- webpack
  - webpack.config.js
    - mode
      - none 使用原始的方式打包
      - development 打包速度优化
      - production production 会自动开启 tree Shaking
    - entry
      - 单入口打包 "./src/main.js"
      - 多入口打包
        ```
        {
          index: './src/index.js',
          album: './src/album.js'
        }
        ```
        - 代码分割
          - 多入口打包 适用：多页应用程序
            - 一个页面对应一个打包入口
            - 公共部单独提取
          - 动态导入
            ```
            import("./posts/posts").then(({ default: album }) => {
              mainElement.appendChild(posts());
            });
            ```
            - 通过这种格式书写，就能使用动态导入的效果
            - vue中的router就是按照这种动态导入的功能进行使用的
          - 魔法注释
            ```
            import(/* webpackChunkName: 'album */"./posts/posts").then(({ default: album
            }) => {
              mainElement.appendChild(posts());
            });
            ```
            - 用/**/的格式添加，就能实现自定义的分包名称了
            - 设置相同的名称就会被打包到一起
        - 默认 ./src/index.js
    - output
      - filename 导出文件名称
      - path 导出文件绝对路径
      - publicPath 设置工作目录路径
      - 多个出口
        - filename:'[name].bunble.js'
        - 用于缓存的hash值
          - filename:'[name]-[hash].bunble.js' 全局级别的，一个更改全部打包都更改hash值
          - filename:'[name]-[chunkhash:8].bunble.js' 分块hash，不同块，hash不一样,.8表示标识的长度
          - chunkhash是解决缓存的最好方法
      - 默认 ./dist/main.js
    - module
      - rules
        - test 检测文件后缀
        - use 使用对应的loader
        ```
        {
          test: /.css$/,
          use: ["style-loader", "css-loader"],
        }
        ```
        每一个loader都要在这里定义对应的名称
      - loader
        - loader加载器，webpack核心
          - 编译转换类
            - css-loader 将css文件编译到js文件中
            - style-loader 将文件中的css通过style加载到页面中
          - 文件操作类
            - file-loader 将文件拷贝到打包目录，然后返回路径地址
            - url-loader 将文件通过base64的形式，作为路径返回
              - 一般设置10KB以内的文件用这种形式
              - 当文件超过10KB默认使用file-loader
            - html-loader 将html文件进行打包
              - 可添加html属性的加载器调用
          - 代码检查类
        - 自定义loader 通过路径引入loader文件
          - 返回值，一定要是javascript代码，例如console.log(123)
          - 需要接收一个source的值，这个值的内容就是加载的文件内容
          ```
          const marked = require("marked");
          module.exports = (source) => {
            const html = marked(source);

            return `module.exports = ${JSON.stringify(html)}`;
          };
          ```
    - Plugins 插件列表，可添加各种功能的插件
      - 常用插件
        - clean-webpack-plugin 用于清空dist目录下的文件
        - html-webpack-plugin 将生成的js文件自动注入到html文件中
        - copy-webpack-plugin 拷贝文件
        - webpack.HotModuleReplacementPlugin() 热加载HMR插件
        - webpack.DefinePlugin() 用于定义全局形式的变量
          ```
          new webpack.DefinePlugin({
            API_BASE_URL: JSON.stringify("https://api.example.com"),
          })
          ```
        - MiniCssExtractPlugin 将css将打包结果中提取出来的插件
          - 直接通过link的方式进行引入，这里就不需要stylr-loader进行style样式导入了，应该用MiniCssExtractPlugin.loader
          - 注意：样式文件体积不大，提取出来就没有意义，超过150KB才有意义
        - optimize-css-assets-webpack-plugin 用于css文件进行压缩 一般压缩功能，都统一放在minimizer中使用
        - terser-webpack-plugin webpack内置的js插件
      - 自定义插件 必须是一个函数或者是一个包含apply方法的对象 获取compiler参数 参数内包含所有钩子函数
        - compiler.hooks.emit.tap("MyPlugin", (compilation) => {}
          - compilation 包含打包的所有内容
          - compilation.assets 所有打包的文件名
          - compilation.assets[name].source() 单个文件的文件内容
          ```
          compilation.assets[name] = {
              // source 是为了覆盖原来的source上面的值
              source: () => withoutComments,
              // 这里的size是规定的，一定要求返回文件的大小
              size: () => withoutComments.length,
            };
          ```
          最后需要覆盖对应的值
        - emit 钩子，所有打包内容写进dist之前执行
    - devServer
      - webpack-dev-server
        - 自动监听编译文件
        - 自动刷新浏览器效果
        - 这里的打包内容是暂时存储在内存当中
      - contentBase 设置一些平时开发不进行编译的静态文件路径
      - proxy 请求代理
        - target 请求的地址前缀
        - pathRewrite 请求关键字重写
        - changeOrigin 设置不能使用localhost:8080 作为请求的主机名
        ```
        proxy: {
            "/api": {
              target: "https://api.github.com",
              pathRewrite: {
                "^/api": "",
              },
              changeOrigin: true,
            },
          },
        ```
      - hot
        - 热加载HMR
          - true/false hot模式，出错之后会调用自动刷新，出错的时候会被刷新掉
          - hotOnly Only模式，错误后不会去执行自动刷新
        - style样式
          - 因为样式是使用loader，loader中定义了，热更新规则，所以可以直接使用
          - 因为样式规则简单可以直接替换，所以热更新可以通用
        - javsscript
          - 毫无规律，无法进行通用
          - 使用框架时，框架就是固定的结构所以可以使用通用的规则进行热更新
    - devtool 用于添加辅助工具
      - source-map
        - 开发环境
          - cheap-module-eval-source-map
            - 首次打包启动速度慢，重写打包相对较快
            - source-map中看到的是打包之前的代码
            - 只定位到行数
        - 生产环境
          - none 不生产source-map
            - source map 会暴露源代码
            - 调试应该是开发阶段的事情
          - nosources-source-map
            - 定位到行数
            - 但是不显示代码
    - optimization 代码优化功能
      - tree Shaking
        - usedExports 负责标记"枯树叶"
        - minimize 负责"摇掉" 它们
        - 组合起来，就是tree Shaking功能
          - babel中自动关闭了将ESmudule转成commJS的功能
          - 代码如果是CommJS格式，usedExports将会失效
          - 摇掉之后会对文件进行压缩处理
      - Scope Hoisting concatenateModules 尽可能将所有模块合并输出到一个函数中，即提升了运行效率也减少了代码体积
        - 作用域提升，配合上面tree Shaking，进一步减少体积
      - sideEffects
        - 副作用： 模块执行时除了导出成员之外所作的事情
        - 一般用于NPM包标记是否有副作用
        - true/false
          - 需要在optimization中为true开启
          - 再在package.json，中听过sideEffects,说明有副作用或者没有副作用true/false
          - 如果有副作用，可以通过数组标识，需要打包的副作用模块
          - 原型方法，CSS代码都属于副作用
        - 例如: 原型方法的添加，就是副作用的一种
          - packge.json
            ```
            "sideEffexts": [
                "./src/extend.js",
                "*.css"
              ]
            ```
          - 要在配置文件中书写，还要再packge文件中标注副作用文件
      - splitChunks 多入口打包，将公共部分单独提取存放 chunks: 'all' 就表示所有公共部分都进行提取
      - minimizer 压缩列表，一般在固定场景才会打开，例如生产模式打包的时候，会开启
        - 压缩css功能，这里开启后会让js自带的压缩失效
        - 重新添加js压缩功能
    - webpack资源加载方式
      - 遵循ES Modules 标准的 import 声明
      - 遵循CommonJS 标准的 require 函数
      - 遵循AMD 标准的 define 函数和 require函数
      - 样式代码中的 @import 指令和 url 指令
      - HTML代码中图片的标签的src属性
      - 不要混合使用，使用一种统一标准就行