<https://webpack.js.org/configuration/>

const path = require('path');
模块module导出exports
module.exports = {

开发环境的打包还是线上环境的打包
[mode](https://webpack.js.org/configuration/mode): "production", // "production" | "development" | "none"
// Chosen mode tells webpack to use its built-in optimizations accordingly.
// 选择模式告诉webpack相应地使用其内置优化。默认 none
[entry](https://webpack.js.org/configuration/entry-context/#entry): "./app/entry", // string | object | array
// defaults to ./src

//默认为./src
// Here the application starts executing

//在这里应用开始执行
// and webpack starts bundling

//和webpack开始捆绑
[output](https://webpack.js.org/configuration/output): {
// options related to how webpack emits results//与webpack如何输出结果有关的选项
path: path.resolve(\_\_dirname, "dist"), // string
// the target directory for all output files//所有输出文件的目标目录
// must be an absolute path (use the Node.js path module)//必须是绝对路径（使用Node.js路径模块）
[filename](https://webpack.js.org/configuration/output/#outputfilename): "bundle.js", // string
// the filename template for entry chunks//条目块的文件名模板
[publicPath](https://webpack.js.org/configuration/output/#outputpublicpath): "/assets/", // string
// the url to the output directory resolved relative to the HTML page//相对于HTML页面解析的输出目录的url
library: "MyLibrary", // string,
// the name of the exported library//导出库的名称
[libraryTarget](https://webpack.js.org/configuration/output/#outputlibrarytarget): "umd", // universal module definition通用模块定义
// the type of the exported library//导出库的类型
/\* Advanced output configuration (click to show) \*// \*高级输出配置（单击以显示）\* /
/\* Expert output configuration (on own risk) \*// \*专家输出配置（风险自负）\* /
},

模块
module: {
// configuration regarding modules有关模块的配置
rules: [
// rules for modules (configure loaders, parser options, etc.)模块规则（配置加载程序，解析器选项等）
{
test: /\.jsx?$/,
include: [
path.resolve(\_\_dirname, "app")
],
exclude: [
path.resolve(\_\_dirname, "app/demo-files")
],
// these are matching conditions, each accepting a regular expression or string

//这些是匹配条件，每个条件都接受正则表达式或字符串
// test and include have the same behavior, both must be matched

//测试和包含行为相同，两者必须匹配
// exclude must not be matched (takes preferrence over test and include)

//排除项不得匹配（优先于测试并包含）
// Best practices:最佳做法：
// - Use RegExp only in test and for filename matching

//-仅在测试中和文件名匹配中使用RegEx
// - Use arrays of absolute paths in include and exclude

//-在包含和排除中使用绝对路径数组
// - Try to avoid exclude and prefer include

//-尽量避免排除，而希望包括
issuer: { test, include, exclude },
// conditions for the issuer (the origin of the import)

//发行人的条件（导入来源）
enforce: "pre",
enforce: "post",
// flags to apply these rules, even if they are overridden (advanced option)

//标记以应用这些规则，即使它们被覆盖（高级选项）
loader: "babel-loader",
// the loader which should be applied, it'll be resolved relative to the context

//应该使用的加载程序，它将相对于上下文进行解析
options: {
presets: ["es2015"]
},
// options for the loader//加载程序的选项
},
{
test: /\.html$/,
use: [
// apply multiple loaders and options//应用多个加载程序和选项
"htmllint-loader",
{
loader: "html-loader",
options: {
/ *...* /
}
}
]
},
{ oneOf: [ / *rules* / ] },
// only use one of these nested rules//仅使用这些嵌套规则之一
{ rules: [ / *rules* / ] },
// use all of these nested rules (combine with conditions to be useful)

//使用所有这些嵌套规则（与条件结合使用）
{ resource: { and: [ / *conditions* / ] } },
// matches only if all conditions are matched//仅在所有条件都匹配时才匹配
{ resource: { or: [ / *conditions* / ] } },
{ resource: [ / *conditions* / ] },
// matches if any condition is matched (default for arrays)

//如果条件匹配则匹配（数组默认）
{ resource: { not: / *condition* / } }
// matches if the condition is not matched//如果条件不匹配则匹配
],
/\* Advanced module configuration (click to show) \*// \*高级模块配置（单击以显示）\* /
},

重命名，冲突的地方怎么解决等
resolve: {
// options for resolving module requests//解决模块请求的选项
// (does not apply to resolving to loaders)//（不适用于解析加载程序）
modules: [
"node\_modules",
path.resolve(\_\_dirname, "app")
],
// directories where to look for modules//目录查找模块
extensions: [".js", ".json", ".jsx", ".css"],
// extensions that are used//使用的扩展名
alias: {
// a list of module name aliases//模块名称别名的列表
"module": "new-module",
// alias "module" -> "new-module" and "module/path/file" -> "new-module/path/file"

//别名“模块”->“新模块”和“模块/路径/文件”->“新模块/路径/文件”
"only-module$": "new-module",
// alias "only-module" -> "new-module", but not "only-module/path/file" -> "new-module/path/file"

//别名“仅模块”->“新模块”，而不是“仅模块/路径/文件”->“新模块/路径/文件”
"module": path.resolve(\_\_dirname, "app/third/module.js"),
// alias "module" -> "./app/third/module.js" and "module/file" results in error

//别名“模块”->“ ./app/third/module.js”和“模块/文件”导致错误
// modules aliases are imported relative to the current context

//相对于当前上下文导入模块别名
},
/\* Alternative alias syntax (click to show) \*/

/ \*替代别名语法（单击以显示）\* /
/\* Advanced resolve configuration (click to show) \*/

/ \*高级解析配置（单击以显示）\* /
},

//性能相关-**enum 是枚举的意思**
performance: {
[hints](https://webpack.js.org/configuration/performance/#performance-hints): "warning", // enum提示:警告 枚举
maxAssetSize: 200000, // int (in bytes),最大资产规模
maxEntrypointSize: 400000, // int (in bytes)最大入口点大小
assetFilter: function(assetFilename) {资源筛选 （资源文件名）
// Function predicate that provides asset filenames提供资产文件名的函数谓词

assetFilename==资源文件名称

endsWith=======尾端
return assetFilename.endsWith('.css') || assetFilename.endsWith('.js');
}
},

//开发工具相关
[devtool](https://webpack.js.org/configuration/devtool): "source-map", // enum
// enhance debugging by adding meta info for the browser devtools通过为浏览器devtools添加元信息来增强调试
// source-map most detailed at the expense of build speed.//以牺牲构建速度为代价的最详细的源映射。
context: \_\_dirname, // string (absolute path!)//字符串（绝对路径！）
// the home directory for webpackwebpack的主目录
// the entry and module.rules.loader optionentry和module.rules.loader选项
// is resolved relative to this directory是相对于此目录解析的
[target](https://webpack.js.org/configuration/target): "web", // enum

webpack可以针对多种环境或目标进行编译。要了解target详细内容，请通读目标概念页面

async-node

编译以在类似Node.js的环境中使用（ 异步使用 fs 和 vm加载块）

electron-main

为主要程序编译 电子。

electron-renderer

编译 电子 的渲染过程中，采用提供了一个目标 JsonpTemplatePlugin ， FunctionModulePlugin 为浏览器环境， NodeTargetPlugin 并 ExternalsPlugin 为CommonJS的和电子的内置模块。

electron-preload

编译 电子 的渲染过程中，采用提供的目标 NodeTemplatePlugin 与 asyncChunkLoading 设置为 true ， FunctionModulePlugin 为浏览器环境， NodeTargetPlugin 并 ExternalsPlugin 为CommonJS的和电子的内置模块。

node

编译以在类似Node.js的环境中使用（使用Node.js require 加载块）

node-webkit

编译以在WebKit中使用，并使用JSONP进行块加载。允许导入内置的Node.js模块和 nw.gui （实验性）

web

编译以在类似浏览器的环境中使用 （默认）

webworker

编译为WebWorker

// the environment in which the bundle should run包应在其中运行的环境
// changes chunk loading behavior and available modules更改块加载行为和可用模块
[externals](https://webpack.js.org/configuration/externals): ["react", /^[@angular](https://webpack.js.org/configuration/externals)/],
// Don't follow/bundle these modules, but request them at runtime from the environment

//不要跟踪/捆绑这些模块，而是在运行时从环境中请求它们
[stats](https://webpack.js.org/configuration/stats): "errors-only",
// lets you precisely control what bundle information gets displayed

//允许您精确控制显示的捆绑信息
devServer: {
proxy: { // proxy URLs to backend development server//到后端开发服务器的代理URL
'/api': 'http://localhost:3000'
},
contentBase: path.join(\_\_dirname, 'public'), // boolean | string | array, static file location
compress: true, // enable gzip compression启用gzip压缩
historyApiFallback: true, // true for index.html upon 404, object for multiple paths对于404上的index.html为true，对于多路径为object
hot: true, // hot module replacement. Depends on HotModuleReplacementPlugin热模块更换。取决于HotModuleReplacementPlugin
https: false, // true for self-signed, object for cert authority对于自签名为true，对于证书颁发机构为object
noInfo: true, // only errors & warns on hot reload仅在热重新加载时出现错误和警告
// ...
},

插件
plugins: [
// ...
],
// list of additional plugins附加插件列表
/\* Advanced configuration (click to show) \*/高级配置（单击以显示）