# 项目打包优化

一、开发模式与发布模式指定不同的打包入口

默认情况下， Vue项目的开发模式与发布模式，共用同一个打包的入口文件（即 src/main.js）。为了将项目的开发过程与发布过程分离，我们可以为两种模式，各自指定打包的入口文件，即：

1、开发模式的入口文件为 src/main-dev.js

2、发布模式的入口文件为 src/main-prod.js

二、configureWebpack 和 chainWebpack

在 vue.config.js 导出的配置对象中，新增 configureWebpack 或 chainWebpack 节点，来自定义 webpack的打包配置。

在这里， configureWebpack 和 chainWebpack 的作用相同，唯一的区别就是它们修改 webpack 配置的方式不同：

1、chainWebpack 通过链式编程的形式，来修改默认的 webpack 配置

2、configureWebpack 通过操作对象的形式，来修改默认的 webpack 配置

三、通过 chainWebpack 自定义打包入口

四、通过 externals 加载外部 CDN 资源

默认情况下，通过 import 语法导入的第三方依赖包，最终会被打包合并到同一个文件中，从而导致打包成功后，单文件体积过大的问题。

为了解决上述问题，可以通过 webpack 的 externals 节点，来配置并加载外部的 CDN 资源。凡是声明在externals 中的第三方依赖包，都不会被打包。

|  |
| --- |
| external是指你可以采用import的方法，来获取这个全局变量的来源。  vue externals 对没有暴露全局变量的库，这个似乎处理不了。  vant不使用cdn 500kb，用了后 170kb |

五、通过 externals 加载外部 CDN 资源

同时，需要在 public/index.html 文件的头部，添加如下的 CDN 资源引用：

https://www.jsdelivr.com/package/npm/vue-waterfall2搜索就可以（这个可能是国外的，访问不了）

国内推荐使用 Staticfile CDN 上的库：

|  |
| --- |
| 你可以通过以下免费 CDN 服务来使用 Vant:  jsdelivr  cdnjs  unpkg |

六、首页内容定制

不同的打包环境下，首页内容可能会有所不同。我们可以通过插件的方式进行定制，插件配置如下：

1、首页内容定制

|  |
| --- |
| <!– 按需渲染页面的标题 -->  <title><%= htmlWebpackPlugin.options.isProd ? '' : 'dev - ' %>电商后台管理系统</title>  <!– 按需加载外部的 CDN 资源 -->  <% if(htmlWebpackPlugin.options.isProd) { %>  <! —通过 externals 加载的外部 CDN 资源-->  <% } %> |

在 public/index.html 首页中，可以根据 isProd 的值，来决定如何渲染页面结构：

七、进行路由懒加载处理

1、安装npm install --save-dev @babel/plugin-syntax-dynamic-import

插件，才能使 Babel 可以正确地解析语法。

|  |
| --- |
| // 切换成路由懒加载，减少对页面的加载速度的影响  // import Home from '../view/home/Home.vue'  // import HomePage from '../view/home/HomePage.vue'  const Home = () => import(/\* webpackChunkName: "Home\_Home" \*/ '../view/home/Home.vue')  const HomePage = () => import(/\* webpackChunkName: "Home\_Home" \*/ '../view/home/HomePage.vue')  黄色部分为分块的名称，同名的就放一起 |

八、在生产环境中去除console.log

1、安装 npm install babel-plugin-transform-remove-console --save-dev

|  |
| --- |
| 在babel.config.js中(语法编译配置)  // 在项目发布阶段才使用到的插件，在开发阶段是不用的  const productPlugins = []  if (process.env.NODE\_ENV === 'production') {  productPlugins.push('transform-remove-console')  }  module.exports = {  presets: [  '@vue/cli-plugin-babel/preset'  ],  plugins: [  ...productPlugins  ]  } |

九、注意

打包出错问题卡了挺久的，vue\_config.js配置错了，都**没有提示**。。。记得记录下来，编译都能通过

vue 项目中出现 Uncaught ReferenceError: exports is not defined=====新的vue-cli默认禁止了commonjs语法，可以添加babel解决

vue/cli 对exports不能识别的配置，暂时不配了，搞不定，一些是局部变量在exterals暂时不知道怎么配置

十、配置示例

|  |
| --- |
| vue.config.js  // postcss.config.js，vant适配rem  const autoprefixer = require('autoprefixer')  const pxtorem = require('postcss-pxtorem')  module.exports = {  //配置 运行和打包时的文件路径  publicPath:process.env.NODE\_ENV==='production'?'./':'/',  // 项目发布配置，通过链式操作  chainWebpack: config => {  config.when(process.env.NODE\_ENV === 'production', config => {  config.entry('app').clear().add('./src/main-prod.js' )  config.set('externals', {  vue: 'Vue',  'vue-router' : 'VueRouter',  axios: 'axios',  vuex: 'Vuex',  // 这两个用的exports，识别不了，暂时不知道配置哪里能处理  // 'vue-waterfall2': 'waterfall',  // fastclick: 'attachFastClick',  'vue-i18n': 'VueI18n',  mockjs: 'Mock',  // 这里一直设置不了全局，暂时无解  // 'better-scroll': 'BScroll',  vant: 'vant'  })  // html根据isProd来判断要不要使用cdn  config.plugin('html').tap(args => {  args[0].isProd = true  return args  })  })  config. when(process.env.NODE\_ENV === 'development' , config => {  config.entry('app').clear().add('./src/main-dev.js' )  // html根据isProd来判断要不要使用cdn  config.plugin('html').tap(args => {  args[0].isProd = false  return args  })  })  // 位置放错了。。。。。  // 把config.plugin写这里了，也不报错  },  css: {  loaderOptions: {  postcss: {  plugins: [  autoprefixer(),  pxtorem({  rootValue: 37.5,  propList: ['\*']  })  ]  },  }  }  } |

