```
//webpack的默认配置文件
/**
 * webpack默认配置文件名称叫webpack.config.js,打包的时候会默认查找这个文件
 * 如果默认配置文件叫其他名如webpackconfig.js,这时候运行npx webpack --config
webpackconfig.js
 * 意思是打包的时候以webpackconfig.js为配置文件
*/
const path=require('path');
/**
 * html-webpack-plugin
 * 此插件会在打包结束后，自动生成一个html文件，并把打包生成的js自动引入到这个html文件中
*/
const HtmlWebpackPlugin=require('html-webpack-plugin')
/**
 * clean-webpack-plugin:在打包之前会先删除dist下的内容，然后再生成
*/
const {CleanWebpackPlugin}=require('clean-webpack-plugin')
/**
 * sourceMap:是一种映射关系。他知道dist目录下main.js96行报错，实际上对应的
 * 是src目录下的index.js文件中的第一行报错，使用映射关系，可配置devtool,生产环境配置成
 * cheap-module-source-map比较好
*/
module.exports={
    mode:'development',//默认production,development||none,设置为development时，生成的
bundle.js是不被压缩的代码
    devtool:'cheap-module-eval-source-map',//将map文件以base64字符串的形式打包到对应的脚本的
最末尾
    //entry:'./src/index.js',
    entry:{
        //将src下的index.js打包两份
        main:'./src/index.js',
        sub:'./src/index.js',
    },
    output:{
        //publicPath:'http://cdn.com.cn',//将静态资源放在cdn上
        //filename:'bundle.js',//打包后的文件名，默认是main.js
        filename:'[name].js',//对应entry的名称
        path:path.resolve(__dirname,'dist'),//打包后的文件要放在哪，后面是绝对路径，打包后生
成dist文件夹
    },
    module:{
        rules:[{
            //处理图片结尾的文件
            test:/\.(jpeg|jpg|png|gif)$/i,
            use:{
                /**
                 * 如果使用url-loader,会将图片处理成base64文件,并返回一个data-url打包到
bundle.js中,
                 * 并不会单独生成一个文件，这虽然会省去http请求，但如果图片文件过大，页面
                 * 加载会很慢，如果图片过小，这种加载方法会少一次http请求
                 * 使用file-loader,会将文件上的import/require解析为url,发送到输出文件夹并返回
相对url
                 */
                loader:'file-loader',
```

```
                //loader:'url-loader',
                options:{
                    //placeholder 占位符
                    //name:'[name]_[hash].[ext]',//logo_xxxxhash.jpeg
                    //name:'[name].[ext]',//logo.jpeg
                    //name:'[path][name].[ext]',//src/logo.jpeg,会在images文件夹下生成完整
的原目录结构
                    name(resourcePath,resourceQuery){
                        //'resourcePath':'src/logo.jpeg'
                        //'resourceQuery':'?foo=bar'
                        if(process.env.NODE_ENV==='development'){
                            return '[path][name].[ext]';
                        }
                        return '[contenthash].[ext]';
                    },
                    outputPath:'images/',  //将图片文件打包生成到dist/images目录中
                    // outputPath(url,resourcePath,context){
                        // //url:eab9a512fc7b3c3d11569ecf5a4c5287.jpeg
                        // //resourcePath:original absolute path to asset.
//'/Users/lijiangbo/Desktop/webpack-demo/src/logo.jpeg'
                        // //context:directory where stored asset('rootContext') or
context option.  //'/Users/lijiangbo/Desktop/webpack-demo'
                        // //to get relative path you can use :const
relativePath=path.relative(context,resourcePath)
                        // if(/logo\.jpeg/.test(resourcePath)){
                            // return `outer_output_path/${url}`;
                        // }
                        // if(/webpack-demo/.test(context)){
                            // return `image_output_path/${url}`;
                        // }
                        // return `output_path/${url}`
                    // },
                    limit:10240,//如果图片小于10kb,会将文件生成base64生成到bundle.js中，大于
2kb单独生成到images目录中,
                    //publicPath:'assets',
                    // publicPath(url,resourcePath,context){
                        // if(/logo\.jpeg/.test(resourcePath)){
                            // return 'assets';
                            // return `other_public_path/${url}`;
                        // }
                        // if(/webpack-demo/.test(context)){
                            // return `image_public_path/${url}`;
                        // }
                        // return `public_path/${url}`
                    // }
                }
            }
        },{
            test:/\.css$/i,
            use:[{
                    loader:'style-loader',
                    options:{
                        injectType:'styleTag'
                    }
                },
                'css-loader'
            ]
```

```
        }]
    },
    /**
     * plugin
     * 可以在webpack运行到某个时刻的时候，帮你做一些事情，比如HtmlWebpackPlugin会在打包结束后生成一个html，
     * 并将打包后的js注入html文件
     */
    plugins:[new HtmlWebpackPlugin({
                template:'src/index.html'
            }),
            new CleanWebpackPlugin()],
}
```