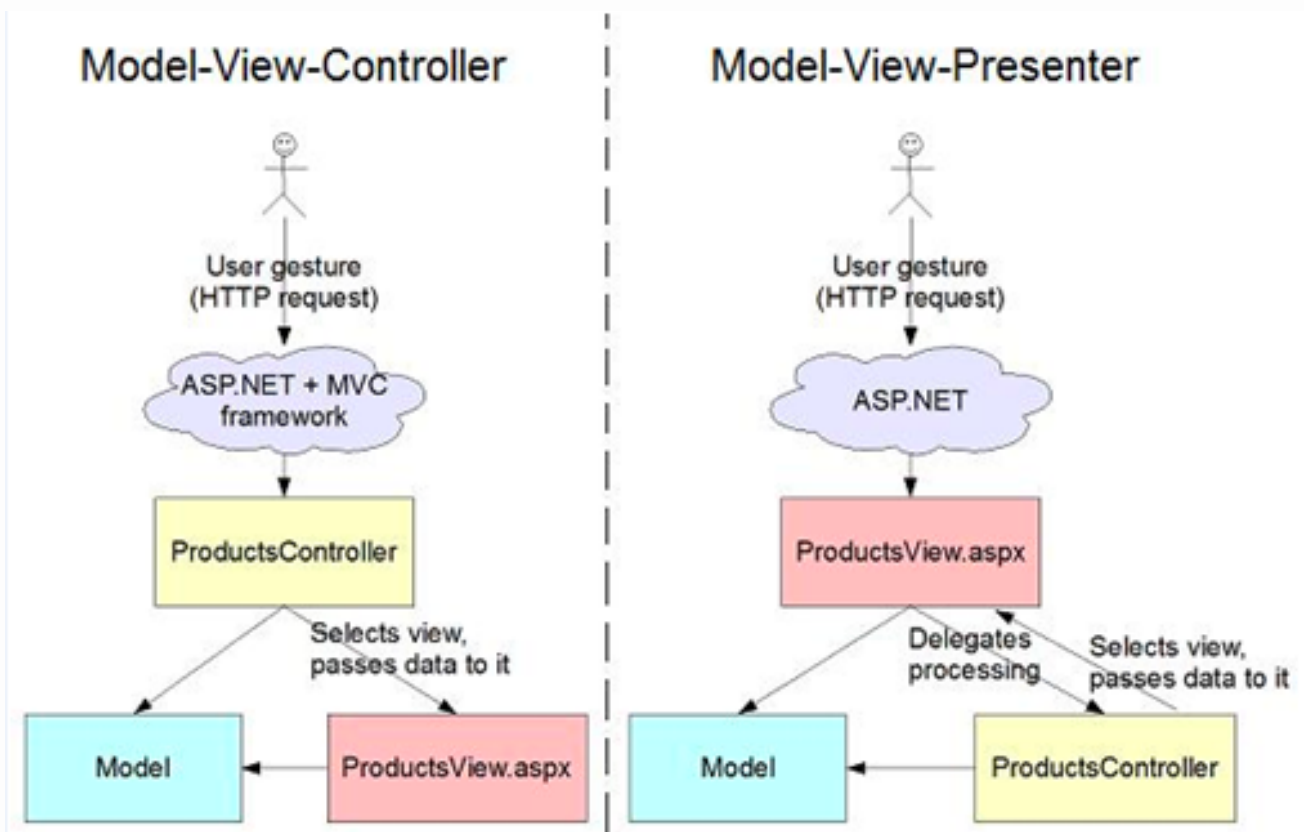
互联网发展的早期，网站的前后端开发是一体的。后端收到浏览器的请求，生成静态页面，发送到浏览器。那时的网站开发，采用的是后端 MVC 模式。

静态页面阶段

前端代码有了读写数据、处理数据、生成视图等功能，因此迫切需要辅助工具，方便开发者组织代码。这导致了前端 MVC 框架的诞

前端MVC阶段

MVX框架模式：MVC+MVP+MVVM



- **MVC**

用户的请求首先会到达Controller，由Controller从Model获取数据，选择合适的View，把处理结果呈现到View上；

- **MVP**

用户的请求首先会到达View，View传递请求到特定的Presenter，Presenter从Model获取数据后，再把处理结果通过接口传递到View。

- **MVVM**

立足于原有MVP框架并且把WPF的新特性(数据绑定DataBind、依赖属性Dependency Property、路由事件Routed Events、命令Command等...)揉合进去。

使用MVVM的好处



- **低耦合**
View可以独立于Model变化和修改，一个ViewModel可以绑定到不同的View上，当View变化的时候Model可以不变，当Model变化的时候View也可以不变。
- **可重用性**
可以把一些视图的逻辑放在ViewModel里面，让很多View重用这段视图逻辑
- **独立开发**
开发人员可以专注与业务逻辑和数据的开发(ViewModel)。设计人员可以专注于界面(View)的设计
- **可试性**
可以针对ViewModel来对界面(View)进行测试。



02

第二部分

前端主流框架介绍

vue、react、angular三大框架的对比，介绍为什么要用vue

三大框架前端框架

比对	Vue	React	Angular
开发与维护	尤雨溪	Facebook	Google
Github star	95284	96651	58492
Github 代码贡献者人数	190	1184	635
日评星数 (最近一年)	111.8	80.2	33.5

Github star 与 Github 代码贡献者数字统计于 2018-5-25。



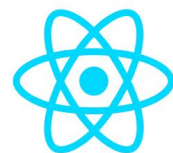
Vue

Vue 是一套用于构建用户界面的**渐进式框架**。与其它大型框架不同的是，Vue 被设计为可以自底向上逐层应用。能够为复杂的单页应用提供驱动。



Angular

Angular 是一款来自谷歌的开源的 web 前端框架，诞生于 2009 年，由 Misko Hevery 等人创建，后为 Google 所收购，被用于 Google 的多款产品当中。



React

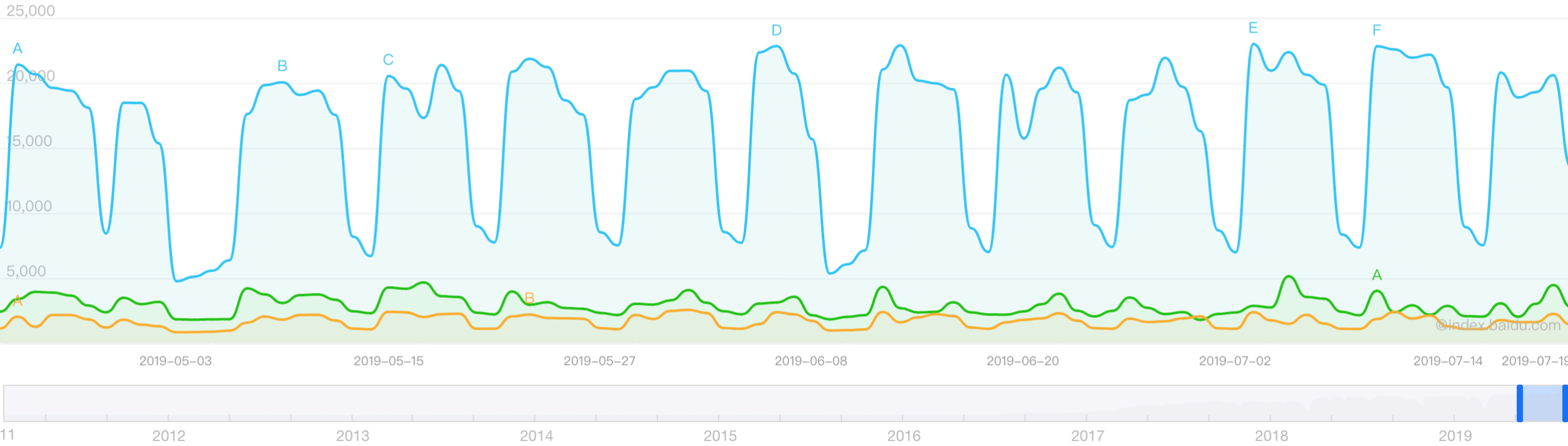
React 是一个声明式，高效且灵活的用于构建用户界面的 JavaScript 库。使用 React 可以将一些简短、独立的代码片段组合成复杂的 UI 界面，这些代码片段被称作“组件”。

vue

react

angular

☒ 新闻头条☐ 平均值



搜索指数概览?

关键词	整体日均值	移动日均值	整体同比	整体环比	移动同比	移动环比
vue	16,105	2,053	22% ↑	16% ↑	-3% ↓	4% ↑
react	2,911	922	-26% ↓	-16% ↓	3% ↑	2% ↑
angular	1,690	319	-6% ↓	5% ↑	-3% ↓	-2% ↓

为什么选择Vue



- 1、vue.js更轻量，压缩后大小只有20K+, 但React压缩后大小为44k, Angular压缩后大小有56k, 所以对于移动端来说, vue.js更适合;
- 2、vue.js更易上手, 学习曲线平稳, 而Angular入门较难, 概念较多(比如依赖注入), 很多思想沿用了后台的技术; react需学习较多东西, 附带react全家桶;
- 3、vue.js吸收两家之长, 借用了angular的指令(比如v-show,v-hide, 对应angular的ng-show,ng-hide)和react的组件化(将一个页面抽成一个组件, 组件具有完整的生命周期)
- 4、vue.js还有自己的特点, 比如计算属性



03

第三部分

Vue、Element UI介绍

具体介绍vue、elementui的原理及使用方法

Vue.js核心思想



Vue.js的组件设计原则



- 1、页面上每个独立的可视/可交互区域视为一个组件
(比如页面的头部, 尾部, 可复用的区块)
- 2、每个组件对应一个工程目录, 组件所需要的各种资源在这个目录下就近维护(组件的就近维护思想体现了前端的工程化思想, 为前端开发提供了很好的分治策略, 在vue.js中, 通过.vue文件将组件依赖的模板js样式写在一个文件中)
- 3、页面不过是组件的容器, 组件可以嵌套自由组合成完整的页面

SPA 单页应用



```
1 <template lang="jade">
2   div
3     p {{ greeting }} World!
4     other-component
5 </template>
6
7 <script>
8   import OtherComponent from './OtherComponent.vue'
9
10  export default {
11    data () {
12      return {
13        greeting: 'Hello'
14      }
15    },
16    components: {
17      OtherComponent
18    }
19  }
20 </script>
21
22 <style lang="stylus" scoped>
23   p
24     font-size 2em
25     text-align center
26 </style>
```

Line 27, Column 1 Spaces: 2 Vue Component

• 介绍

复杂的项目中：多页模式会有以下缺点

字符串模板：缺乏语法高亮，在 HTML 有多行的时候，需要用到丑陋的

不支持 CSS：意味着当 HTML 和 JavaScript 组件化时，CSS 明显被遗漏

没有构建步骤：限制只能使用 HTML 和 ES5 JavaScript，而不能使用预处理器，如 Pug (formerly Jade) 和 Babel

single-file components(单文件组件) 为以上所有问题提供了解决方法

• 怎么看待关注点分离呢

关注点分离不等于文件类型分离。在现代 UI 开发中，我们已经发现相比于把代码库分离成三个大的层次并将其相互交织起来，把它们划分为松散耦合的组件再将其组合起来更合理一些。

即便你不喜欢单文件组件，你仍然可以把 JavaScript、CSS 分离成独立的文件然后做到热重载和预编译。



1 Vuejs

Vue.js的官方文档及教程

2 Vue Router

Vue Router 是 [Vue.js](#) 官方的路由管理器。它和 Vue.js 的核心深度集成，让构建单页面应用变得易如反掌。

3 Vuex

Vuex 是一个专为 Vue.js 应用程序开发的**状态管理模式**。它采用集中式存储管理应用的所有组件的状态，并以相应的规则保证状态以一种可预测的方式发生变化。



● 选择Element UI的原因

- 有大厂背书：虽然核心开发只有两三个人，但至少不用担心哪天就不维护，带着小姨子跑路了
- 持续迭代：`element-ui` 发版至今release了四十多个版本，之前平均都是一周一个小版本更新(是不是不小心暴露了它bug多的问题/(ToT)/~~)(ps: 至2017.12.4 已经迭代了74个版本，还保持着较高更新频率)。
- 生态圈优异，社区活跃：其 contributors已经有250多人(前期我有饶有兴致的贡献过几个pr，参与过七八十个issue)，社区里也有很多基于 `element-ui` 的拓展组件，也有很多相关的 qq 讨论群或者 [gitter](#)。
- 社区的认可:目前Element已经是vue相关最多star的开源项目了，体现出了社区对其的认可。



04

第四部分

前端框架开发实战

实际开发案例代码及编写，前端工具使用介绍



一个后台前端解决方案，它基于 **vue** 和 **element-ui** 实现。它使用了最新的前端技术栈，动态路由，权限验证，提炼了典型的业务模型，提供了丰富的功能组件，它可以帮助你快速搭建企业级中后台产品原型。

浏览器支持

 IE / Edge	 Firefox	 Chrome	 Safari
IE10, IE11, Edge	last 2 versions	last 2 versions	last 2 versions

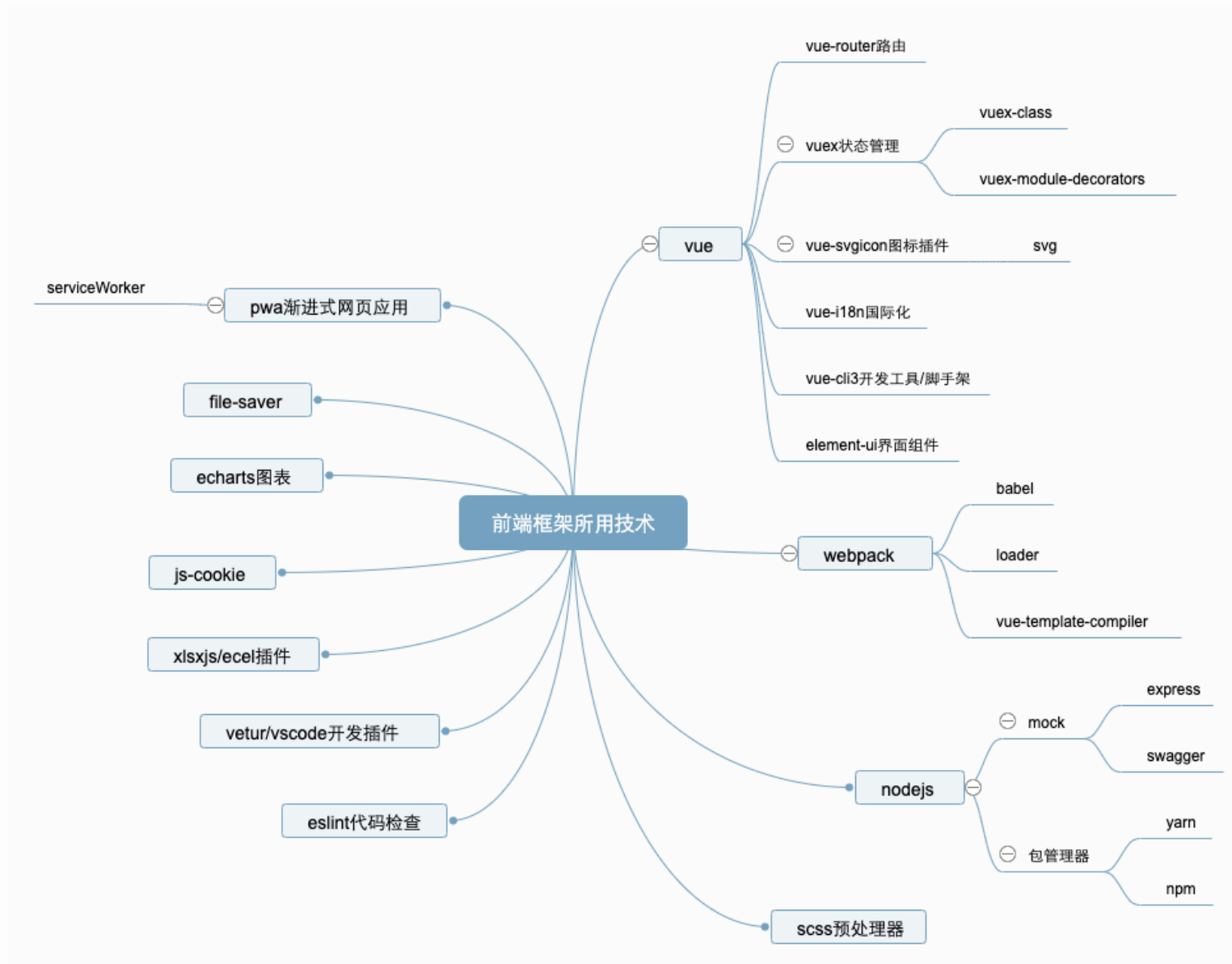
前端框架技术架构图

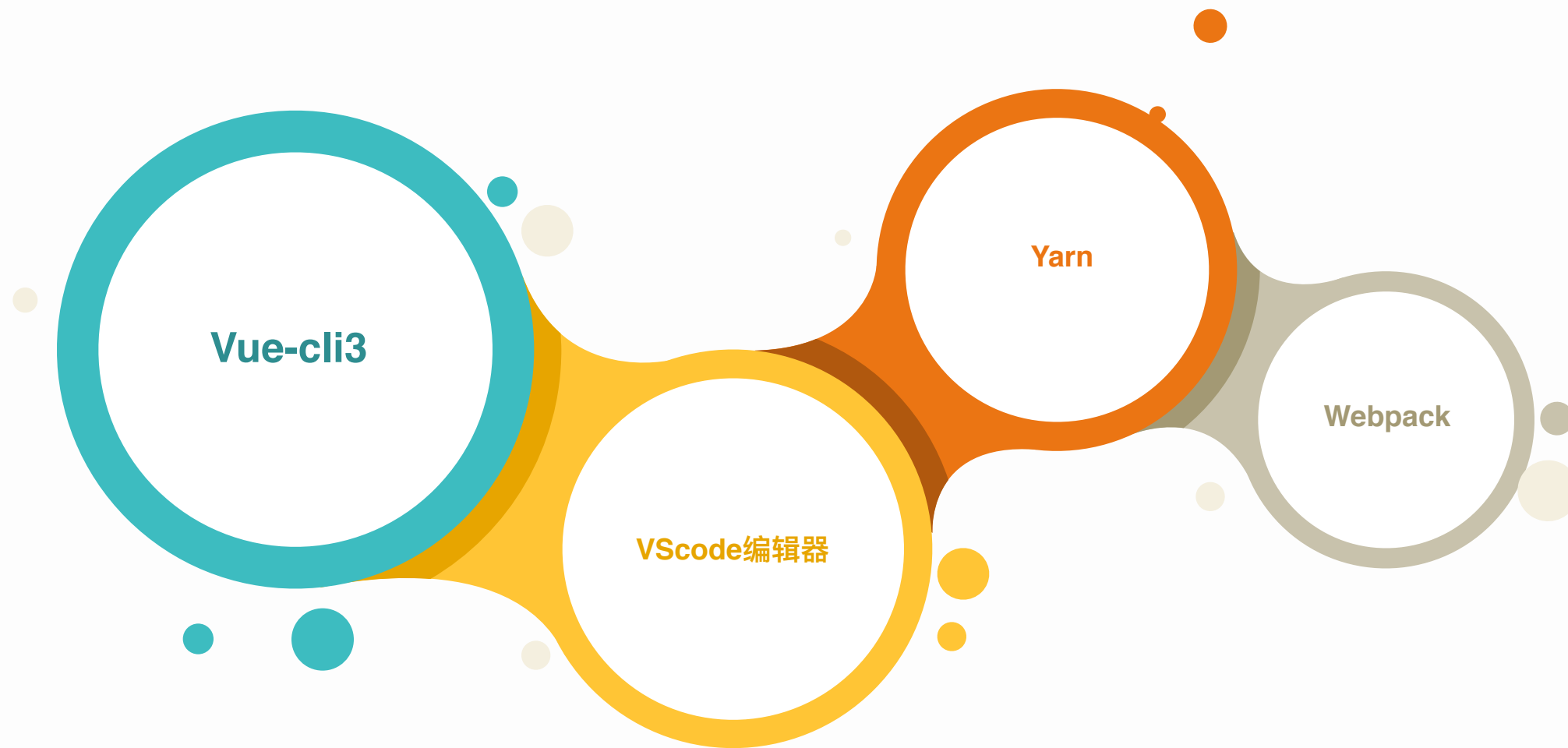
所用到的前端技术栈

本项目技术栈基于 ES2015+、vue、vuex、vue-router、vue-cli、axios 和 element-ui。提前了解和学习这些知识会对使用本项目有很大的帮助。

所需环境

你需要在本地安装 node 和 git, nodejs-v10.0以上, npm6.0以上, 全局vue-cli





Vue CLI 是一个基于 Vue.js 进行快速开发的完整系统，提供：

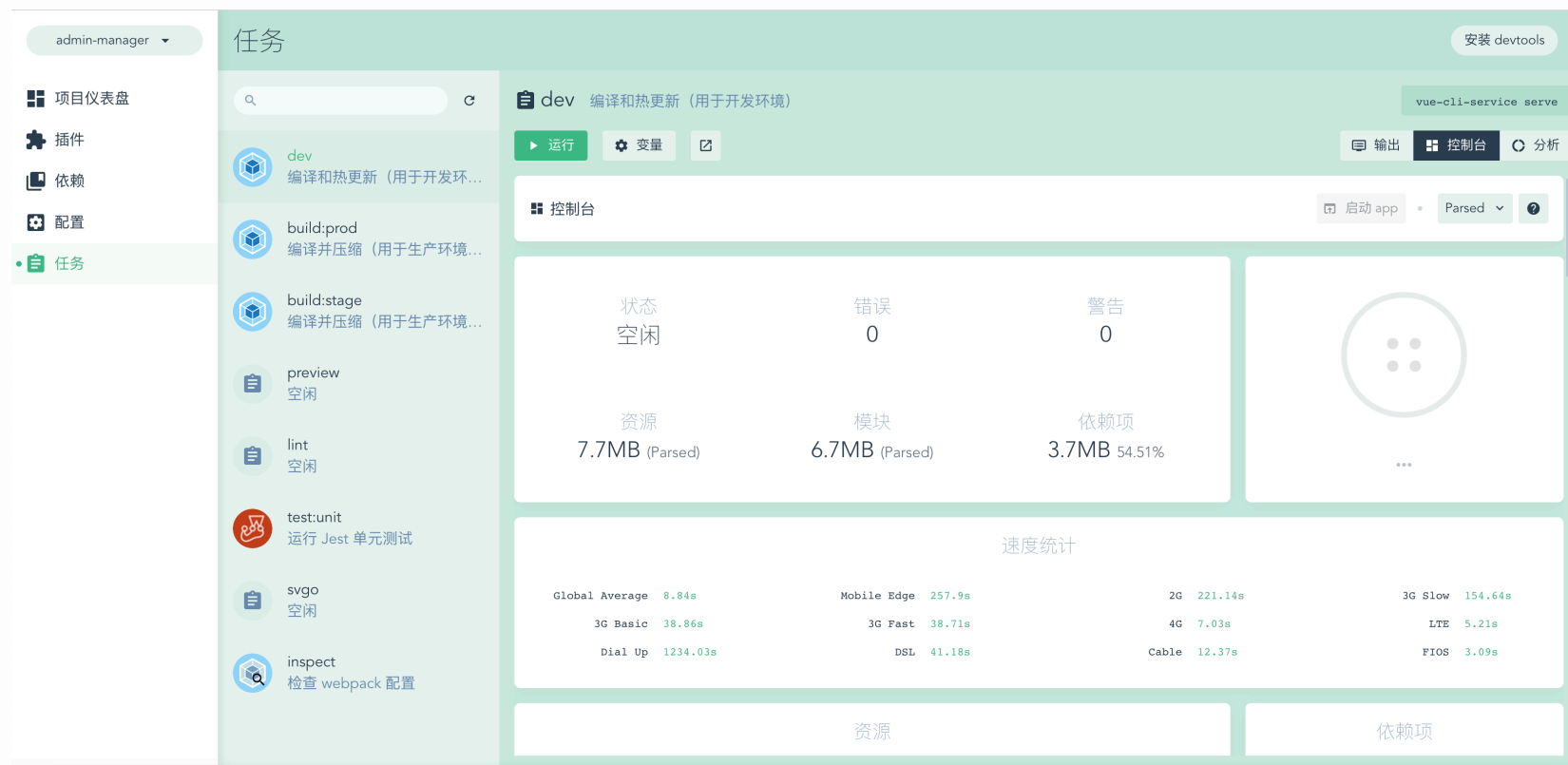
- 通过 `@vue/cli` 搭建交互式的项目脚手架。
- 通过 `@vue/cli + @vue/cli-service-global` 快速开始零配置原型开发。
- 一个运行时依赖 (`@vue/cli-service`)，该依赖：
 - 可升级；
 - 基于 webpack 构建，并带有合理的默认配置；
 - 可以通过项目内的配置文件进行配置；
 - 可以通过插件进行扩展。
- 一个丰富的官方插件集合，集成了前端生态中最好的工具。
- 一套完全图形化的创建和管理 Vue.js 项目的用户界面。

Vue CLI 致力于将 Vue 生态中的工具基础标准化。它确保了各种构建工具能够基于智能的默认配置即可平稳衔接，这样你可以专注在撰写应用上，而不必花好几天去纠结配置的问题。与此同时，它也为每个工具提供了调整配置的灵活性。

可视化cli管理界面

Vue CLI 还提供一个可视化的管理界面：

- 终端输入vue ui指令
- 导入你的项目
- 开始你的表演



市面主流编辑器介绍



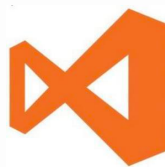
ATOM: 来自github的编辑器。这款编辑器的优点是主题多，插件多，子窗口可以随意组合，十分自由。



SublimeText3: 非常多前端使用的编辑器，轻量级，快速启动，丰富的插件，安装方便。这是一款成名已久的编辑器，各方面中规中矩。没有明显的缺点和优点。



Brackets: 来自adobe的编辑器，继承了adobe的优秀传统，自带即时预览和众多好用的插件。在其他方面没什么优势。



VS Code: 来自微软的编辑器，被称作披着编辑器外衣的IDE，自带debug模块和git模块，除了涵盖前端开发的各个方面，同时各种后端代码的高亮甚至调制都有相应插件支持，可谓是相当强大。



Hbuilder: 国产优秀IDE，基于eclipse，有很丰富的配置选项，这也是Hbuilder作为一个IDE和其他几个编辑器很大的区别。同时支持边看边改模式Hbuilder最强大的是可以很方便做移动端开发，甚至直接打包hybrid应用。

1.安装下载

<https://code.visualstudio.com/Download> 下载后一路点下一步安装

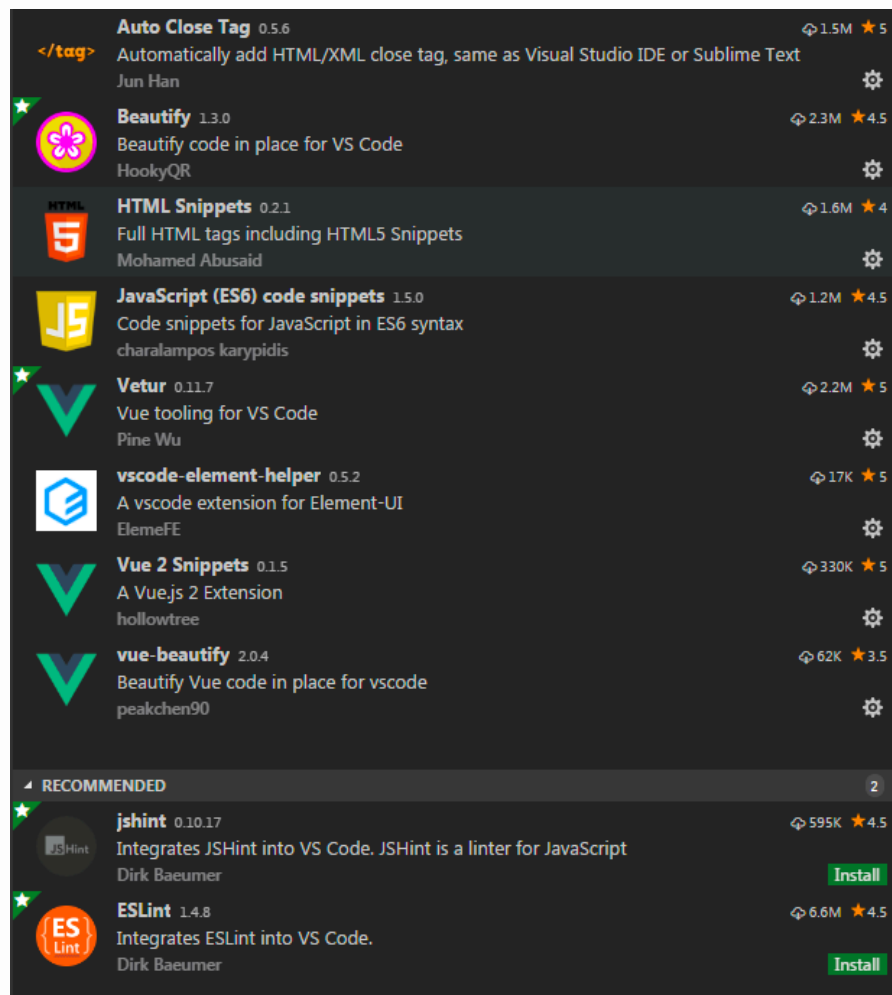
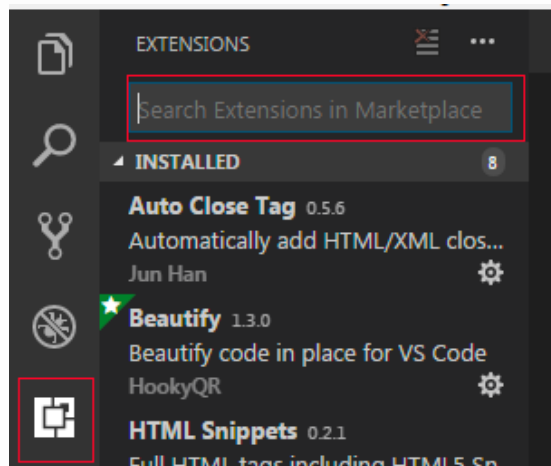
2.修改语言

安装后打开编辑器，按 ctrl+shift+p 在输入框中输入configure language，修改locale为 zh-CN点击保存后重启软件

```
{ } locale.json x
1  {
2      // Defines VSCode's display language.
3      // See https://go.microsoft.com/fwlink/?LinkId=761051 for a list of supported languages.
4      // Changing the value requires restarting VSCode.
5      "locale": "zh-CN"
6  }
```


3. 插件安装

在左边菜单栏中选择最后一个，然后再输入框中搜索下载心仪的插件，下面有一些推荐下载的插件



[Yarn](#)对你的代码来说是一个包管理器。

通过Yarn你可以使用其他开发者针对不同问题的解决方案，使自己的开发过程更简单。使用过程中遇到问题，你可以将其上报或者贡献解决方案。一旦问题被修复，Yarn会更新保持同步。

代码通过 **包 (package)** (或者称为 **模块 (module)**) 的方式来共享。一个包里包含所有需要共享的代码，以及描述包信息的文件，称为 `package.json`。

这里是用 [vue-cli](#) 的 [webpack-template](#) 为基础模板构建的，简单说一些需要注意到地方。

当项目逐渐变大之后，文件与文件直接的引用关系会很复杂，这时候就需要使用[alias](#) 了。有的人喜欢alias 指向src目录下，再使用相对路径找文件

```
resolve: {  
  alias: {  
    '~': resolve(__dirname, 'src')  
  }  
}
```

//使用

```
import stickTop from '~/components/stickTop'
```

多环境

vue-cli 默认只提供了 `dev` 和 `prod` 两种环境。但其实正真的开发流程可能还会多一个 `sit` 或者 `stage` 环境，就是所谓的测试环境和预发布环境。所以我们就要简单的修改一下代码。其实很简单就是设置不同的环境变量

```
"build:prod": "NODE_ENV=production node build/build.js",  
"build:sit": "NODE_ENV=sit node build/build.js",
```

之后在代码里自行判断，想干就干啥

```
var env = process.env.NODE_ENV === 'production' ? config.build.prodEnv : config.build.sitEnv
```

安装与部署

可以通过阅读项目里的Readme文件获取项目详细部署启动方法，下面做一个简介：

```
# 进入项目目录
cd “项目文件夹”
# 包管理器有两种yarn/npm，我们先讲npm

# 安装依赖
npm install
# 可以通过如下操作解决 npm 下载速度慢的问题
npm install --registry=https://registry.npm.taobao.org
# 本地开发 启动项目
npm run dev
# 生成生产版
npm run build:prod
# 生产文件预览
npm run preview
```

TIP

强烈建议不要用直接使用 cnpm 安装，会有各种诡异的 bug，可以通过重新指定 registry 来解决 npm 安装速度慢的问题。若还是不行，可使用 yarn 替代 npm。
Windows 用户若安装不成功，很大概率是 node-sass 安装失败。解决方案：另外因为 node-sass 是依赖 python 环境的，如果你之前没有安装和配置过的话，需要自行查看一下相关安装教程。

启动完成后会自动打开浏览器访问 <http://localhost:9527>，你看到下面的页面就代表操作成功了。

系统登录

 请输入域

 用户名

 密码



登 录

用户名: admin 域: tc-admin 密码: 123

布局

页面整体布局是一个产品最外层的框架结构，往往会包含导航、侧边栏、面包屑以及内容等。
想要了解一个后台项目，先要了解它的基础布局。



项目中大部分页面都是基于这个 `layout` 的，除了个别页面如：

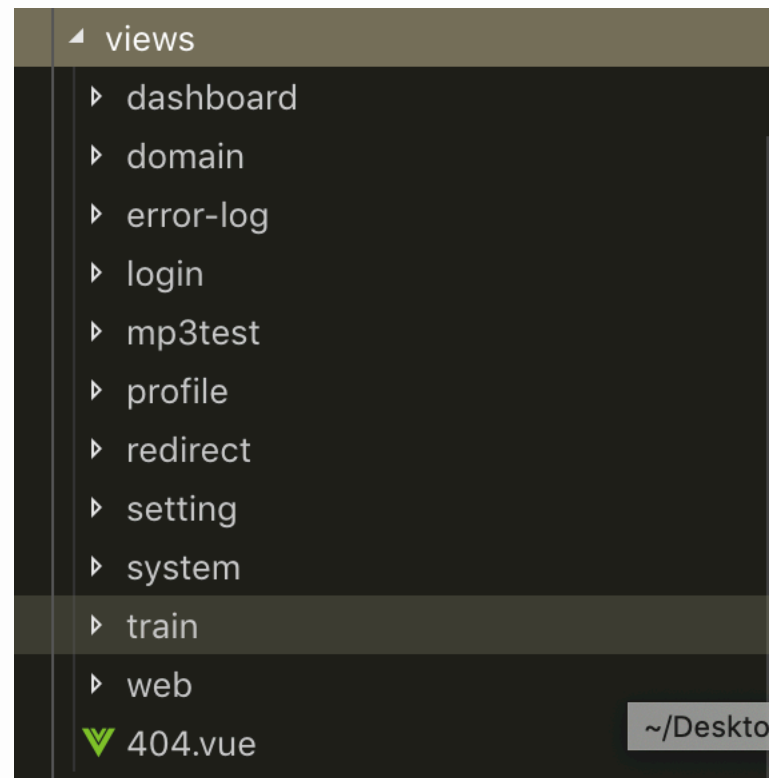
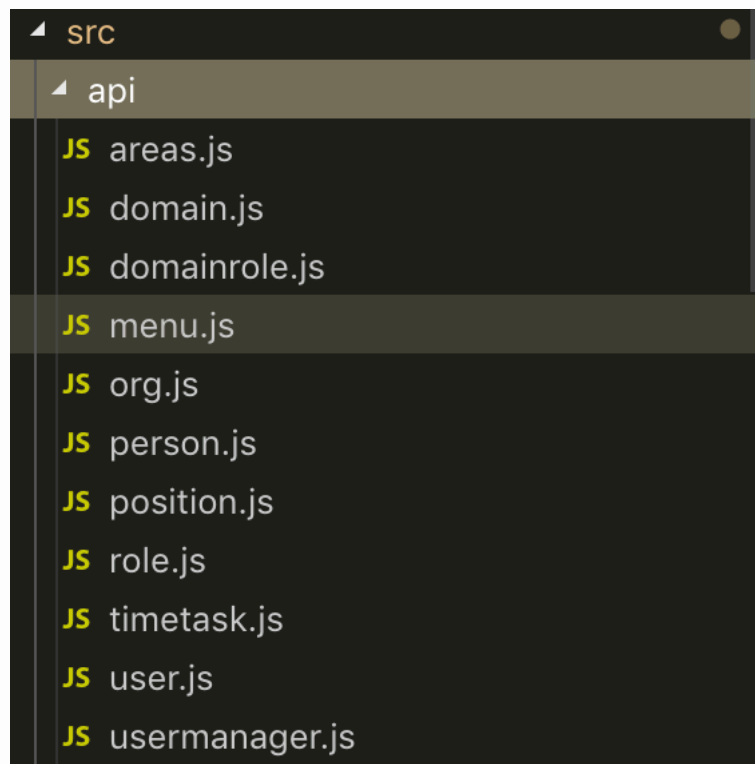
`login` , `404`, `401` 等页面没有使用该 `layout`。如果你想在一个项目中有多种不同的 `layout` 也是很方便的，只要在一级路由那里选择不同的 `layout` 组件就行。

目录结构

```
```bash
├── public # 静态资源（会被直接复制）
│ ├── favicon.ico # favicon图标
│ ├── manifest.json # PWA 配置文件
│ └── index.html # html模板
├── src # 源代码
│ ├── api # 所有请求
│ ├── assets # 主题 字体等静态资源（由 webpack 处理加载）
│ ├── components # 全局组件
│ ├── directive # 全局指令
│ ├── filters # 全局过滤函数
│ ├── icons # svg 图标
│ ├── lang # 国际化
│ ├── layout # 全局布局
│ ├── pwa # PWA service worker 相关的文件
│ ├── router # 路由
│ ├── store # 全局 vuex store
│ ├── styles # 全局样式
│ ├── vendor # 第三方插件
│ ├── utils # 全局方法
│ ├── views # 所有页面
│ ├── App.vue # 入口页面
│ ├── main.js # 入口文件 加载组件 初始化等
│ ├── permission.js # 权限管理
│ └── settings.js # 设置文件
├── tests # 测试
├── .browserslistrc # browserslistrc 配置文件（用于支持 Autoprefixer）
├── .editorconfig # 编辑相关配置
├── .env.xxx # 环境变量配置,配置不同环境下的一些路径参数,如接口地址
├── .eslintignore # 忽略lint配置
├── .eslintrc.js # eslint 配置
├── babel.config.js # babel-loader 配置
├── jest.config.js # jest 单元测试配置
├── package.json # package.json 依赖
├── postcss.config.js # postcss 配置
├── tsconfig.json # typescript 配置
└── vue.config.js # vue-cli 配置
```
```


src文件api 和 views

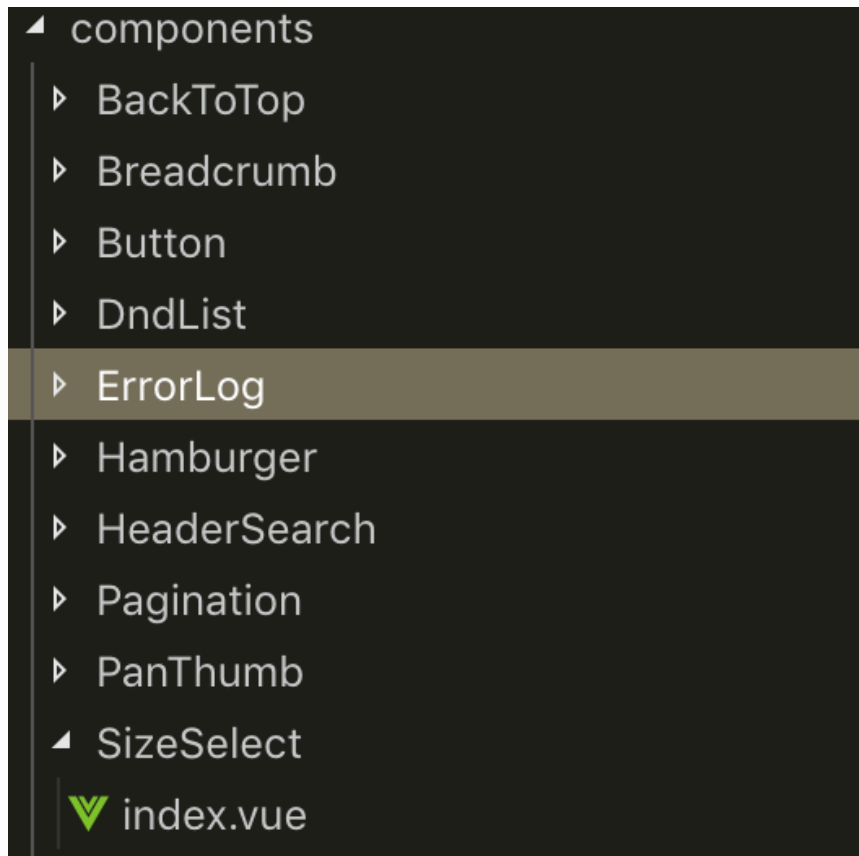
简单截取一下公司后台项目的api模块和views模块



views 和 api 两个模块对应，从而方便维护

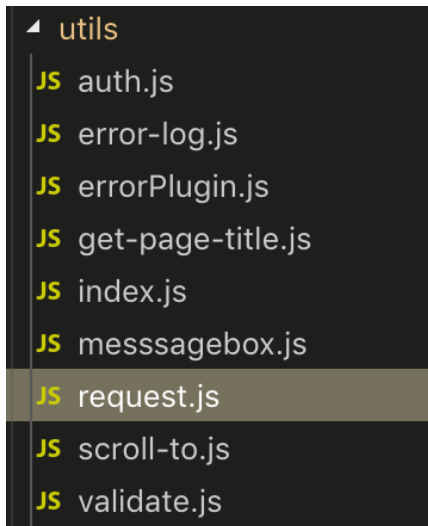
components

这里的 components 放置的都是全局公用的一些组件，如错误日志，面包屑，富文本等等。一些页面级的组件建议还是放在各自views文件下，方便管理。如图：



前后端交互

我们在utils下的request.js中，里面有针对项目的公共请求方法，用的时候直接在页面中import进来



```
import { request } from "@utils/request"

export const getLists = (params) => {
  return request('/api-admin/tip/rest/dm/list', 'list', params)
}

export const insert = (params) => request('/api-admin/tip/rest/dm/insert', 'insert', params)
export const update = (params) => request('/api-admin/tip/rest/dm/update', 'update', params)
```

```
import * as api from "@api/domain";
import {
  searchSelector,
  getDomainRoles,
  setDomainRoles
} from "@api/domainrole";
```

关于跨域问题，在 dev 开发模式下可以下使用 webpack 的 proxy 使用也是很方便，但这种方法在生产环境是不能使用的。在生产环境中需要使用 nginx 进行反向代理。

| 开发环境 | 生产环境 |
|-------|-------|
| cors | cors |
| proxy | nginx |

Layout

```
// No layout
{
  path: '/401',
  component: () => import('errorPage/401')
}

// Has layout
{
  path: '/documentation',

  // 你可以选择不同的layout组件
  component: Layout,

  // 这里开始对应的路由都会显示在app-main中 如上图所示
  children: [{
    path: 'index',
    component: () => import('documentation/index'),
    name: 'documentation'
  }]
}
```

项目中大部分页面都是基于这个 `layout` 的，除了个别页面如： `login` , `404`, `401` 等页面没有使用该 `layout`。

如果你想在项目中有多种不同的 `layout` 也是很方便的，只要在一级路由那里选择不同的 `layout` 组件就行。

这里使用了 `vue-router` 路由嵌套，所以一般情况下，你增加或者修改页面只会影响 `app-main` 这个主体区域。其它配置在 `layout` 中的内容如：侧边栏或者导航栏都是不会随着你主体页面变化而变化的。

本项目里的侧边栏是根据 router.js 配置的路由并且根据权限动态生成的，这样就省去了写一遍路由还要手动再写一次侧边栏这种麻烦事

- 侧边栏高亮问题:

element-ui官方已经给了default-active所以我们只要

:default-active="\$route.path"将default-active一直指向当前路由就可以了，就是这么简单。

- 点击侧边栏 刷新当前路由

在用 spa(单页面开发) 这种开发模式之前，大部分都是多页面后台，用户每次点击侧边栏都会重新请求这个页面，用户渐渐养成了点击侧边栏当前路由来刷新页面的习惯。但现在 spa 就不一样了，用户点击当前高亮的路由并不会刷新view，因为vue-router会拦截你的路由，它判断你的url并没有任何变化，所以它不会触发任何钩子或者是view的变化。

redirect 刷新页面

在不刷新页面的情况下，更新页面。

当遇到你需要刷新页面的情况，你就手动重定向页面到redirect页面或配置重定向路由，它会将页面重新redirect重定向回来，由于页面的 key 发生了变化，从而间接实现了刷新页面组件的效果。

```
// 手动重定向页面到 '/redirect' 页面
```

```
const { fullPath } = this.$route
```

```
this.$router.replace({
```

```
  path: '/redirect' + fullPath
```

```
})
```

```
path: '/web/train',
component: 'Layout',
hidden: false,
redirect: '/web/train/team',
meta: {
  title: '培训',
  icon: 'nested'
},
children: [
  {
    path: 'team',
    hidden: false,
    component: 'views/web/train/train/team/team/index',
    meta: {
```



谢谢观看