Day03 Webpack/Babel/工程化

全栈然叔

预习内容

- 什么是webpack
  https://www.bilibili.com/video/BV12S4y1k7pr
- 手写webpack
  https://www.bilibili.com/video/BV1dV411p7gp/
- Bundless时代、Vite
  https://www.bilibili.com/video/BV1Df4y1n777

Lesson03 Webpack

```
- Webpack的核心概念
- Sourcemap
- 文件指纹技术
- Babel与AST
- Treeshaking树摇
- 优化篇
  - 构建速度
  - 提高页面性能
- 原理篇
  - Webpack
  - Plugin
  - Loader
- 展望篇
  - Vite与Bundleless
```

核心概念

- entry: 入口，webpack 构建第一步；
- output: 输出
- loader: 模块转换器，用于将模块的原内容按照需求转换成新内容；
- plugin: 扩展插件,在 webpack 构建过程的特定时机注入扩展逻辑，用来改变或优化构建结果；
- mode: 控制打包环境
  通过选择 development, production 或 none 之中的一个，来设置 mode 参数，你可以启用 webpack 内置在相应环境下的优化。其默认值为 production。
  环境
- devserver: 是一个小型的 node.js Express 服务器，使用 webpack-dev-middleware 中间件来为通过

webpack 打包生成的资源文件提供 web 服务。

---

entry point(入口起点)

> 入口起点(entry point) 指示 webpack 应该使用哪个模块，来作为构建其内部 依赖图(dependency graph) 的开始。进入入口起点后，webpack 会找出有哪些模块和库是入口起点（直接和间接）依赖的。

index.js

```
import { add } from "./add";
console.log(add(3, 3));
```

add.js

```
export const add = (a,b) => a + b
```

webpack.config.js

```
module.exports = {
  entry: "./src/index.js"
};
```

---

output (输出)

> output 属性告诉 webpack 在哪里输出它所创建的 bundle，以及如何命名这些文件。主要输出文件的默认值是 ./dist/main.js，其他生成文件默认放置在 ./dist 文件夹中。

```
const path = require("path");
const HtmlWebpackPlugin = require('html-webpack-plugin');
module.exports = {

  output: {
    path: path.resolve(__dirname, "dist"),
    filename: "main.bundle.js",
  },

};
```

Loader加载器

> loader 用于对模块的源代码进行转换。loader 可以使你在 import 或 "load(加载)" 模块时预处理文件。

- 默认只能处理json与js
- 其他文件需要通过专门的加载器处理

hello.txt

```
Hello Raw Loader
import hello from "./hello.txt";
document.writeln(hello);
yarn add raw-loader
```

webpack.config.js

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

---

plugin 插件

> 扩展插件，在 webpack 构建过程的特定时机注入扩展逻辑，用来改变或优化构建结果；

```
yarn add html-webpack-plugin -d
```

webpack.config.js

```


const HtmlWebpackPlugin = require('html-webpack-plugin');
module.exports = {

  plugins: [new HtmlWebpackPlugin({ template: './src/index.html' })],

};

<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
</head>
<body>
    <h1>Hello webpack3</h1>
</body>
```

```
</html>
```

DevServer开发服务器

> 提供了一个基本的 web server，并且具有 live reloading(实时重新加载) 功能。。

- 静态服务比如:图片
- live reloading(实时重新加载)
- 反向代理接口

```
yarn add -D webpack-dev-server
npx webpack serve
```

SourceMap是什么

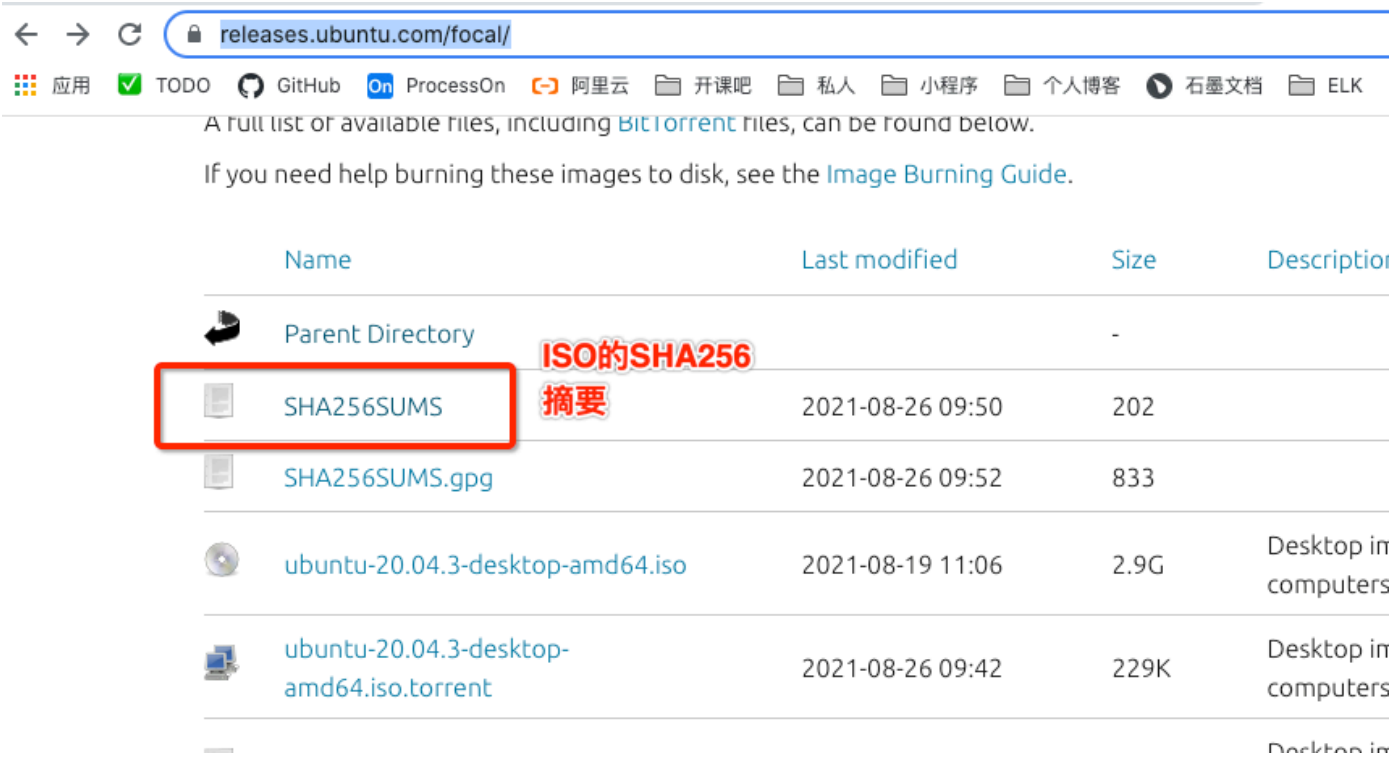
在编译处理的过程中，在生成产物代码的同时生成产物代码中被转换的部分与源代码中相应部分的映射关系表。

模式	解释
eval	每个module会封装到 eval 里包裹起来执行，并且会在末尾追加注释 `//@ sourceURL`.
source-map	生成一个**SourceMap**文件（编译速度最慢）
hidden-source-map	和 source-map 一样，但不会在 bundle 末尾追加注释.
inline-source-map	生成一个 **DataUrl** 形式的 SourceMap 文件.
eval-source-map	每个module会通过eval()来执行，并且生成一个DataUrl形式的SourceMap.
cheap-source-map	生成一个没有列信息（column-mappings）的SourceMaps文件，不包含loader的 sourcemap（譬如 babel 的 sourcemap）
cheap-module-source-map	生成一个没有列信息（column-mappings）的SourceMaps文件，同时 loader 的 sourcemap 也被简化为只包含对应行的。

文件指纹是什么怎么用

如何判断Ubuntu ISO镜像的真伪

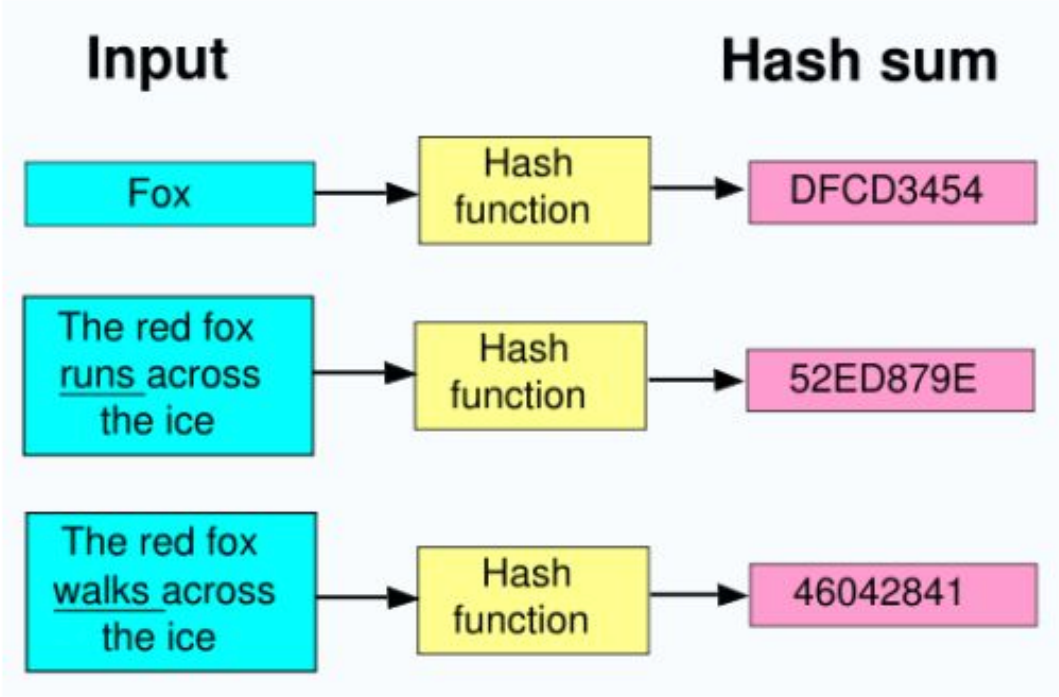
Ubuntu https://releases.ubuntu.com/focal/



摘要与哈希算法

摘要算法又称哈希算法、散列算法。摘要也称哈希值，表示输入任意长度的数据，都会输出固定长度的数据。通过摘要算法（比如MDS和SHA-1）就可以得到该哈希值。

什么是哈希算法



- 不定长输入转定长摘要
- 满足雪崩效应
- 单项不可逆

常见的哈希算法

- MD5： 输出128bit长度的二进制串
- SHA-1：输出160bit长度的二进制串
- SHA-256： 输出256bit长度的二进制串
- SHA-512： 输出512bit长度的二进制串

Webpack与文件指纹

- **版本管理：** 在发布版本时，通过文件指纹来区分 修改的文件 和 未修改的文件。
- **使用缓存：** 未修改的文件，文件指纹保持不变，浏览器继续使用缓存访问。

文件指纹设置

我们在配置文件（`webpack.config.js`）中，通过占位符设置文件指纹。

占位符

名称	含义
[ext]	资源后缀名
[id]	文件标识符
[name]	文件名称
[path]	文件的相对路径
[folder]	文件所在的文件夹
[hash]	模块标识符的 hash
[chunkhash]	chunk 内容的 hash
[contenthash]	文件内容的 hash
[query]	文件的 query，例如，文件名？后面的字符串
[emoji]	一个随机的指代文件内容的 emoji

设置方法

```
// webpack.config.js

module.exports = {
  // ...
  entry: {
    app: "./src/app.js",
    index: "./src/index.js",
  },
  output: {
    filename: "[name][chunkhash:8].js",
    // ...path
  },
};
```

CSS文件、图片和字体的设置

https://juejin.cn/post/7030737652173242382#heading-9

谈谈什么是chunk

webpack 会根据模块依赖图的内容组织分包 —— Chunk 对象，默认的分包规则

- 同一个 `entry` 下触达到的模块组织成一个 chunk
- 异步模块单独组织为一个 chunk
- `entry.runtime` 单独组织成一个 chunk

Webpack原理

课件： https://juejin.cn/post/6961961165656326152

视频： https://www.bilibili.com/video/BV1dV411p7gp/

loader原理

loader是链式传递的，对文件资源从上一个loader传递到下一个，而loader的处理也遵循着从下到上的顺序，我们简单了解一下loader的开发原则：

1. 单一原则: 每个Loader只做一件事，简单易用，便于维护；
2. 链式调用: Webpack 会按顺序链式调用每个Loader；
3. 统一原则: 遵循`Webpack`制定的设计规则和结构，输入与输出均为字符串，各个`Loader`完全独立，即插即用；

4. 无状态原则：在转换不同模块时，不应该在loader中保留状态；

因此我们就来尝试写一个`less-loader`和`style-loader`，将`less文件`处理后通过style标签的方式渲染到页面上去。

plugin 原理

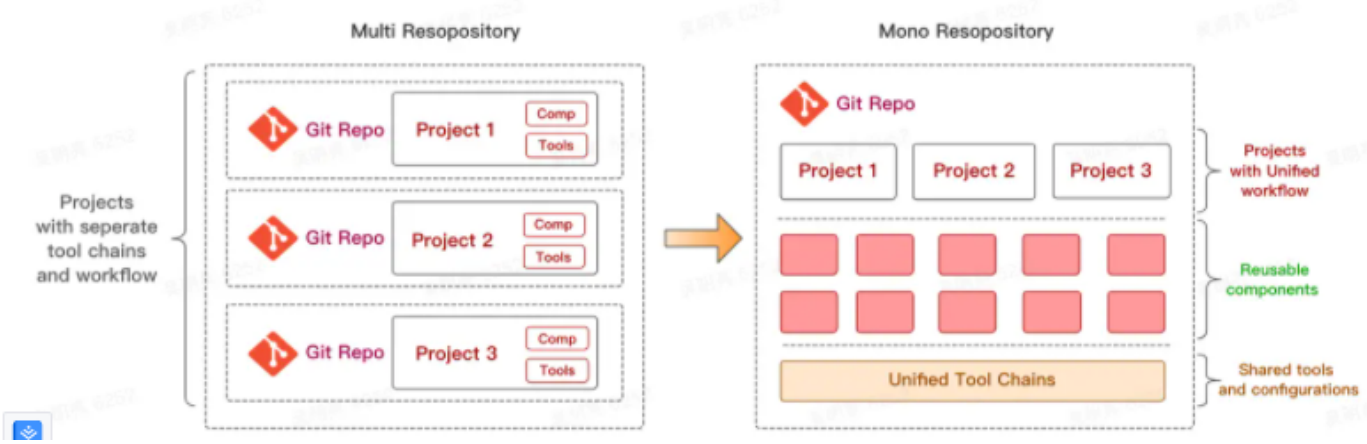
> 参考异常监控中的sourcemap上传部分

https://juejin.cn/post/6844904119136698381#heading-27

工程化展望

monorepo风格

Monorepo 其实不是一个新的概念，在软件工程领域，它已经有着十多年的历史了。概念上很好理解，就是把**多个项目**放在**一个仓库**里面，相对立的是传统的 MultiRepo 模式，即每个项目对应一个单独的仓库来分散管理。



bundleless与vite

Bundleless 说到底，就是指**无打包构建**，与我们当下流行的**打包构建**相对，而**打包器**则是我们前端开发者用于将 JS 模块打包成单一的、可在浏览器内运行的文件的工具。

Vite原理需要了解

https://www.bilibili.com/video/BV1Df4y1n777