TypeScript & Webpack

Webpack

webpack 개요

1. webpack은?

Module bundler이다.

2. Module Bundler란?

간단히 말하자면 여러 모듈을 하나로 통합 한 파일을 출력하는 도구를 말함.
※ 출력 설정에 따라 여러 개의 파일을 출력 할 수 있다.
webpack 외에도 다음과 같은 모듈 Bundler가 존재한다.

Browserify
RequireJS

Webpack

webpack (Module Bundler) 관련 기사에서 자주 사용되는 용어

3.1 모듈

기능별로 분할 된 파일을 의미합니다.
따라서 webpack는 "여러 파일을 정리하여 한 개의 파일"로 출력하는 도구로 인식 해두면 일단은 문제 없습니다.

3.2 Bundle

정리 된 파일을 의미한다. Bundle file이라고도 한다. 따라서 다음의 의미는 대략 같다.
"번들이 크다"= "정리 된 파일의 크기가 크다"
"번들을 생성"= "정리 된 파일을 생성"

3.3 Build

webpack에서의 "빌드"는 "번들을 출력 할 때까지의 일련의 처리 '라는 뜻으로 사용되는 경우가 많다.
따라서 다음의 의미는 대략 같다.
"빌드"= "번들을 출력 할 때까지의 일련의 처리를 실행한다"
"빌드가 느린"= "번들을 출력 할 때까지의 일련의 처리가 늦다"

Webpack

왜 webpack을 이용하는가?

4. 왜 webpack을 이용하는가?

기능별로 파일을 분할(모듈화)하여 개발 할 수 있다.

자신이 만든 모듈만이 아니라 외부 모듈(npm 등으로 설치할 수 있는 패키지 등)도 이용할 수 있다.

요청 수를 줄일 수 있다.

종속성을 해결 한 파일을 출력 할 수 있다.

개발 작업의 분담과 테스트가 쉬워진다.

네임 스페이스를 생성 할 수 있다(변수의 경쟁과 글로벌 오염을 방지)

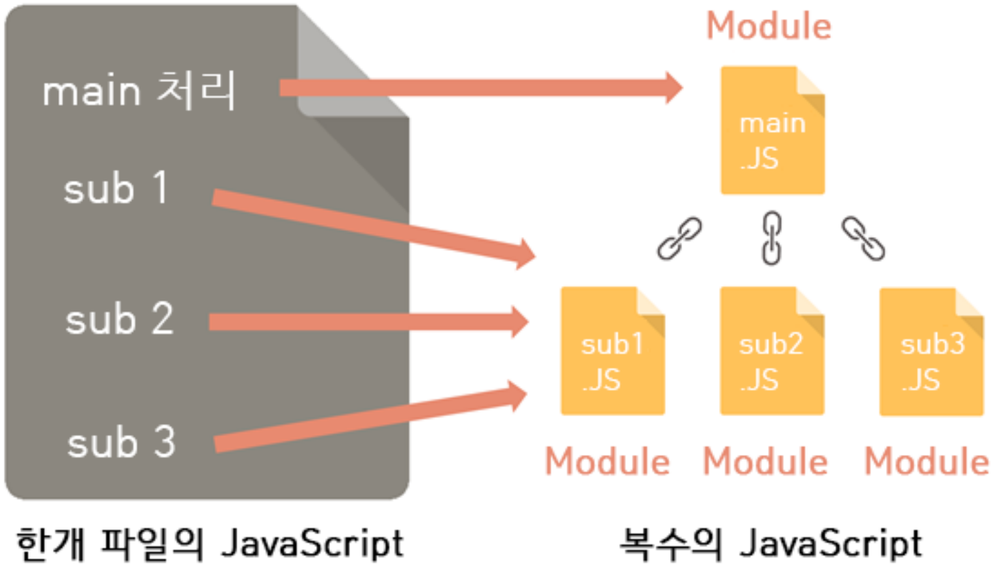
위와 같이 모듈의 개발 또는 재사용 시 번들이 매력적이기 때문이다.

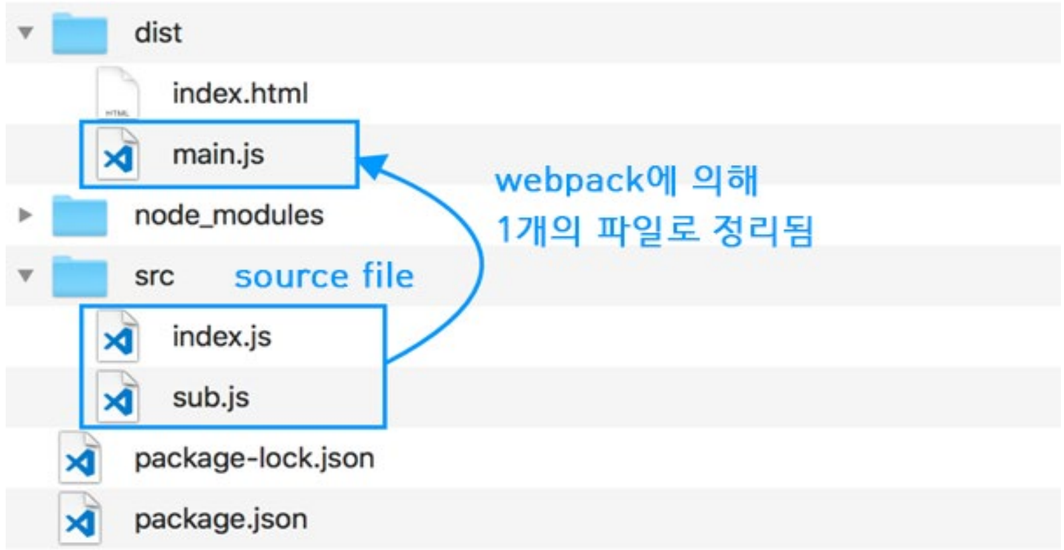
Webpack

왜 webpack을 이용하는가?

5. webpack에서 JS 파일을 정리

지금은 모듈 방식으로 JavaScript를 쓰는 것이 좋다.
하나의 JavaScript 파일에 긴 처리를 기술하면 가독성이 떨어진다. 이 문제를 해결하려면 여러 파일로 분할하는 것이 좋다. 웹 Front-End에서는 기능별로 분할 된 JavaScript 파일을 일반적으로 "모듈"이라고 한다.





Webpack 설치

1. package.json 파일 작성.

```
npm init -y
```

2. webpack, webpack-cli 전역 설치

```
npm install -g webpack webpack-cli
```

3. webpack, webpack-cli 로컬 설치

```
npm install webpack webpack-cli --save-dev
```

4. ts-loader, typescript 설치.

```
npm install ts-loader typescript --save-dev
```

5. uglifyjs-webpack-plugin 설치 [필요에 따라 설치]

```
npm install uglifyjs-webpack-plugin --save-dev
```

Webpack

webpack.config.js

webpack.config.js

```
const path = require("path");
module.exports = {
    mode: "production",
    entry: path.join(__dirname, ""),
    output: {
        filename: "",
        path: path.join(__dirname, "dist")
    },
    module: {
        rules: [
            {
            }
        ]
    }
}
```

Webpack

webpack.config.js

webpack.config.js

1. JS 출력 방식 설정

```
mode: "production"              // development, production, none
```

2. 변경할 파일 지정

```
entry: path.join(__dirname, "source/main.ts")
```

3. 출력 JS 파일 위치 및 이름 설정

```
output: {
        filename: "main.js",
        path: path.join(__dirname, "dist")
},
```

Webpack

webpack.config.js

webpack.config.js

4. module 설정

```
module: {
    rules: [
        {
            test: /\.ts$/,
            exclude: /node_modules/,
            use: "ts-loader"
        }
    ]
},
resolve: {
    extensions: [".ts", ".js"]
}
```