Webpack静态模块打包器

——详见xlytest项目的webpack.config.js

学习记录：<https://www.webpackjs.com/guides/caching/>

1. 简单属性介绍
   1. entry[入口文件]
      1. 单个入口语法：entry:String/Array<String>
         1. main：入口

const config = {

entry: {

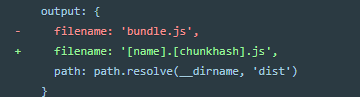
main: './path/to/my/entry/file.js'

}};

* + 1. 对象入口语法：entry:{[entryChunkName:String]:String|ArrayString}  
        这是应用程序中定义入口的最可扩展的方式
       1. app.js应用程序和vendor.js第三方库入口
  1. 缓存

Webpack的缓存：项目打包之后会生成./dist的文件夹，当./dist目录部署到服务器上的时候，在客户端/浏览器中就能够访问此网站及其资源，由于获取数据（前台请求后台接口）是项目里最费时的，所以基本上现在的浏览器都会带缓存，降低网络流量，等待下一次访问的时候会调用缓存数据，加速网站的加载速度，但如果项目重新打包部署了，原来的缓存数据还在，要获取新的数据就不好获取了，而通过配置webpack编译生成的文件能够被客户端缓存，而在文件内容变化后，能够请求到新的文件。

* 1. 输出文件的文件名

output:{filename:’文件名’}  


1. 插件
   1. clean-webpack-plugin:清除文件  
       下载：npm install --save-dev clean-webpack-plugin  
       引入：const CleanWebpackPlugin = require('clean-webpack-plugin')  
       plugin模块引入：new CleanWebpackPlugin(['build'])  
       运行打包命令：webpack --mode development
   2. html-webpack-plugin:创建文件入口  
      <https://www.jianshu.com/p/08a60756ffda>  
       为html文件中引入的外部资源如script、link动态添加每次compile后的hash，防止引用缓存的外部文件问题；可以生成创建html入口文件，比如单页面可以生成一个html文件入口，配置N个html-webpack-plugin可以生成N个页面入口
      1. 属性
         1. title:标题  
            就是我们写html的<head><title>标题<title></head>
         2. filename:html的文件名
         3. templete:模板(模板可以任意选择，需要安装对应的loader)  
            <https://www.webpackjs.com/concepts/loaders/>
         4. Inject:注入选项【这个不是特别懂，大概是选择html的js插入位置】
            1. true:默认值script标签位于html文件的 body 底部
            2. body:同true
            3. head:script放置head标签内
            4. false:不插入js文件
         5. favicon:html页面地址栏的图标  
            <https://blog.csdn.net/woshixuye/article/details/9218107>
         6. minify:对html进行压缩，（压缩成什么样我不知道）属性默认值是false
            1. caseSensitive:是否对大小写敏感
            2. collapseBooleanAttributes:是否简介boolean格式的属性(例disabled=’disabled’)
            3. minifyCSS:是否压缩html里的css（使用clean-css进行压缩）
            4. minifyJS:是否压缩html里的js（使用uglify-js进行压缩）  
               压缩方法：<https://www.jianshu.com/p/dd847647b7e4>
            5. preventAttributesEscaping:防止属性值转义
            6. …………详见  
               <https://blog.csdn.net/weixin_30640291/article/details/95483578>
         7. hash:给生成的js文件一个hash值

<script type=text/javascript src=bundle.js?22b9692e22e7be37b57e></script>

bundle.js 文件后跟的一串 hash 值就是此次 webpack 编译对应的 hash 值

* + - 1. cache:默认是true的，表示内容变化的时候生成一个新的文件
      2. showErrors:显示错误信息，这个是在webpack编译出现错误的时候，会将错误信息包裹在pre标签内，默认值为true，开启之后方便错误定位
      3. chunks:选择入口文件用的，我们在entry里面定义多个入口文件之后，webpack编译之后会生成多个打包的文件，chunks就是选择要使用的js文件的

entry: {

index: path.resolve(\_\_dirname, './src/index.js'),

devor: path.resolve(\_\_dirname, './src/devor.js'),

main: path.resolve(\_\_dirname, './src/main.js')}

plugins: [

new httpWebpackPlugin({

chunks: ['index','main']

})]

那么编译后：而如果没有指定 chunks 选项，默认会全部引用。

<script type=text/javascript src="index.js"></script> <script type=text/javascript src="main.js"></script>

* + - 1. excludeChunks:这个和Chunks的作用相反，排除js的

plugins: [

new httpWebpackPlugin({

excludeChunks: ['devor.js']//和的等等效

})]

编译之后

<script type=text/javascript src="index.js"></script>

<script type=text/javascript src="main.js"></script>

* + 1. 新版本可能遇到的问题  
       throw new Error('Cyclic dependency' + nodeRep)  
       原因：循环引用依赖  
       解决办法：  
        使用Alpha的版本npm i --save-dev html-webpack-plugin@next或者加入chunksSortMode: 'none'就可以了，但是chunksSortMode会改变chunks的加载顺序，如果设置为none，你的 chunk 加载在页面中加载的顺序就不能够保证了，可能会出现样式被覆盖的情况。【注释：如果在自定义的css里面修改了第三方的element-ui的样式，再通过加载的顺序来覆盖它，如果设置为none，则会先加载自定义的css再加载element-ui，后者将前者覆盖，除非你使用important或者其它 css 权重的方式覆盖它，但这明显是不太合理的。】
  1. [CommonsChunkPlugin:去除重复模块](https://www.webpackjs.com/guides/caching/" \l "%E6%8F%90%E5%8F%96%E6%A8%A1%E6%9D%BF-extracting-boilerplate-)  
      我理解的是两个js文件里面引入了相同的js，在打包的时候如果不去除，那么打包了的两个js文件里就包含了相同的js，为了可以按需加载或并行加载这些文件，可以用[代码分离](https://www.webpackjs.com/guides/code-splitting/)来获取更小的 bundle，减少加载时间，所以用这个来去重
     1. 