一、下载模板。

# 全局安装 vue-cli

$ npm install --global vue-cli

# 创建一个基于 webpack 模板的新项目

$ vue init webpack my-project

# 安装依赖，走你

$ cd my-project

$ npm run dev

二、引入axios。

1、安装axios :npm install axios

2、在main.js 中全局引入。

import axios from 'axios';

// 改写VUE的原型属性  
Vue.prototype.$ajax= axios;

1. 在组件中使用。

this.$ajax.get('api/getNewsList')

.then((response)=>{ this.newsList=response.data.data; })

.catch((response)=>{ console.log(response); })

1. 配置api路径及解决跨域请求。

使用http-proxy-middleware 代理解决（项目使用vue-cli脚手架搭建）

例如请求的url:“http://f.apiplus.cn/bj11x5.json”

4.1、打开config/index.js,在proxyTable中添写如下代码：

proxyTable: {

'/api': { //使用"/api"来代替"http://f.apiplus.c"

target: 'http://f.apiplus.cn', //源地址

changeOrigin: true, //改变源

pathRewrite: {

'^/api': 'http://f.apiplus.cn' //路径重写

}

}

}

4.2、使用axios请求数据时直接使用“/api”：

getData () {

axios.get('/api/bj11x5.json', function (res) {

console.log(res)

})

通过这中方法去解决跨域，打包部署时还按这种方法会出问题。解决方法如下：

let serverUrl = '/api/' //本地调试时

// let serverUrl = 'http://f.apiplus.cn/' //打包部署上线时

export default {

dataUrl: serverUrl + 'bj11x5.json'

}

调试时定义一个serverUrl来替换我们的“/api”，最后打包时，只需要将“http://www.xxx.com”替换这个“/api”就可以了。

三、改变测试IP。

在生成的问价中找到config-->index.js

修改 module.exports-->dev-->host值为'0.0.0.0'。

修改后就可以用本地IP访问了，用手机可以在局域网内查看。

四、引入scss。

1、安装sass的依赖包

npm install --save-dev sass-loader

//sass-loader依赖于node-sass

npm install --save-dev node-sass

2、在build文件夹下的webpack.base.conf.js的rules里面添加配置

{

test: /\.scss$/,

loaders: ['style', 'css', 'sass']

}

3、修改style标签

<style lang="scss">

修改运行成功后就可以使用scss了。

五、文件路径配置。

1、在引入资源时使用，别名alias，vue init webpack生成的项目在build/webpack.base.conf.js文件中进行更改

alias: {

'vue$': 'vue/dist/vue.esm.js', // 默认的

'@': resolve('src'), // 默认的

'assets': resolve('src/assets') // 自己定义的

}

之后再项目中就可以使用了

<img src="~assets/images/[图片名]"> // images: src/assets 下存放图片的文件夹

backgroud: url(~assets/images/[图片名]) js中 直接用 assets/...，不需要加 ~

2、打包时，可以将路径全部改成相对路径，个人认为有利于部署，以下配置为在使用上一条配置的情况下进行

2.1、修改config/index.js文件中build对象中的

assetsPublicPath: '/' // 改为：

assetsPublicPath: './'

2.2、修改build/utils.js, 针对css样式文件，因为在使用别名的情况下，打包出来的css文件中资源文件的引入方式为 static/\*

// Extract CSS when that option is specified

// (which is the case during production build)

if (options.extract) {

return ExtractTextPlugin.extract({

use: loaders,

fallback: 'vue-style-loader'

})

} else {

return ['vue-style-loader'].concat(loaders)

}

// 新增一条属性publicPath

// publicPath (打包时改用相对路径，在static/\*前面加上publicPath)

if (options.extract) {

return ExtractTextPlugin.extract({

use: loaders,

fallback: 'vue-style-loader',

publicPath: '../../'

})

} else {

return ['vue-style-loader'].concat(loaders)

}

六、引入自适应插件lib-flexible+px2rem-loader。

1、安装 lib-flexible

1.1、npm i lib-flexible --save。

1.2、引入 lib-flexible在项目入口文件 main.js 里 引入 lib-flexible

import 'lib-flexible/flexible.js'

1.3、添加meta标签，在项目根目录的 index.html 中添加如下 meta。

<meta name="viewport" content="width=device-width, initial-scale=1.0">

2、安装 px2rem-loader。

2.1、npm install px2rem-loader

2.2、配置 px2rem-loade。

在 vue-cli 生成的 webpack 配置中，vue-loader 的 options 和其他样式文件 loader 最终是都是由 build/utils.js 里的一个方法生成的。我们只需在 cssLoader 后再加上一个 px2remLoader 即可，px2rem-loader 的 remUnit 选项意思是 1rem=多少像素，结合 lib-flexible 的方案，我们将 px2remLoader 的 options.remUnit 设置成设计稿宽度的 1/10，这里我们假设设计稿宽为 750px。

// utils.js

const cssLoader = {

loader: 'css-loader',

options: {

minimize: process.env.NODE\_ENV === 'production',

sourceMap: options.sourceMap

}

}

const px2remLoader = {

loader: 'px2rem-loader',

options: {

remUnit: 75

}

}

// ...

并放进 loaders 数组中

// utils.js

function generateLoaders(loader, loaderOptions) {

var loaders = [cssLoader, px2remLoader]

// ...

修改配置后需要重启，然后我们在组件中写单位直接写 px，设计稿量多少就可以写多少了，舒服多了。

六、引入外部js 插件、vue-awesome-swiper。

安装vue-awesome-swiper:npm install vue-awesome-swiper --save

组件中使用：

html模板:

<!-- The ref attr used to find the swiper instance -->

<template>

<swiper :options="swiperOption" ref="mySwiper" @someSwiperEvent="callback">

<!-- slides -->

<swiper-slide>I'm Slide 1</swiper-slide>

<swiper-slide>I'm Slide 2</swiper-slide>

<swiper-slide>I'm Slide 3</swiper-slide>

<swiper-slide>I'm Slide 4</swiper-slide>

<swiper-slide>I'm Slide 5</swiper-slide>

<swiper-slide>I'm Slide 6</swiper-slide>

<swiper-slide>I'm Slide 7</swiper-slide>

<!-- Optional controls -->

<div class="swiper-pagination" slot="pagination"></div>

<div class="swiper-button-prev" slot="button-prev"></div>

<div class="swiper-button-next" slot="button-next"></div>

<div class="swiper-scrollbar" slot="scrollbar"></div>

</swiper>

</template>

script:

// require styles

import 'swiper/dist/css/swiper.css'

import { swiper, swiperSlide } from 'vue-awesome-swiper'

export default {

name: 'home',

components: {

swiper,

swiperSlide

},

data() {

return {

swiperOption: {

// some swiper options/callbacks

// 所有的参数同 swiper 官方 api 参数

// ...

loop : true,

autoplay: {

delay: 3000,

stopOnLastSlide: false,

disableOnInteraction: true,

},

pagination: {

el: '.swiper-pagination',

clickable :true,

}

}

}

},

computed: {

swiper() {

return this.$refs.mySwiper.swiper

}

},

mounted() {

// current swiper instance

// 然后你就可以使用当前上下文内的swiper对象去做你想做的事了

console.log('this is current swiper instance object', this.swiper)

this.swiper.slideTo(3, 1000, false)

}

}