**1.1 项目基本配置**

npm init 初始化一个npm项目  
npm i webpack@3.10.0 vue@2.5.13 vue-loader@13.6.0 安装webpack和vue,使用vue要安装vue-loader  
npm i css-loader@0.28.7 vue-template-compiler@2.5.13 根据终端WARN提示安装css-loader,因为vue-loader依赖css-loader //针对各版本做了详细指定,由于更新过快,避免版本差异性问题,故指定了版本

在app.vue中书写基本的vue结构

首先在webpack.config.js设置入口entry  
声明我们的入口文件index.js

示例中app.vue实际是一个组件,组件是不能直接挂载到我们的html中去,需要在index.js中挂载

webpack.config.js同样设置出口文件bundle.js及存放路径

配置完后,在webpack.config.js中配置build脚本, --config 指定我们的config文件 因为在这里面写,当你调用时才会调用这个项目里面的webpack,否则将会调用全局的webpack,全局webpack和项目中的版本可能存在差异,建议使用这种方式会好一点

**1.2 各种静态资源的加载**

webpack对其他类型的文件处理,可在配置文件中配置rules规则.  
同样根据配置中的处理的loader都要安装.  
npm i style-loader@0.19.1 url-loader@0.6.2 file-loader@1.1.6

stylus的css预处理器 npm i stylus-loader@3.0.1 stylus@0.54.5  
同理其他的像sass,less等其他的预处理器都可以类似的方法去使用

**1.3 webpack-dev-server的配置**

npm i webpack-dev-server@2.9.1  
webpack-dev-server在开发环境中会给我们带来与webpack不一样的效果,用的都是同一个配置文件

"build": "webpack --config webpack.config.js",

"dev": "webpack-dev-server --config webpack.config.js"

同一个配置文件,那么其中必然会根据一个环境变量判断,来判断是开发环境还是正式环境  
NODE\_ENV就是这个环境变量,在linux下 直接NODE\_ENV=production,在windows环境下 需要set NODE\_ENV=production,解决这种跨平台设置的差异性,我们可以安装cross-env@5.1.3  
npm i cross-env@5.1.3

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

"dev": "cross-env NODE\_ENV=development webpack-dev-server --config webpack.config.js"

在webpack.config.js配置好我们的测试环境后,我们还需要引入一个html-webpack-plugin,用于将我们打包好后的js融入到我们的HTml中去  
npm i html-webpack-plugin@2.30.1  
完成webpack.config.js中后,你便可以使用npm run dev见证奇迹的时刻了

**项目实战**

npm i post i postcss-loader@2.0.9 autoprefixer@7.2.3 babel-loader@7.1.2 babel-core@6.26.0  
新建postcss和babel的配置文件 并配置这两个文件

**项目业务**

详见源码

**配置css单独分离打包及其他一些测试和生产环境的区别**

安装extract-text-webpack-plugin  
npm i extract-text-webpack-plugin@3.0.2

**打包类库代码及hash优化**