## 项目搭建

vue init webpack programName

## 项目知识

index 主页

asserts 资源集

components 组件集

plugins 插件集

utils 工具集

views 页面集

App.vue

static 静态资源

config 基础配置

router 路由集

build文件夹里面是对 webpack 开发和打包的相关设置，包括入口文件、输出文件、使用的模块等；

dependencies：应用能够正常运行所依赖的包。

devDependencies：开发应用时所依赖的工具包。

export default{} //{}内的内容被向外导出

Vue.use() 安装 Vue.js 插件

在vue组件中，主要由template，script，style三部分组成

template 三种写法

<myTemplate/>

<my-template></my-template>

<myTemplate></myTemplate>

### 组件通信

父子组件通信：prop(父传子)、export(子传父)、$emit、ref

vuex共享信息通信：state==data mutation==method

Eventbus

localStorage

### 技巧

rem为元素设定字体大小时，仍然是相对大小，但相对的只是HTML根元素

注意: 由::before 和::after 生成的伪元素 包含在元素格式框内， 因此不能应用在替换元素上， **比如<img>或<br> 元素**。

#### 防止掉块：padding-bottom

例：padding-bottom：50%；%定义基于父元素**宽度**的百分比下内边距。行内非替换元素上设置的下内边距不会影响行高计算

#### 样式穿透

用scope是防止污染全局css但这样会导致无法修改插件的css，此时在父类与要修改的子类间加>>>

#### img

img需要单独设置宽高，否则可能溢出

**img使用本地资源时需要require(localPosition),而不是直接使用localPosition**

text-align：center 使文本居中

line-height 设置行高可以达到不增大文本的同时增加行内高度

父级relative子级absolute 子级基于父级(去掉该子级后)的位置

### axios使用及传值

**将所有请求集中到父类，再由父类根据子组件的属性绑定传给子组件的props**。

具体来说，先在mai.js中引入并设置axios的请求方式

import axios from 'axios'

Vue.prototype.$http =axios;//设置$http为Vue中请求axios的方法

再在请求的父类中的搭载阶段使用

mounted(){

this.$http.get("http://localhost:8080/static/mock/dataHome.json")

.then((res)=>{//处理请求到的data

})

}

### 文件脉络清理

index.html 项目的入口html文件

main.js 项目的入口js文件（导入了App等各种插件

App.vue 项目的入口Vue文件（各种vue文件的集合）

babel.config.js 若在vue项目中则为Vue项目兼容的编译器

yarn.lock 跨机安装所要求的详细配置(比package.json更详细，要求配置的版本号)

package.json 定义这个项目所需要的各种模块，以及项目的配置信息（比如名称、版本、许可证等元数据）。

license 项目的许可证(规定项目的开源类型)

git 是一个开源的分布式版本控制系统，处理从很小到非常大的项目版本管理

setting.json vscode针对工作区域的配置文件

### 文件所需插件

npm

axios done

element-ui done

moment done

vue done

vue-router done

vue cli

## javascript模块

### module

### import

import Vue from 'vue'等效于import Vue from "../node\_modules/vue/dist/vue.js";

1.import...from...的from命令后面可以跟很多路径格式，若只给出vue，axios这样的包名，则会自动到node\_modules中加载；若给出相对路径及文件前缀，则到指定位置寻找。

2.可以加载各种各样的文件：.js、.vue、.less等等。

3.可以省略掉from直接引入。

**import Vue from ‘vue’ 解析为 const Vue = require(‘vue’)。**

### export

### require

## webpack

可设置特点字符指代引用文件位置

config中的index.js内设置代理项，即proxyTable

file-loader：把文件输出到一个文件夹中，在代码中通过相对 URL 去引用输出的文件

url-loader：和 file-loader 类似，但是能在文件很小的情况下以 base64 的方式把文件内容注入到代码中去

source-map-loader：加载额外的 Source Map 文件，以方便断点调试

image-loader：加载并且压缩图片文件

babel-loader：把 ES6 转换成 ES5

css-loader：加载 CSS，支持模块化、压缩、文件导入等特性

style-loader：把 CSS 代码注入到 JavaScript 中，通过 DOM 操作去加载 CSS。

eslint-loader：通过 ESLint 检查 JavaScript 代码

define-plugin：定义环境变量

commons-chunk-plugin：提取公共代码

uglifyjs-webpack-plugin：通过UglifyES压缩ES6代码

## vue-router

<router-link to=’’></router-link >实现路由之间的跳转 点击该标签时发生

< router-view/>当你的路由path 与指定的访问地址相符时，会将指定的组件替换该 router-view

### router原理

**“通过更改url更新视图但不重新请求页面”是前端路由原理的核心之一**

vue-router通过hash与History interface两种方式实现前端路由

history分为HashHistory和HTML5History

**HashHistory有push和replace方法来替换路由**

HashHistory.push() //将新路由添加到浏览器访问历史的栈顶，然后通过监测

在全局的window对象上触发hashChange事件并更新路由location.hash

实现跳转

HashHistory.replace()//将当前路由替换

**HTML5History**

History interface是浏览器历史记录栈提供的接口，通过back(), forward(), go()等方法，我们可以读取浏览器历史记录栈的信息，进行各种跳转操作。

html5后：pushState(), replaceState()还可以对浏览器历史记录栈进行修改

设置mode参数决定实现模式，默认hash

例：const router = new VueRouter({

mode: 'history',

routes: [...]

})

当请求路由不存在时返回404

## 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。

state 储存变量

action 提交一个mutation。可异步

mutation 提交更新数据的方法，必须是同步的

getter 从state派生的数据，相当于state计算属性

module 模块化

## axios

1.import axios；

2.添加进vue原型中 Vue.prototype.$functionName = axios;

3.在vue中搭载该方法并请求资源

mounted(){

this.$functionName.get(url).then((res)=>{

const data = res.xxx

});

}

## swiper

swiper6只支持vue3

且swiper6不支持分页器（原因在于，在第六版导入只能导入swiper核心内容，而部分组件效果如分页器需要单独导入）

解决方法：重新安装swiper npm install [swiper@5.2.1](mailto:swiper@5.2.1)

并且在main.js中改动引入样式

## Git

git常用命令

git init 初始化仓库

git push 上传远程代码并合并

git clone 复制一份仓库到本地(即下载)

git add 添加文件到暂存区

git commit 提交暂存区到仓库

git checkout 创建分支和切换分支，也可以称为检出分支。

git merge 合并分支

## babel

# node.js

## npm

which node // 查看node的安装目录

which npm // 查看npm的安装目录

npm root -g // 查看全局包的安装目录

npm list -g --depth 0 //查看全局安装过的包

### 全局安装

npm install <pageName> -g //（这里-g是-global的简写）

通常全局包安装在node目录下的node\_modules文件夹。

全局安装的包可提供直接执行的命令。比如gulp全局安装后，可以在**命令行上直接执行**gulp -v、gulp -h等（原理：全局安装的gulp会将其package.json中的bin命令注入到了全局环境，使得你可以全局执行。

**如果只是全局安装了而没本地安装，就得require('<pagePath>')；如果安装了本地包，那么就可以直接require('<pageName>')引入使用。**

### 本地安装

npm install <pageName>

包安装在你当前项目文件夹下的node\_modules文件夹中。

可细分为2种形式

保存到开发依赖(devDependencies): npm install <pageName> --save-dev ；会被记录在package.json的dependencies。

保存到生产依赖(dependencies): npm install <pageName> --save；会被记录在package.json的devependencies

生成环境：指的是你的项目已经完成编码，并发布上线可供用户浏览的阶段时的环境。代码可能经过了压缩、优化等处理。

开发环境：指的是你的项目尚且在编码阶段时的环境。你在代码可能还有各种console.log()、注释、格式化等。