# Vue多页面开发和打包正确处理方法

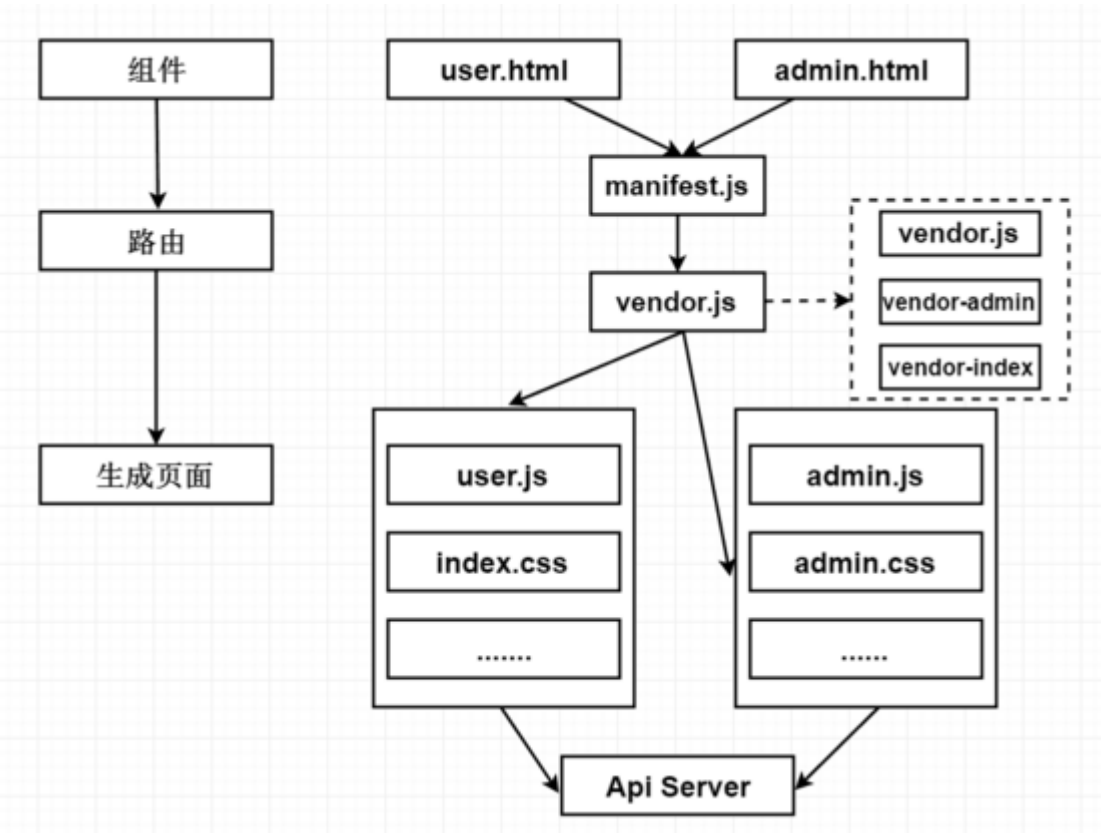
前段时间做项目，技术栈是vue+webpack,主要就是官网首页加后台管理系统 根据当时情况，分析出三种方案

1. 一个项目代码里面嵌两个spa应用(官网和后台系统)
2. 分开两套项目源码
3. 一套项目源码里面就一个spa应用

## 思考

1. 直接否定了一套项目源码里一个spa应用(ui样式会相互覆盖，如果没有代码规范后期比较难维护)
2. 两套源码的话，后台可能开两个端口，然后需要用nginx反向代理可能比较麻烦，而且前端开发也比较麻烦麻烦，毕竟需要维护两个git仓库，两套git上线流程，可能会损耗很多时间。
3. 对自己的技术(盲目)自信,也想尝尝鲜，分析出需求也不算很复杂。选了第一种方案，就是多个单页面应用在一套源码里面

多页面的结构图



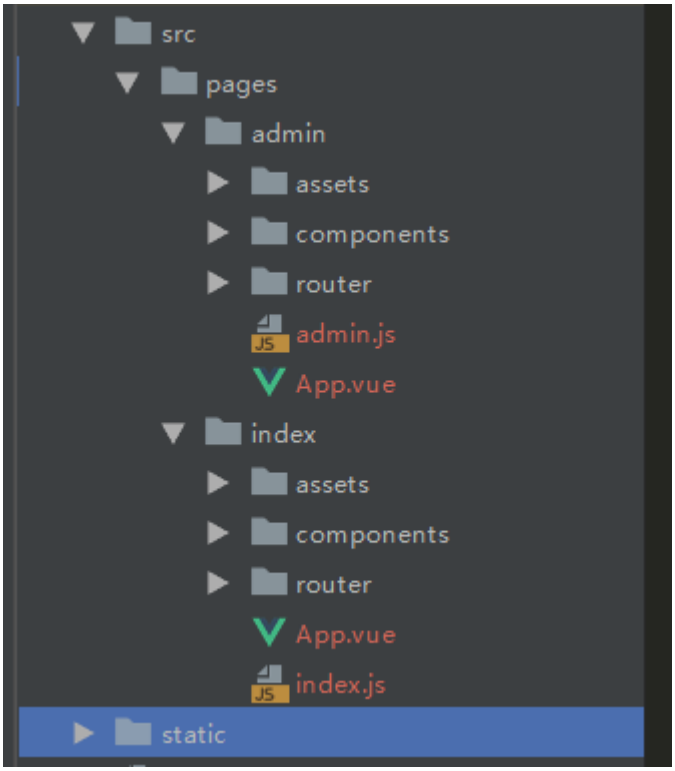

## 下载vue spa模板

```
npm install vue-cli -g
vue init webpack multiple-vue-amazing
```

# 改造多页面应用

```
npm install glob --save-dev
```

修改src文件夹下面的目录结构




在util.js里面加入

```
/* 这里是添加的部分 ---------------------------- 开始 */
// glob是webpack安装时依赖的一个第三方模块，还模块允许你使用 *等符号， 例如lib/*.js就是获取lib文件夹下的所有js后缀名的文件
var glob = require('glob')
// 页面模板
var HtmlWebpackPlugin = require('html-webpack-plugin')
// 取得相应的页面路径，因为之前的配置，所以是src文件夹下的pages文件夹
var PAGE_PATH = path.resolve(__dirname, '../src/pages')
// 用于做相应的merge处理
var merge = require('webpack-merge')
//多入口配置
// 通过glob模块读取pages文件夹下的所有对应文件夹下的js后缀文件，如果该文件存在
// 那么就作为入口处理
exports.entries = function () {
 var entryFiles = glob.sync(PAGE_PATH + '/*/*.js')
 var map = {}
 entryFiles.forEach((filePath) => {
  var filename = filePath.substring(filePath.lastIndexOf('\/') + 1,
filePath.lastIndexOf('.'))
  map[filename] = filePath
 })
 return map
```

```
}
//多页面输出配置
// 与上面的多页面入口配置相同，读取pages文件夹下的对应的html后缀文件，然后放入数组中
exports.htmlPlugin = function () {
 let entryHtml = glob.sync(PAGE_PATH + '/*/*.html')
 let arr = []
 entryHtml.forEach((filePath) => {
  let filename = filePath.substring(filePath.lastIndexOf('\/') + 1,
filePath.lastIndexOf('.'))
  let conf = {
   // 模板来源
   template: filePath,
   // 文件名称
   filename: filename + '.html',
   // 页面模板需要加对应的js脚本，如果不加这行则每个页面都会引入所有的js脚本
   chunks: ['manifest', 'vendor', filename],
   inject: true
  }
  if (process.env.NODE_ENV === 'production') {
   conf = merge(conf, {
    minify: {
     removeComments: true,
     collapseWhitespace: true,
     removeAttributeQuotes: true
    },
    chunksSortMode: 'dependency'
   })
  }
  arr.push(new HtmlWebpackPlugin(conf))
 })
 return arr
}
/* 这里是添加的部分 ---------------------------- 结束 */
webpack.base.conf.js 文件
/* 修改部分 ---------------- 开始 */
 entry: utils.entries(),
 /* 修改部分 ---------------- 结束 */
webpack.dev.conf.js 文件
/* 注释这个区域的文件 ------------- 开始 */
 // new HtmlWebpackPlugin({
 // filename: 'index.html',
 // template: 'index.html',
 // inject: true
 // }),
 /* 注释这个区域的文件 ------------- 结束 */
 new FriendlyErrorsPlugin()
 /* 添加 .concat(utils.htmlPlugin()) ------------------ */
 ].concat(utils.htmlPlugin())
webpack.prod.conf.js 文件
/* 注释这个区域的内容 ---------------------- 开始 */
 // new HtmlWebpackPlugin({
 // filename: config.build.index,
 // template: 'index.html',
```

```
  // inject: true,
  // minify: {
  //  removeComments: true,
  //  collapseWhitespace: true,
  //  removeAttributeQuotes: true
  //  // more options:
  //  // https://github.com/kangax/html-minifier#options-quick-reference
  // },
  // // necessary to consistently work with multiple chunks via CommonsChunkPlugin
  // chunksSortMode: 'dependency'
  // }),
  /* 注释这个区域的内容 ---------------------- 结束 */
  // copy custom static assets
  new CopyWebpackPlugin([
   {
   from: path.resolve(__dirname, '../static'),
   to: config.build.assetsSubDirectory,
   ignore: ['.*']
   }
  ])
  /* 该位置添加 .concat(utils.htmlPlugin()) ------------------- */
  ].concat(utils.htmlPlugin())
```

引入第三方ui库

```
npm install element-ui bootstrap-vue --save
```

分别在不同的页面引入不同的ui

index.js:

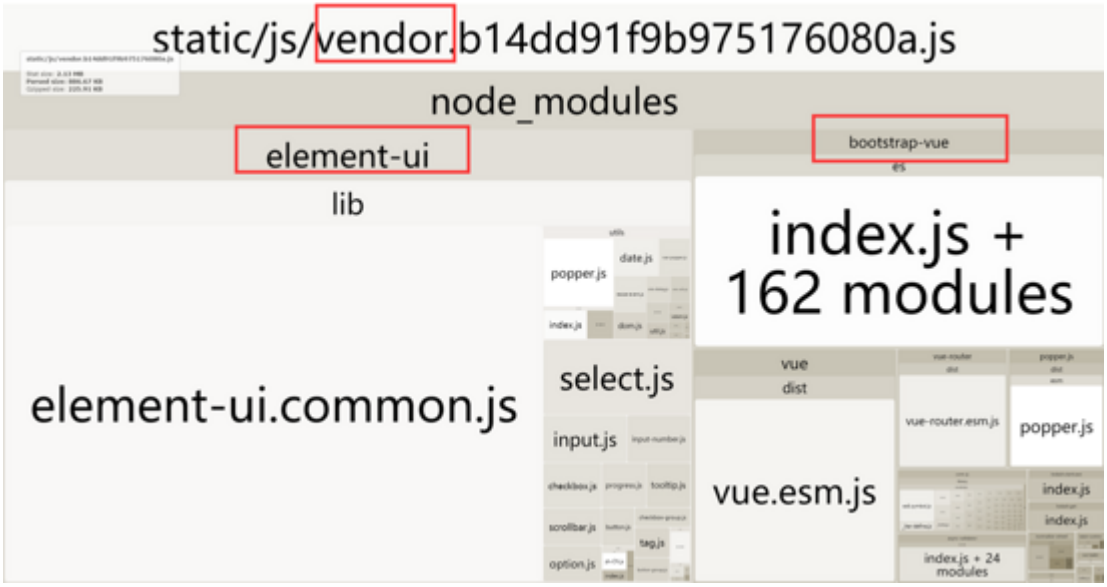
```
import BootstrapVue from 'bootstrap-vue'
Vue.use(BootstrapVue)
```

admin.js:

```
import ElementUI from 'element-ui'
import 'element-ui/lib/theme-chalk/index.css'
Vue.use(ElementUI)
```

上面多页面的配置是参考网上的，而且网上的思路大都很相似，核心就是改多个entry，配置完成了之后，开发的时候也是发现不了问题的，然后大概就开发了一个月，开发完之后对官网进行性能分析时发现，webpack打包的vendor.js网络加载时间特别长，导致首屏的白屏时间非常长，最终通过-webpack-bundle-analyzer分析得到了结论

```
npm run build --report
```




## 遇到问题

你会发现vendor.js包含了index.html和admin.html的共同部分，所以这个vendor包注定会很大很冗余。

## 解决思路

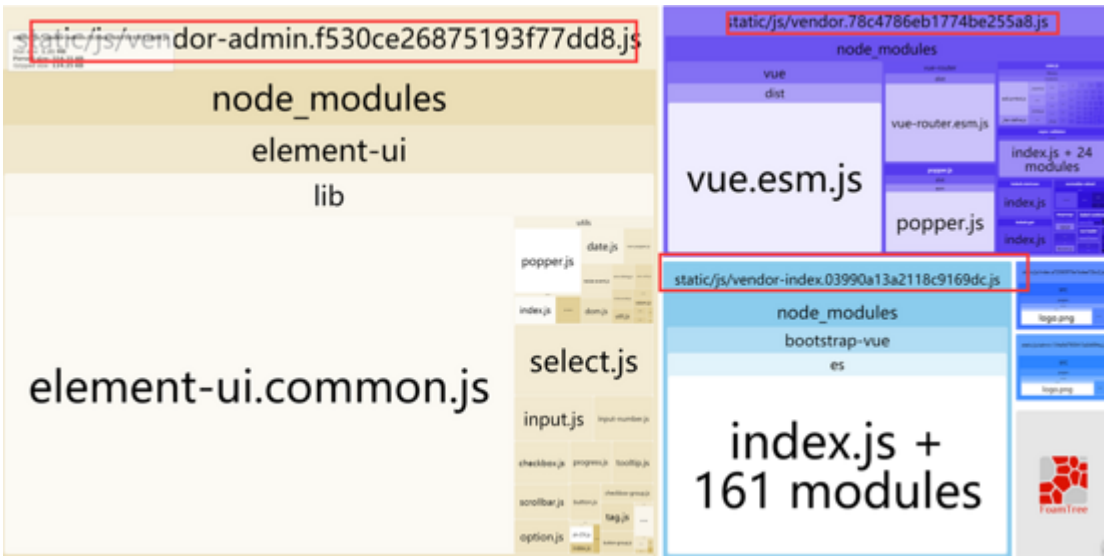
既然是vendor过大引起加载速度慢，那就分离这个vendor就好了。我是这样想的，把各个页面中都使用到的第三方代码提取至vendor.js中，然后各个页面中用到的第三方代码再打包成各自的vendor-x.js，例如现有页面index.html、admin.html，则最终会打包出vendor.js、vendor-index.js、vendor-admin.js。

webpack.prod.conf.js 文件

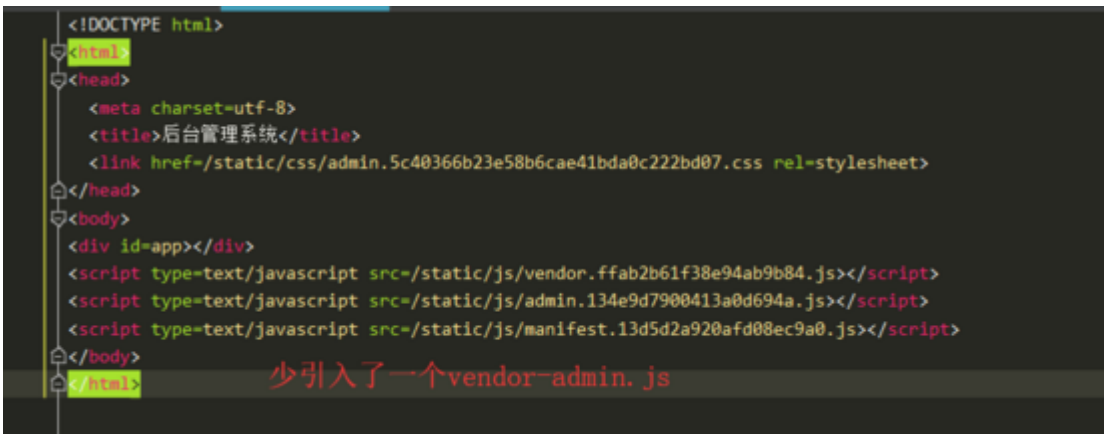
```
new webpack.optimize.CommonsChunkPlugin({
  name: 'vendor-admin',
  chunks: ['vendor'],
  minChunks: function (module, count) {
  return (
   module.resource &&
   /\.js$/.test(module.resource) &&
   module.resource.indexOf(path.join(__dirname, '../node_modules')) === 0 &&
   module.resource.indexOf('element-ui') != -1
  )
  }
 }),
 new webpack.optimize.CommonsChunkPlugin({
  name: 'vendor-index',
  chunks: ['vendor'],
  minChunks: function (module, count) {
  return (
   module.resource &&
   /\.js$/.test(module.resource) &&
   module.resource.indexOf(path.join(__dirname, '../node_modules')) === 0 &&
   module.resource.indexOf('bootstrap-vue') != -1
```

```
    )
    }
  }),
```

再次分析，一切都很ok，vendor.js被分离成了vendor.js、vendor-index、vendor-admin.js。




本来以为解决了CommonsChunkPlugin的分离vendor.js的问题，就可以了，然后打包出来发现index.html和admin.html都少了一个引入(各自对应的那个vendor-xx.js)。




# 解决方案

这个问题其实就是HtmlWebpackPlugin的问题 把原来的 chunksSortMode: 'dependency'改成自定义函数的配置，如下：

util.js文件

```
chunksSortMode: function (chunk1, chunk2) {
   var order1 = chunks.indexOf(chunk1.names[0])
   var order2 = chunks.indexOf(chunk2.names[0])
   return order1 - order2
  },
```

# 最终实现

- 每个页面加载各自的chunk
- 每个页面有不同的参数
- 每个页面能共享公共chunk
- 浏览器缓存，性能更好
- 如果还嫌慢的话，开启gzip

## 感想

大功告成了，虽然配置看起来很简单，不过我当时开发的时候，思考了很久，所以假如你CommonsChunkPlugin和HtmlWebpackPlugin不熟悉或者只会用别人第三方的配置表，估计会踩大坑，比如说,CommonsChunkPlugin不指定chunks，默认是什么？minChunks大多数人只会写一个数值，然而自定义一个函数的写法其实才是最强大的，根据我个人的经验chunks结合minChunks自定义函数的写法，能解决几乎所有CommonsChunkPlugin灵异的事件。

## 总结

以上所述是小编给大家介绍的vue多页面开发和打包正确处理方法，希望对大家有所帮助，如果大家有任何疑问请给我留言，小编会及时回复大家的。在此也非常感谢大家对脚本之家网站的支持！