# 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



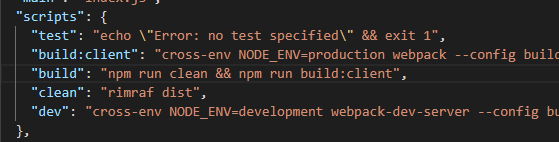


## 2-2 vue-loader配置

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

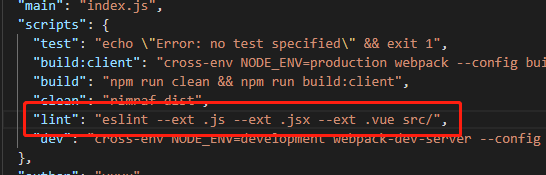




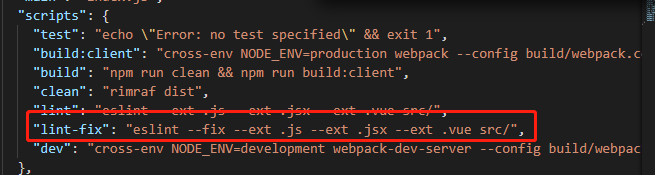
vueCoreTodo>npm i eslint eslint-config-standard eslint-plugin-standard eslint-plugin-promise eslint-plugin-import eslint-plugin-node.

新建文件 .eslintrc



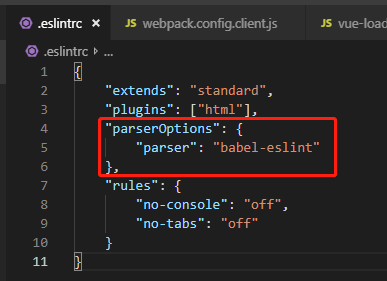


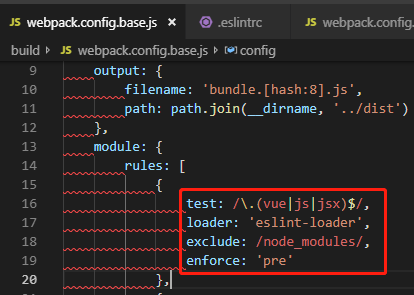
Npm run lint 很多报错，eslint自动修复



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/" \l "v-html" \o "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" \t "https://cn.vuejs.org/v2/api/_blank)**。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" \t "https://cn.vuejs.org/v2/api/_blank)** 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

