Webpack 简介

何为Webpack

JavaScript 模块打包工具
解决模块之间的依赖
模块打包：将各个模块按照特定规则和顺序组织在一起，最终合并为一个或多个JS文件

为什么需要Webpack

一个工程中有多个JS文件，最早只能用script标签插入。

- 需要手动维护JS加载顺序，因为script之间可能会有依赖
- HTTP2没出现，简历连接成本很高，过多请求影响网页渲染速度(存疑？HTTP1.1 长连接)
- 每个scrip中顶层作用域即全局作用域，容易造成全局作用域污染

打包解决了这些问题：

- 导入导出语句可以清晰的看到依赖关系
- 减少网络开销，只需要请求合并后的资源
- 多个模块之间作用域是隔离的，不会有命名冲突

Webpack解决了

- 无法使用code splitting 和 tree shaking
- 大部分npm包还是commonJS形式，浏览器不支持其语法
- 浏览器兼容性问题

模块打包工具

模块打包工具（module bundler）任务就是解决模块之间的依赖，使其打包后的结果能运行再浏览器上。

- 将存在依赖关系的模块按照特定规则合并成单个JS文件，一次性加载进页面中
- 在页面初始时加载一个入口模块，其他模块异步进行加载

Webpack，Rollup，Parcel

为什么选择Webpack

- 支持多种模块标准，AMD,CommonJS,CMD,ES6等，对于同时使用多种模块标准的工程非常有用。能处理不同类型模块之间的依赖关系
- 有完备的代码分割（code splitting）解决方案。
  分割打包后的资源，首次只加载必要部分，其他部分在后面动态加载。
  对资源体积大的应用尤为重要，有效减小资源体积，提升首页渲染速度

- 处理各种类型资源。js，样式，模板，图片等。 loader能处理
- 社区支持。

实践Webpack

```
npx webpack --entry=./index.js --output-filename=bundle.js --mode=development
```

- entry 资源打包入口，Webpack在这里进行模块依赖的查找
- output-filename 输出资源名， dist文件夹中的名字
- mode 打包模式 默认development模式。
  development，production，none 3种模式
  dev，prod模式会自动添加一些配置

可以直接写在package.json中scripts里

配置文件webpack.config.js

```
module.exports = {
    entry:'./src/index.js',
    output:{
        filename:'bundle.js'
    },
    mode:'development'
}
```

直接运行 webpack会自动访问配置文件

webpack-dev-server

需要在webpack.config.js中配置devServer

- 令Webpack进行模块打包，并处理打包结果的资源请求
- 普通的Web Server，处理静态资源文件请求

与webpack命令的区别在于，webpack生成dist，webpack-dev-server将打包结果放在内存中

webpack-dev-server 会自动刷新(live-reloading)

小结

Webpack功能，处理模块之间的依赖，串联起来合并一个或多个JS文件
配置文件 Webpack.config.js
webpack-dev-server,打包的东西在内存中，有自动刷新（live-reloading）功能，监听文件变化，自动刷新页面