1、构建工具的功能

转换ES6、转换JSX，CSS前处理器、压缩混淆、图片压缩

2、前端构建工具演变历史

ant + YUI Tool → grunt → Fis3 / gulp → rollup / webpack / parcel

task runner

3、基础用法、概念、代码基本用法

① 基本概念：entry 入口为 entry 非代码文件也会加入

单入口：entry是一个字符串

```
module.exports = {
    entry: './path/to/me/entry/file.js'
}
```

多入口：entry是一个对象 → 键值对的形式

```
module.exports = {
  entry: {
    app: './src/app.js',
    adminApp: './src/adminApp.js'
  }
}
```

② 基本概念 Output

单入口 → 指定 filename & path

多入口 → 通过占位符确保文件名称的唯一

③ 核心概念：Loaders

文件以字符串形式导入 ↑

原生只支持 js 与 json 通过 Loaders 支持其它文件

babel-loader、css-loader、less-loader、ts-loader、file-loader、raw-loader、thread-loader

用法：

```
module: {
  rules: [
    { test: /\.txt$/, use: 'raw-loader' }
  ]
}
```

test 指定匹配规则

use 指定 loader 名称

④ 核心概念 Plugins

用于 bundle 文件优化、资源管理和环境变量注入，作用于整个构建过程

commonsChunkPlugin —— 提取公共 js

CleanWebpackPlugin —— 清理构建目录

Plugins: [↑] 放置到 Plugins 数组里

⑤ 核心概念 Mode

Mode 用于指定当前构建环境为 development、production 与 node

会默认开启一些优化选项

解析 CSS

css-loader 用于加载 .css 文件，并转换为 commonjs 对象

style-loader 将样式通过 <style> 标签插入到 head 中

解析图片 解析字体

file-loader 用于处理文件 ⟶ 图片处理

file-loader 也可以用于解析字体

也可以使用 url-loader 处理图片与字体

小资源自动转换 base64

```
use: [
  {
    loader: 'url-loader',
    options: {
      limit: 10240 ⟶ 表示小于10240b自动转换为base64格式
    }
  }
]
```

⑥ webpack 中文件监听 ⟶ 两种方式

- 启动 webpack 命令时，带上 --watch 参数
- 在 webpack.config.js 设置 watch: true

缺点：需要手动刷新浏览器

原理：轮询判断 文件最后编辑时间是否变化

```
module.export = {
  watch: true,
  watchOptions: {
    ignored: /node_modules/, ⟶ 文件监听忽略
    aggregateTimeout: 300, ⟶ 间隔多少秒去执行构建
    poll: 1000 ⟶ 每秒轮询次数
  }
}
```

⑦ webpack 热更新及其构建文件方式

WDS：webpack-dev-server WDS 不刷新浏览器、WDS 不输出文件，而是放在内存之中

"webpack-dev-server --open"

热更新另一种方式：webpack-dev-middleware。 WDM 将 webpack 输出文件输给服务器。

△ 热更新的原理

Webpack Compile：将 JS 编译成 bundle

HMR Server：将热更新的文件输出给 HMR Runtime。

Bundle server：提供文件在浏览器中的访问

HMR Runtime：会被注入到服务器，更新文件的变化

bundle.js 构建输出的文件

热更新实现原理

1、webpack-dev-server 启动本地服务 ⟶ 使用 webpack 方法去生成 compiler 实例

① 本地 server 启动服务，让浏览器可以请求本地静态资源

② server 启动后，启动 websocket 服务，可以建立本地服务与浏览器双向通信

① Webpack通过 Compiler类 run方法开启编译构建过程，通过 Watch方法监听文件变更，变更完成后重新编译

② webpack-dev-middleware 1、通过 memory-fs 将静态资源请求直接访问内存
2、通过Express服务器启动一个Server

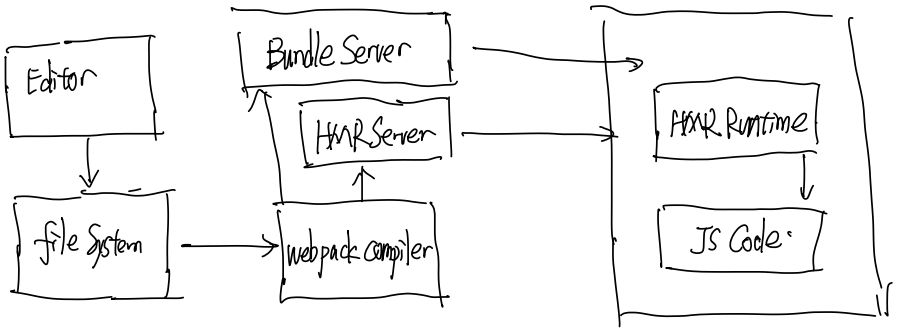
③ 浏览器通过SSE接收Server端关于已经更新的消息 ⟹ SSE Server-Sent Events
服务端向客户端声明：发送数据类型为流（stream）
SSE基于 HTTP 单向通道，仅支持Server ⟹ Client.
客户端Api部署在Event Source

④ 浏览器通过 HotModuleReplacement布署在项目中的代码发送 jsonp请求来获最新资源 → runtime.js来定义如何请求

⑤ 通过hash值判断更新的模块（Hot Check）

```
if ('EventSource' in window) {
    var source = new EventSource(url)
}
```
⟹ source实例有 onopen、onmessage、onerror方法

HMR Server + HMR Runtime Plugin作用为注入 Runtime到 bundle.js → websocket + SSE



△文件指纹策略