重点难点不是业务开发、性能要求并不是很高，不会做在线ps一样的应用

最重要的是前端工程化的问题。

**前端的价值**

1. 搭建前端工程
2. 网络优化

减少http请求

压缩静态资源文件

使用浏览器强缓存使浏览器的流量变更小、加载速度更快

1. API定制
2. nodejs层

# 2-1 vue-loader+webpack项目配置

新建文件夹，在VS code里打开，终端运行 npm init ，初始化成一个npm项目



一些选项可默认

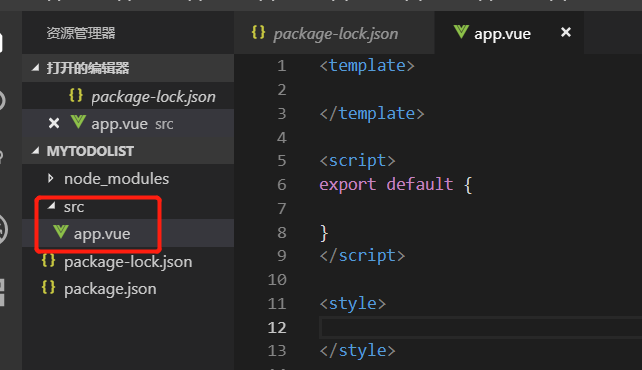
生成package.json文件，然后可安装所需要的包

配置vue 和wepack



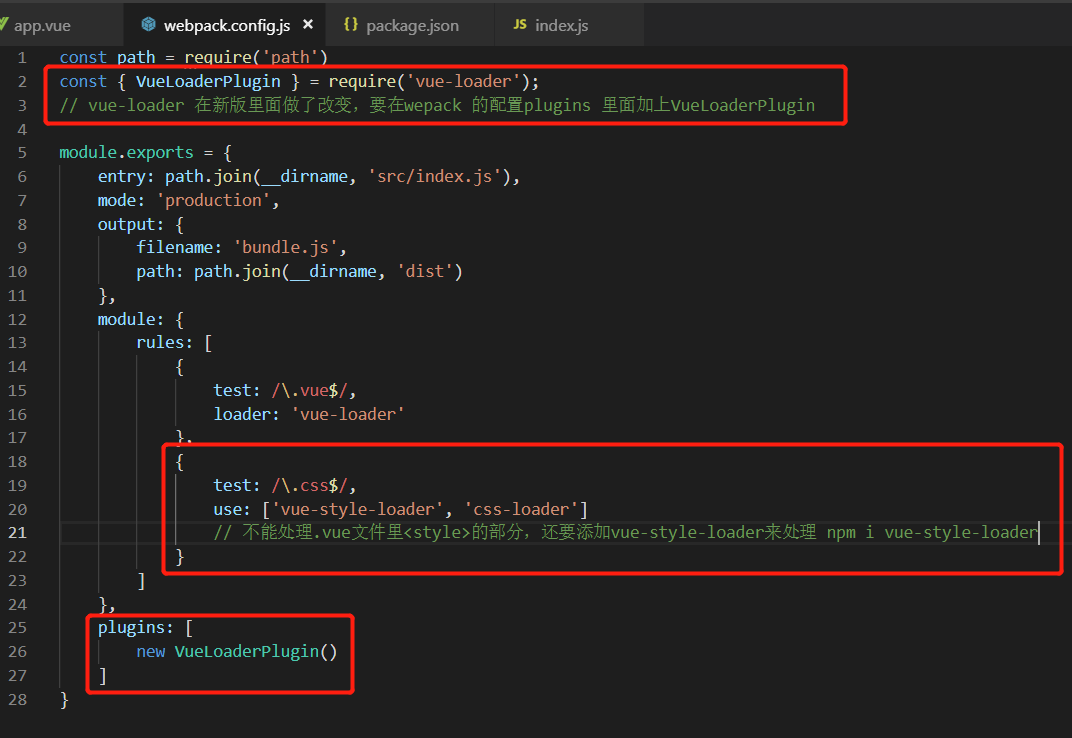
根据警告，继续配置css-loader和 vue-template-compiler

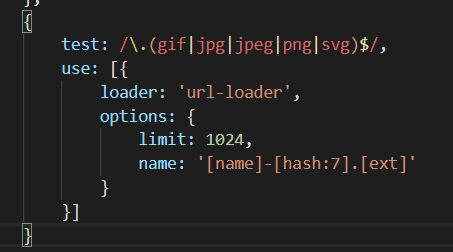


新建文件夹和app.vue

# 2-2webpack配置项目加载各种静态资源及css预处理器

配置webpack.config.js



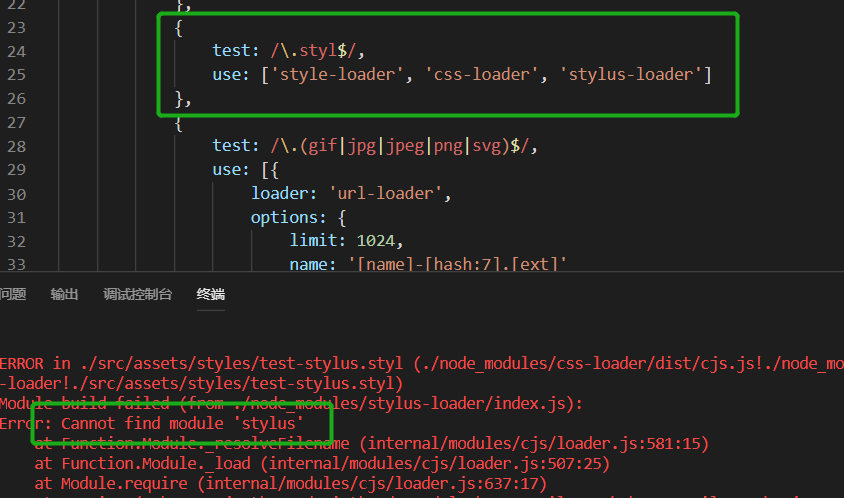


图片处理

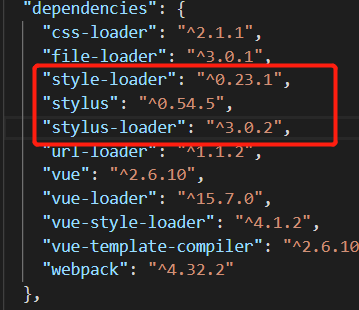
url-loader 依赖于 file-loader

所以要一起安装





这个报错，少装了个依赖

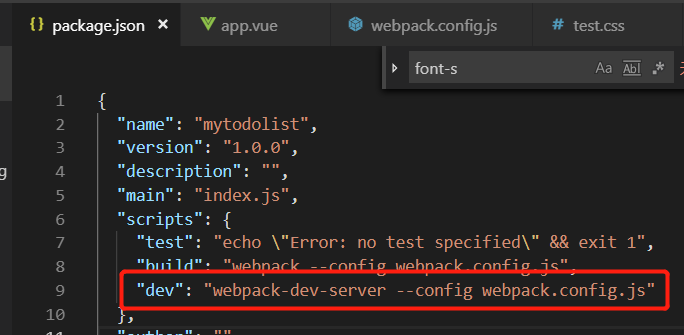


# 2-3 webpack-dev-server的配置和使用

首先安排webpack-dev-server的包

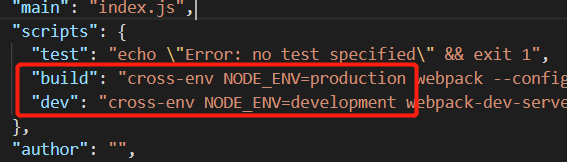


配置



应用于开发环境

安装包 cross-env



安装webpack插件，引入，将打包的文件自动引入html中



然后在webpack.config.js里配置

const HtmlPlugin = require('html-webpack-plugin')

plugins: [

new webpack.DefinePlugin({

'procss.env': {

NODE\_ENV: isDev ? '"devolopment"' : '"production"'

}

}),

new VueLoaderPlugin(),

new HtmlPlugin()

]

'"devolopment"' : '"production"'

里面一定要有双引号

if (isDev) {

config.devtool = '#cheap-module-eval-source-map'

config.devServer = {

port: 8000,

host: '0.0.0.0',

overlay: {

errors: true

},

open: true, // 运行npm run dev 自动打开浏览器

hot: true // 改组件代码，页面只重新渲染组件，不加载整个页面

}

config.plugins.push(

new webpack.HotModuleReplacementPlugin(), // hot: true 需要的插件

new webpack.NoEmitOnErrorsPlugin()

)

}

# 3-1 vue2的核心知识介绍

数据绑定

VUE文件开发方式

render方法

**API重点**

生命周期方法

computed 计算属性

# 3-2 配置VUE的jsx写法一级postcss

安装

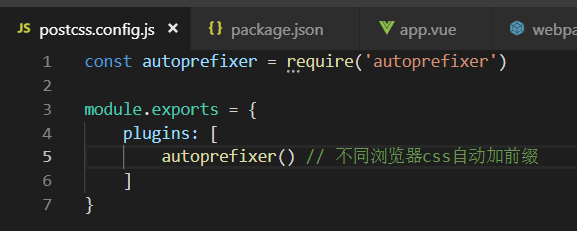


创建文件





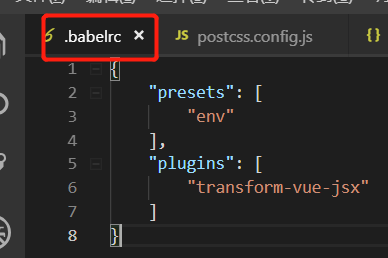
配置



安装



配置



补充安装

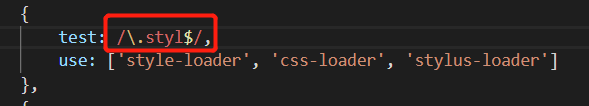


webpack.config.js增加配置



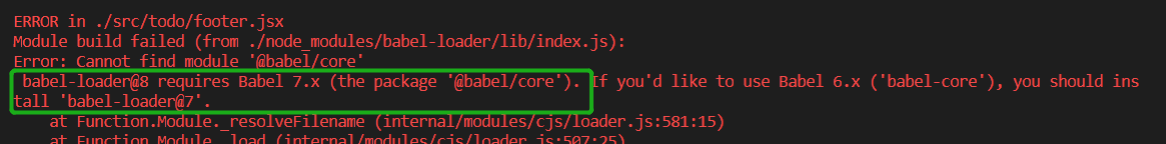
# 3-3 实现头都应用的界面

stylus不可以在.vue文件内使用时，修改webpack.config,js文件



不要$

错误



官方默认babel-loader | babel 对应的版本需要一致: 即babel-loader需要搭配最新版本babel

两种解决方案:

1、回退低版本

npm install -D babel-loader@7 babel-core babel-preset-env

2、更新到最高版本:

npm install -D babel-loader @babel/core @babel/preset-env webpack

参考： <https://www.cnblogs.com/soyxiaobi/p/9554565.html>

