# 2-1 项目目录升级-一个正式项目的目录结构

配置相关的可都放文件夹 build 下

Build的下，建webpack.config.base.js，放置webpack共同的配置，其他开发环境，正式环境等，依赖于base.js

使用webpack-merge扩展config



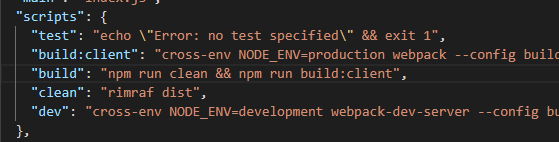


## 2-2 vue-loader配置

每次打包前删除dist文件夹

先安装

配置packjson



意思是，运行npm run build后，先运行npm run clean，删除dist文件夹，然后再运行npm run build:client，重新生成dist文件夹

## 2-3 css module 配置

## 2-4安装使用eslint和editorconfig以及precommit

安装eslint

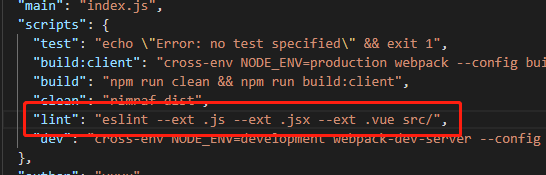




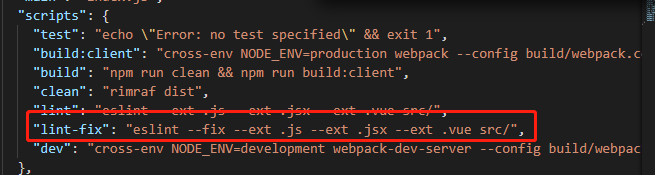
vueCoreTodo>npm i eslint eslint-config-standard eslint-plugin-standard eslint-plugin-promise eslint-plugin-import eslint-plugin-node.

新建文件 .eslintrc



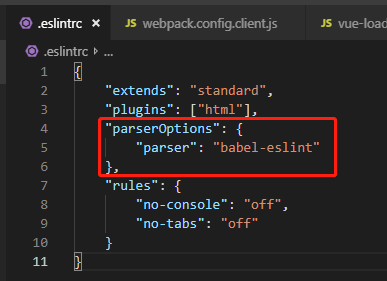


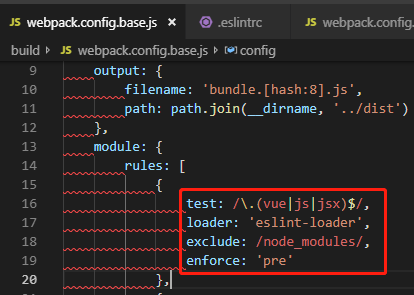
Npm run lint 很多报错，eslint自动修复



写代码时eslint自动检查

先安装 

配置eslintrc 

配置webpack-config.js

.editorconfig 规范编辑器配置，不同的编辑器中，使用相同的配置

Git 在提交代码的时候，先使用precommit钩子，运行npm run lint-fix，如果没通过，就没法提交，这样就不会将有问题的代码提交到远程

先安装 npm i husky -D

配置package.json

"scripts": {

"test": "echo \"Error: no test specified\" && exit 1",

"build:client": "cross-env NODE\_ENV=production webpack --config build/webpack.config.client.js",

"build": "npm run clean && npm run build:client",

"clean": "rimraf dist",

"lint": "eslint --ext .js --ext .jsx --ext .vue src/",

"lint-fix": "eslint --fix --ext .js --ext .jsx --ext .vue src/",

"precommit": "npm run lint-fix"

},

**打开项目先git init一下，再进行其他操作**

# 3 VUE核心知识

## VUE实例

Ex：

new Vue({

el: '#root',

template: '<div>aaaa</div>'

})

VUE实例的创建和作用

挂载的另一个方式

const app = new Vue({

// el: '#root',

template: '<div>aaaa</div>'

})

app.$mount('#root')

// console.log(app.$data)

// console.log(app.$props)

// console.log(app.$el)

// console.log(app.$options)

// app.$options.render = (h) => {

// return h('div', {}, 'new render function')

// }

// console.log(app.$root === app)

// console.log(app.$children)

// console.log(app.$slots)

// console.log(app.$scopedSlots)

// console.log(app.$refs)

// console.log(app.$isServer)

### [v-html](https://cn.vuejs.org/v2/api/#v-html)

**预期**：string

**详细**：

更新元素的 innerHTML 。**注意：内容按普通 HTML 插入 - 不会作为 Vue 模板进行编译** 。如果试图使用 v-html 组合模板，可以重新考虑是否通过使用组件来替代。

在网站上动态渲染任意 HTML 是非常危险的，因为容易导致 [**XSS 攻击**](https://en.wikipedia.org/wiki/Cross-site_scripting)。只在可信内容上使用 v-html，**永不**用在用户提交的内容上。

在[**单文件组件**](https://cn.vuejs.org/v2/guide/single-file-components.html)里，scoped 的样式不会应用在 v-html 内部，因为那部分 HTML 没有被 Vue 的模板编译器处理。如果你希望针对 v-html 的内容设置带作用域的 CSS，你可以替换为 [**CSS Modules**](https://vue-loader.vuejs.org/en/features/css-modules.html) 或用一个额外的全局 <style> 元素手动设置类似 BEM 的作用域策略。

**示例**：

<div v-html="html"></div>

data: {

html: ‘<span>123</span>’

}

然后页面就成了

<div>

<span>123</span>

</div>

动态绑定class

template: `

<div :class="[{ active: isActive }]"

:style="[styles, styles2]">

<p>123</p>

</div>

`,

data: {

isActive: false,

styles: {

color: 'red',

appearance: 'none'

},

styles2: {

color: 'black'

}

},

v-show 给DOM加上属性 display: none

v-if 直接移除DOM 性能增加

安装路由



配置路由

import Todo from '../views/todo/todo.vue'

import Login from '../views/login/logni.vue'

export default = [

{

path: '/',

component: Todo

},

{

path: '/login',

component: Login

}

]

配置入口

import Router from 'vue-router'

import routes from './routes'

export default () => {

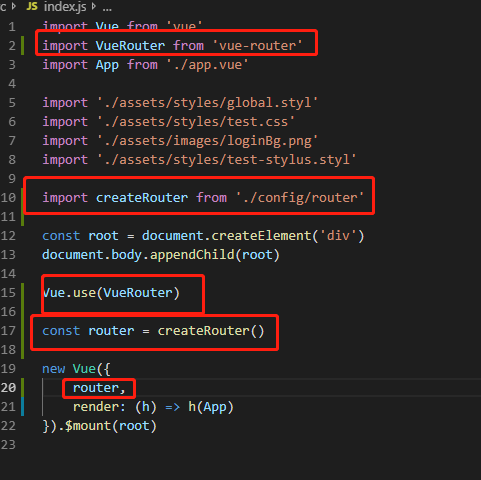
const router = new Router({

routes

})

}

配置index.js



export default () => {

return new Router({

routes,

// base: '/base/', // 路由前都会加上这个base

linkActiveClass: 'active-line', // router-link 样式的变化

linkExactActiveClass: 'exact-active-link', // router-link 样式的变化

scrollBehavior (to, from, savedPosition) {

// 页面进行路由跳转时，页面是否滚动

// to 去到的路由

// from 从哪去的路由

// savedPosition 如果到过这个路由，自动保存之前滚动到的位置

if (savedPosition) {

return savedPosition

} else {

return { x: 0, y: 0 }

}

}

// mode: 'history' // 去掉8000后面的 # 有问题，不知道咋整

})

}

默认路由

{

path: '/',

redirect: '/app'

}

路由配置

{

path: '/app/:id',

// props: true,

// props: {

// id: '123'

// },

// props: (route) => ({ id: route.query.b }),

component: Todo,

name: 'app',

meta: {

title: 'this is app',

description: 'aaabbb'

}

// children: [

// // 嵌套路由

// // 要在父组件 Todo 内加上 router-vie 标签，children才会显示

// // 路径为 /app/test

// {

// path: 'test',

// component: Login

// }

// ]

},

<router-link to="/app/123">app</router-link>

路由可命名

<router-view name="a"/>

对应修改

path: '/app',

components: {

default: Todo,

a: Login

},

路由的导航守卫

// index.js 路由跳转前触发,执行完之后才会跳转

router.beforeEach((to, from, next) => {

// 可做些验证，比如没登陆信息就不允许访问其他页面

// 直接跳回登陆页面

next('/login')

// next()

})

// 跟beforeEach差不多用法

router.beforeResolve((to, from, next) => {

next()

})

// 路由跳转之后才触发

router.beforeResolve((to, from) => {

})

4-5 Vuex之集成

安装

npm I vuex -S

新建文件夹 stores

新建文件 store

import Vuex from 'vuex'

import defaultState from './state/state'

import mutations from './mutations/mutations'

import getters from './getters/getters'

import actions from './actions/actions'

export default () => {

const store = new Vuex.Store({

state: defaultState,

mutations,

getters,

actions

})

修改index.js

import Vue from 'vue'

import Vuex from 'vuex'

Vue.use(Vuex)

new Vue({

router,

store,

render: (h) => h(App)

}).$mount('#root')

State: 类似声明变量

export default {

firstName: 'Jokcy',

lastName: 'Lou'

}

getters：拿出VUEX里的变量，可组合一下

（可以认为是 store 的计算属性）

export default {

fullName (state) {

return `${state.firstName} 123 ${state.lastName}`

}

}

...*mapGetters*({  
 **'vsHeight'**: **'global/visualHeight'**}),

未定稿的语法，需安装 babel-preset-stage-1

npm I babel-preset-stage-1 -D

mutation：更改 Vuex 的 store 中的状态的唯一方法是提交 mutation。

action：Action 类似于 mutation，不同在于：

Action 提交的是 mutation，而不是直接变更状态。

Action 可以包含任意异步操作。