

aws

ssh aws@free-tier

博客园cnblogs.com

首页 新闻 博文 专区 闪存 班级 代码改变世界

注册 登录

既然选择远方，注定风雨兼程！

博客园 首页 新随笔 联系 管理 订阅

随笔- 17 文章- 0 评论- 56

Vue项目的搭建 (vue cli2)

前言：使用vue cli2 搭建前端项目而整理的一份文档

一、项目环境

1、node安装

下载地址：<http://node.js.cn/download/>

nodejs

首页 下载 文档 搜索 腾讯社招

下载

最新的长期支持版本: 10.16.3

v10.16.3

推荐大多数用户使用

Windows 安装包

node-v10.16.3-x64.msi

v12.8.1

最新的特性

macOS 安装包

node-v10.16.3.pkg

源代码

node-v10.16.3.tar.gz

腾讯云

热门云产品限量秒杀

Windows 安装包 (.msi)	32 位	64 位	
Windows 二进制文件 (.zip)	32 位	64 位	
macOS 安装包 (.pkg)	64 位		
macOS 二进制文件 (.pkg)	64 位		
Linux 二进制文件 (x64)	32 位		
Linux 二进制文件 (ARM)	ARMv6	ARMv7	ARMv8
Docker 镜像	官方镜像		
全部安装包	阿里云镜像		

安装检查：

命令：

node -v

npm -v

注：若需更改node模块 (node\_modules) 的安装目录自行百度，由于创建的Vue项目中node模块是安装在项目文件夹下就不做更改了。

若使用淘宝npm镜像 (cnpm) 通过以下命令安装：

昵称： 墨道

园龄： 3年

粉丝： 15

关注： 1

+加关注

SEPHORA x 博客园

为他选礼 实用更贴心

{ 驻守程序员魅力阵地 }

LOEWE

LAB SERIES

SEPHORA

< 2020年12月 >

日	一	二	三	四	五	六
29	30	1	2	3	4	5
6	7	8	9	10	11	12
13	14	15	16	17	18	19
20	21	22	23	24	25	26
27	28	29	30	31	1	2
3	4	5	6	7	8	9

搜索

找找看

谷歌搜索

常用链接

我的随笔

我的评论

我的参与

最新评论

我的标签

我的标签

Cordova(2)

前端(2)

Linux配置(1)

积分与排名

积分 - 29100

排名 - 35570

https://www.cnblogs.com/VoiceOfDreams/p/11442650.html

1/16

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

2、Vue-CLI安装

1）、vue-cli@2.X版本

官方文档：<https://github.com/vuejs/vue-cli/tree/v2#vue-cli-->

注：使用的一些UI组件不适配vue-cli@3.x，所以需要安装vue-cli@2.x

比如VUX

VUX 必须配合 `vux-loader` 使用，如果不使用 `vux2` 模板请按照[文档](#)正确配置。  
`less@3.x` 有严重的兼容问题，请暂时使用 `less@^2.7.3`。  
暂未适配 `vue-cli@3.x`，请知悉。

安装命令：

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

查看版本：

```
$ vue -version
```

使用：

```
$ vue init <template-name> <project-name>
```

例如：

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

将从vuejs-templates/webpack中提取模板信息，根据此模板信息生成项目（./my-project/）

2）、@vue/cli 3.x 版本

官方文档：<https://cli.vuejs.org/zh/guide/>

安装命令：

```
$ npm install -g @vue/cli
```

查看版本：

```
$ vue --version
```

使用：

```
$ vue create <project-name>
```

警告

如果你在 Windows 上通过 minTTY 使用 Git Bash，交互提示符并不工作。你必须通过 `winpty` `vue.cmd create hello-world` 启动这个命令。不过，如果你仍想使用 `vue create hello-world`，则可以通过在 `~/.bashrc` 文件中添加以下行来为命令添加别名。`alias vue='winpty vue.cmd'` 你需要重新启动 Git Bash 终端会话以使更新后的 `bashrc` 文件生效。

二、项目创建

本项目使用的vue-cli版本：2.9.6

1、创建本地项目

在本地/d/Projects目录下创建名为project-demo的项目



随笔分类

Linux(7)

前端(7)

数据库(4)



随笔档案

2020年4月(1)

2019年11月(4)

2019年9月(1)

2019年6月(2)

2019年1月(2)

2018年8月(1)

2018年3月(2)

2018年1月(4)



最新评论

1. Re:Cordova 项目从 UIWebView 更换为 WKWebView

cordova plugin add ios@6.1.0 cordova plugin add  
按第二个插件更改config.xml文档配置就可以...

--shuixianliyu

2. Re:Rlwrap工具的安装和配置

问下你的那个rlwrap包在哪下载的？谢谢！

--smile-you-me

3. Re:Vue+webpack项目的多环境打包配置(vue-cli 2.x)

文章很详细

--藏在心

4. Re:CentOS 7安装

感谢您的CentOS7 Oracle 安装文档，非常有用，安装成功，介绍的非常详细

感谢您的分享，感谢感谢

--haohaoxyz

5. Re:Cordova 项目从 UIWebView 更换为 WKWebView

@0J 我碰到跟你一样的问题，你是怎么解决的...

--blackworm



阅读排行榜

1. Oracle GoldenGate实现数据库同步(28627)

2. CentOS 7安装Oracle 11gR2以及设置自启动(6872)

3. Cordova 项目从 UIWebView 更换为 WKWebView (5728)

4. Cordova自定义插件开发(4353)

5. Vue+webpack项目的多环境打包配置(vue-cli 2.x) (4147)



评论排行榜

1. Cordova 项目从 UIWebView 更换为 WKWebView (51)

2. Rlwrap工具的安装和配置(3)

3. Vue+webpack项目的多环境打包配置(vue-cli 2.x) (1)

4. CentOS 7安装(1)



推荐排行榜

1. Oracle GoldenGate实现数据库同步(9)

2. CentOS 7安装Oracle 11gR2以及设置自启动(4)

3. Tree命令的使用(1)

4. Vue项目的搭建（vue cli2）(1)

5. Vue+webpack项目的多环境打包配置(vue-cli 2.x) (1)

```
DELL@DESKTOP-8M3VQSF MINGW64 /d/Projects
$ vue init webpack project-demo

? Project name project-demo
? Project description demo
? Author 
? Vue build
  Runtime + Compiler: recommended for most users
  > Runtime-only: about 6KB lighter min+gzip, but templates (or any Vue-specific HTML) are ONLY allowed in .vue files - render functions are required elsewhere
```

Vue build

第一个选项可以不基于.vue文件开发，在运行时进行编译

第二个选项基于.vue文件开发，比第一个选项小6KB

```
DELL@DESKTOP-8M3VQSF MINGW64 /d/Projects
$ vue init webpack project-demo

? Project name project-demo
? Project description demo
? Author 
? Vue build runtime
? Install vue-router? Yes
? Use ESLint to lint your code? Yes
? Pick an ESLint preset (Use arrow keys)
  > Standard (https://github.com/standard/standard)
  Airbnb (https://github.com/airbnb/javascript)
  none (configure it yourself)
```

使用vue-router

使用ESLint进行代码检测 (ESLint 是一个语法规则和代码风格的检查工具，可以用来保证写出语法正确、风格统一的代码。)

ESLint代码检测规则(本项目使用Standard)，官方文档连接：

Standard:

<https://github.com/standard/standard/blob/master/docs/README-zhcn.md>

Airbnb: <https://github.com/airbnb/javascript>

```
DELL@DESKTOP-8M3VQSF MINGW64 /d/Projects
$ vue init webpack project-demo

? Project name project-demo
? Project description demo
? Author 
? Vue build runtime
? Install vue-router? Yes
? Use ESLint to lint your code? Yes
? Pick an ESLint preset Standard
? Set up unit tests No
? Setup e2e tests with Nightwatch? No
? Should we run `npm install` for you after the project has been created? (recommended) (Use arrow keys)
  > Yes, use NPM
  Yes, use Yarn
  No, I will handle that myself
```

```
Running eslint --fix to comply with chosen preset rules...
# =====

> project-demo@1.0.0 lint D:\Projects\project-demo
> eslint --ext .js,.vue src "--fix"

# Project initialization finished!
# =====

To get started:

  cd project-demo
  npm run dev

Documentation can be found at https://vuejs-templates.github.io/webpack
```

安装完成后运行上述命令：

```
$ cd project-demo
$ npm run dev
```

根据运行成功后的地址在浏览器进行访问

2、目录结构如下：

```

.
|-- README.md
|-- build
|   |-- build.js
|   |-- check-versions.js
|   |-- logo.png
|   |-- utils.js
|   |-- vue-loader.conf.js
|   |-- webpack.base.conf.js
|   |-- webpack.dev.conf.js
|   `-- webpack.prod.conf.js
|-- config
|   |-- dev.env.js
|   |-- index.js
|   `-- prod.env.js
|-- index.html
|-- package-lock.json
|-- package.json
|-- src
|   |-- App.vue
|   |-- assets
|   |   |-- logo.png
|   |-- components
|   |   |-- HelloWorld.vue
|   |-- main.js
|   |-- router
|   |   |-- index.js
|-- static

```

### 3、将本地仓库与线上仓库关联：

```

$ cd /project-demo
$ git init
$ git remote add origin <线上git仓库地址 (SSH) >
$ git add .
$ git commit
$ git push -u origin master

```

## 三、项目初始化配置

附：使用**Visual Studio Code**推荐安装以下扩展程序

```

1)  Vetur
2)  ESLint
3)  Beautify
4)  GitLens - Git supercharged
5)  Git History
6)  HTML CSS Support
7)  HTML Snippets
8)  Vue 2 Snippets
9)  Debugger for Chrome

```

附：目录结构

项目结构多采用vue-element-admin项目结构：

<https://panjiachen.gitee.io/vue-element-admin-site/zh/guide/>

```

|-- README.md
|-- build                # 项目打包配置
|-- config              # 项目配置
|-- index.html          # html模板

```

```

|-- package-lock.json      # postcss 配置
|-- package.json           # package.json
|-- src                   # 源代码
|   |-- App.vue            # 入口页面
|   |-- api                # 所有请求
|   |-- assets             # 主题 字体等静态资源
|   |-- axios              # axios通用配置
|   |-- components        # 全局公用组件
|   |-- directive         # 全局指令
|   |-- filters            # 全局 filter
|   |-- layout             # 全局 layout
|   |-- main.js            # 入口文件 加载组件 初始化等
|   |-- router             # 路由
|   |-- store              # vuex配置, 全局 store管理
|   |-- styles             # 全局样式、自定义主题等
|   |-- utils              # 全局公用方法
|   |-- views              # 所有页面
|-- static

```

## 1、ESLint规则配置

修改.eslintrc.js文件配置, 在rules对象中添加以下配置(其他配置默认):

```

// 在函数括号之前不允许空格
'space-before-function-paren': ['error', 'never']

// add your custom rules here
rules: {
  // allow async-await
  'generator-star-spacing': 'off',
  // allow debugger during development
  'no-debugger': process.env.NODE_ENV === 'production' ? 'error' : 'off',
  // 在函数括号之前不允许空格
  'space-before-function-paren': ['error', 'never']
}

```

修改package.json文件配置:

```

"scripts": {
  "dev": "webpack-dev-server --inline --progress --config build/webpack.dev.conf.js",
  "start": "npm run dev",
  "lint": "eslint --fix --ext .js,.vue src",
  "build": "node build/build.js"
},

```

添加后运行以下命令可以快速修复ESLint报错:

```
$ npm run lint
```

## 2、UI组件安装

### 1)、PC端(element-ui)

官方文档: <https://element.eleme.cn/#/zh-CN/component/installation>

npm安装:

```
$ npm i element-ui -S
```

Element完整引入

在 main.js 中写入以下内容:



```
import Vue from 'vue'

import ElementUI from 'element-ui'
import 'element-ui/lib/theme-chalk/index.css'

import App from './App'
import router from './router'

Vue.config.productionTip = false

Vue.use(ElementUI)

/* eslint-disable no-new */
new Vue({
  el: '#app',
  router,
  render: h => h(App)
})
```

## 2)、移动端(VUX/Mint UI)

VUX文档: <https://doc.vux.li/zh-CN/>

Mint UI文档: <https://mint-ui.github.io/docs/#/zh-cn2>

## 3、css预处理器安装 (less/sass/stylus)

本项目安装sass预处理器

Sass文档: <https://www.sass.hk/docs/>

安装:

```
$ npm install node-sass --save-dev
$ npm install sass-loader --save-dev
```

## 4、axios配置

文档: <https://www.kancloud.cn/yunye/axios/234845>

### 1)、npm安装:

```
$ npm install axios
```

### 2)、请求拦截配置

于src目录下创建axios文件夹, 并于其中创建index.js文件, 内容如下 (PC端配置了element-ui的一个加载组件):

```
import axios from 'axios'
import { Notification, Loading } from 'element-ui'

// 根据环境设置请求baseUrl
axios.defaults.baseUrl = process.env.API_ROOT
// 请求超时时间
axios.defaults.timeout = 6000
// post请求头
axios.defaults.headers.post['Content-Type'] = 'application/json;charset=utf-8'

let loading
// 请求拦截器
axios.interceptors.request.use(
  config => {
    // 加载提示
    loading = Loading.service({
      lock: true,
      text: '拼命加载中',
      spinner: 'el-icon-loading',
      background: 'rgba(0, 0, 0, 0.7)'
      // background: 'rgba(255, 255, 255, 0)'
    })
  },
  error => Promise.reject(error)
```



```
    })
    return config
  },
  error => {
    return Promise.reject(error)
  }
})

// 响应拦截器
axios.interceptors.response.use(
  response => {
    // 关闭加载提示
    loading.close()
    return response
  },
  error => {
    // 关闭加载提示
    loading.close()
    if (error && error.response) {
      switch (error.response.status) {
        case 400:
          Notification.error({
            message: '请求错误'
          })
          break

        case 403:
          Notification.error({
            message: '拒绝访问'
          })
          break

        case 404:
          Notification.error({
            message: '请求地址出错'
          })
          break

        case 408:
          Notification.error({
            message: '请求超时'
          })
          break

        case 500:
          Notification.error({
            message: '服务出错'
          })
          break

        case 501:
          Notification.error({
            message: '网络未实现'
          })
          break

        case 502:
          Notification.error({
            message: '网络错误'
          })
          break

        case 503:
          Notification.error({
            message: '服务不可用'
          })
          break

        case 504:
          Notification.error({
            message: '网络超时'
          })
          break

        default:
      }
    }
    return Promise.reject(error)
  }
)
} else if (error.message === 'Network Error') {
```

```
Notification.error({
  message: '网络错误'
})
} else if (error.code === 'ECONNABORTED' && error.message.indexOf('timeout'))
Notification.error({
  message: '请求超时，请重试'
})
}

return Promise.reject(error)
}
)

export default axios
```

### 3)、API请求管理

于src目录下新建api文件夹，文件夹中根据API类型新建对应的js文件模块以便管理。

例如用户信息请求API新建userApi.js文件

文件内容格式如下：

```
import request from '@/axios'

/**
 * 用户登录
 * @param {Object} data 用户账号信息对象
 */
export function login(data) {
  return request({
    url: '/user/login',
    method: 'post',
    data
  })
}

/**
 * 获取用户信息
 * @param {String} token token
 */
export function getInfo(token) {
  return request({
    url: '/user/info',
    method: 'get',
    params: { token }
  })
}
```

### 4)、js中调用

```
//script标签内进行导入
import * as userApi from '@/api/userApi'

//使用：
userApi.login({username: '张三', password: '123456'})
  .then(res => {
    const { data } = res
    console.log(data)
  })
```

## 5、Vue Router配置

### 1)、router/index.js配置文件修改



文档: <https://router.vuejs.org/zh/installation.html>

```
|-- src
...
|   |-- layout
|   |   |-- index.vue
...
|   |-- router
|   |   |-- index.js
|   |   |-- modules
|   |       |-- userCenter.js
```

```
import Vue from 'vue'
import Router from 'vue-router'

import Layout from '@/layout'

/* Router Modules */
import userRouter from './modules/userCenter'

Vue.use(Router)

const router = new Router({
  // 路由跳转后重新定位到顶部
  scrollBehavior: () => ({ y: 0 }),
  routes: [
    userRouter,
    {
      path: '/',
      name: 'Layout',
      component: Layout,
      meta: {title: '主页'}
    },
    { path: '*', component: () => import('@/components/404') }
  ]
})

export default router
```

可以根据项目需要可配置导航守卫、路由模块等相关配置

## 2)、全局layout

在src目录下新建layout文件夹，其中新建index.vue文件

```
<template>
  <div>
    Layout
    <router-view/>
  </div>
</template>

<script>
export default {
  name: 'Layout'
}
</script>

<style lang="scss" scoped>

</style>
```

## 6、Vuex配置

文档: <https://vuex.vuejs.org/zh/>

### 1)、Vuex安装

```
$ npm install vuex --save
```

### 2)、vuex-persistedstate安装

文档: <https://github.com/robinvdvleuten/vuex-persistedstate>

vuex-persistedstate是一个将vuex持久化的插件

```
$ npm install --save vuex-persistedstate
```

### 3)、配置

于src目录下新建store文件夹，其下新建index.js、mutation-types.js文件和modules文件夹

```
-- store
| |-- index.js
| |-- modules
| |   |-- user.js
|   |-- mutation-types.js
```

将 store 分割成模块 (module)，每个模块拥有自己的 state、mutation、action、getter、甚至是嵌套子模块——从上至下进行同样方式的分割。

使用常量替代 mutation 事件类型 (使用 ES2015 风格的计算属性命名功能来使用一个常量作为函数名)，把这些常量放在单独的文件中可以让你的代码合作者对整个 app 包含的 mutation 一目了然。

新建mutation-types.js文件存放常量，例如：

```
// mutation-types.js
// user modules
export const SET_USER_NAME = 'SET_USER_NAME'
```

```
// user.js
import * as types from '../mutation-types'

const state = {
  userName: ''
}

const mutations = {
  [types.SET_USER_NAME](state, str) {
    state.userName = str
  }
}

const actions = {
  saveUserName({ commit }, str) {
    commit(types.SET_USER_NAME, str)
  }
}

export default {
  namespaced: true,
  state,
  mutations,
  actions
}
```

index.js配置如下：

vuex-persistedstate插件仅将需要持久化的状态值进行持久化。



```
import Vue from 'vue'
import Vuex from 'vuex'
import CreatePersistedState from 'vuex-persistedstate'

Vue.use(Vuex)

// webpack v4.35.2 依赖管理
// https://webpack.js.org/guides/dependency-management/#requirecontext
const modulesFiles = require.context('./modules', false, /\.js$/)

// 自动请求modules目录下的模块文件
const modules = modulesFiles.keys().reduce((modules, modulePath) => {
  // 模块名为modules目录下的js文件名
  const moduleName = modulePath.replace(/^\.\/(.*)\.\w+$/, '$1')

  // value 为请求到的模块文件内容 {default:{state:{...},mutations:{...},actions:{...}}
  const value = modulesFiles(modulePath)
  modules[moduleName] = value.default
  return modules
}, {})

const vuexPersisted = new CreatePersistedState({
  key: 'VuexPersisted',
  storage: window.localStorage,
  reducer: state => ({
    user: {
      userName: state.user.userName
    }
  })
})

const store = new Vuex.Store({
  modules,
  plugins: [vuexPersisted]
})


export default store
```



#### 4)、组件中使用

使用方式就不一一列举了，以下仅列举其中一种，其他的可以去看官方文档。


访问state值时，使用对象展开运算符将此对象混入到局部计算属性中。




```
import { mapState } from 'vuex'

export default {
  computed: {
    ...mapState('user', ['userName'])
  },

  mounted() {
    console.log(this.userName)
  }
}
```



访问actions，使用对象展开运算符将此对象混入到局部 methods 中。



```
import { mapActions } from 'vuex'

export default {
  methods: {
    ...mapActions('user', ['saveUserName']),
  }
}
```

```
// 登录时存储用户名
login() {
  this.saveUserName('80xxxxx06@qq.com')
}
}
```

## 7、自定义样式/主题配置

于src目录下新建styles文件夹

```
|-- src
...
|   |-- styles
|   |   |-- border.css           #app一像素边框解决
|   |   |-- element-ui.scss     #自定义的element-ui组件样式
|   |   |-- element-variables.scss #通过element-ui样式变量修改通用主题
|   |   |-- index.scss          #全局通用样式引入/定义
|   |   |-- mixin.scss          #自定义的scss混合指令
|   |   |-- reset.css           #css样式重置文件
|   |   |-- transition.scss     #自定义过渡样式
|   |   |-- variables.scss      #自定义样式变量
```

在main.js中引入

```
import '@/styles/index.scss' // 全局 css 样式
```

## 8、打包相关配置修改

### 1)、config/index.js文件修改

将文件中build对象中assetsPublicPath值改为'../'

```
assetsPublicPath: '../',
```

### 2)、build/utils.js文件修改

```
if (options.extract) {
  return ExtractTextPlugin.extract({
    use: loaders,
    publicPath: '../..',
    fallback: 'vue-style-loader'
  })
} else {
  return ['vue-style-loader'].concat(loaders)
}
```

以上配置中添加了 publicPath 配置 '../..'

此配置解决打包后图片读取路径错误问题

## 9、多环境打包配置

一)、配置方式一:

<https://www.cnblogs.com/VoiceOfDreams/p/11052447.html>

二)、配置方式二:

注: 由于我之前写文档一的配置方式稍显繁琐, 现将文档一的配置进行简化。此次文档就只配置开发、测试、生产三个环境, 添加其他环境的配置方式一样。

### 1、安装依赖: cross-env

```
$ npm i --save-dev cross-env
```

## 2、修改项目 `package.json` 文件

```
"scripts": {
  "dev": "webpack-dev-server --inline --progress --config build/webpack.dev.conf.js",
  "start": "npm run dev",
  "lint": "eslint --fix --ext .js,.vue src",
  "build:dev": "cross-env NODE_ENV=production ENV_CONFIG=dev node build/build.js",
  "build:test": "cross-env NODE_ENV=production ENV_CONFIG=test node build/build.js",
  "build:prod": "cross-env NODE_ENV=production ENV_CONFIG=prod node build/build.js"
},
```

## 3、修改项目 `config` 配置文件

### 目录结构

```
|-- config
|   |-- dev.env.js
|   |-- index.js
|   |-- prod.env.js
|   `-- test.env.js
```

### 1)、修改 `dev.env.js` 文件

```
'use strict'
const merge = require('webpack-merge')
const prodEnv = require('./prod.env')

const env = process.env.NODE_ENV

module.exports = merge(prodEnv, {
  NODE_ENV: '"development"',
  ENV_CONFIG: '"dev"',
  API_ROOT: env === 'production' ? '"http://(线上开发环境请求地址)"' : '"apis"'
})
```

### 2)、添加 `test.env.js` 文件

```
'use strict'
const merge = require('webpack-merge')
const prodEnv = require('./prod.env')

module.exports = merge(prodEnv, {
  NODE_ENV: '"testing"',
  ENV_CONFIG: '"test"',
  API_ROOT: '"http://(线上测试环境请求地址)"'
})
```

### 3)、修改 `prod.env.js` 文件

```
'use strict'
module.exports = {
  NODE_ENV: '"production"',
  ENV_CONFIG: '"prod"',
  API_ROOT: '"http://(线上生产环境请求地址)"'
}
```

### 4)、修改 `index.js` 文件

解决本地开发启动后浏览器跨域问题，在此处配置服务代理。

dev对象参数下修改如下配置：

```

proxyTable: {
  '/apis': {
    target: 'http://(本地开发环境请求地址)',
    changeOrigin: true, // 是否允许跨域
    pathRewrite: {
      '^/apis': '' // 重写
    }
  },
},
},

```

```

module.exports = {
  dev: {
    // Paths
    assetsSubDirectory: 'static',
    assetsPublicPath: '/',
    proxyTable: {
      '/apis': {
        target: 'http://(本地开发环境请求地址)',
        changeOrigin: true, // 是否允许跨域
        pathRewrite: {
          '^/apis': '' // 重写
        }
      },
    },
  },
  // Various Dev Server settings
  host: 'localhost', // can be overwritten by process.env.HOST

```

build 对象参数下添加如下参数

```

devEnv: require('./dev.env'),
testEnv: require('./test.env'),
prodEnv: require('./prod.env'),

```

```

build: {
  devEnv: require('./dev.env'),
  testEnv: require('./test.env'),
  prodEnv: require('./prod.env'),

  // Template for index.html
  index: path.resolve(__dirname, '../dist/index.html'),

  // Paths
  assetsRoot: path.resolve(__dirname, '../dist'),
  assetsSubDirectory: 'static',
  assetsPublicPath: './',

```

#### 4、修改 build.js 文件

```

// process.env.NODE_ENV = 'production' // 将此行代码注释

// const spinner = ora('building for production...')
const spinner = ora('building for ' + process.env.NODE_ENV + ' of ' + process.env

```

```

1 // process.env.NODE_ENV = 'production'
2
3 You've a day ago + 011030311
4
5 const ora = require('ora')
6 const rm = require('rimraf')
7 const path = require('path')
8 const chalk = require('chalk')
9 const webpack = require('webpack')
10 const config = require('../config')
11 const webpackConfig = require('./webpack.prod.conf')
12
13
14 // const spinner = ora('building for production...')
15 const spinner = ora('building for ' + process.env.NODE_ENV + ' of ' + process.env.ENV_CONFIG + ' production...')
16 spinner.start()

```

#### 5、修改 webpack.prod.conf.js 文件

原代码

```
const env = require('../config/prod.env')
```

修改后

```
const env = config.build[process.env.ENV_CONFIG+'Env']
```

```
13
14 // const env = require('../config/prod.env')
15 const env = config.build[process.env.ENV_CONFIG+'Env']
16
```

### 三)、项目中 HTTP 请求设置

配置文档一中是在 API 文件中获取配置请求地址,也可以在 axios 配置中设置统一的默认请求地址

详见 axios 配置中, axios 文件夹下的 index.js 配置文件

```
// 根据环境设置请求baseUrl
axios.defaults.baseURL = process.env.API_ROOT
```

如果您觉得阅读本文对您有帮助,请点一下“推荐”按钮  
如果标题被标记为转载,转载请注明原作者地址  
版权声明,转载请注明出处: <https://www.cnblogs.com/VoiceOfDreams>

分类: [前端](#)标签: [前端](#)

好文要顶

关注我

收藏该文



墨道

[关注 - 1](#)[粉丝 - 15](#)[+加关注](#)

1

推荐

0

反对

« 上一篇: [Cordova自定义插件开发](#)» 下一篇: [exp/imp 数据库数据导出/导入](#)posted @ 2019-09-01 17:50 墨道 阅读(1595) 评论(0) [编辑](#) [收藏](#)[刷新评论](#) [刷新页面](#) [返回顶部](#)登录后才能发表评论,立即 [登录](#) 或 [注册](#), [访问](#) 网站首页[写给园友们的一封求助信](#)

相关博文:

- [vuecli2引入iconfont](#)
- [\[Vue专题\]对比vue-cli2.x和vue-cli3.x的搭建](#)
- [vuecli2.x配置多环境打包](#)
- [Vue](#)
- [vue](#)
- » [更多推荐...](#)

最新 IT 新闻:

- 周鸿祎称年轻人不要太敏感:以后受到更大的打击 容易受不了
- 无需用眼 大脑直接成像!失明的人也能重新“看见”
- 一万人研发!华为全球首发车规级激光雷达 150米探测距离
- 美股或暴跌50%!对冲基金对新冠变异病毒发出预警



· [长征八号运载火箭首次飞行试验取得圆满成功](#)  
» [更多新闻...](#)