11장 자바스크립트와 상호 동작

내용

링크

- 이번 11장에서는 타입을 쓰지 않는 언어에서 타입스크립트를 접목하는 방법을 알아본다.
  - 서드 파티 자바스크립트 라이브러리
  - 빠른 패치를 위해 타입 안전성을 포기하는 등의 상황 같은 실전 상황

11.1 타입 선언

타입선언이란 .d.ts 확장자를 가진 파일이다.

- 타입이 없는 자바스크립트 코드에 타입스크립트 타입을 부여할 수 있는 수단
- 타입스크립트 코드(.ts) 타입 선언(.d.ts)
  - 타입만 포함. (값은 포함할 수 없음, 함수, 클래스, 객체 등)
  - 값을 정의할 수 없지만, declare 키워드를 사용해 자바스크립트의 다른 어딘가에 값이 있다는 사실을 선언할 수 있다.
  - 소비자가 볼 수 있는 대상에만 타입 선언 O (볼 수 없는 대상은 X 노출되지 않은 타입, 함수 안 지역 변수 등)

```
// Observable.ts
export class Observable<T> implements Subscribable<T> {
 public _isScalar: boolean = false
 constructor();
 static create<T>(subscribe?: (subscriber: Subscriber<T>) => TeardownLogic) {
  return new Observable<T>(subscribe)
 }
 }

// d.ts
export declare class Observable<T> implements Subscribable<T> {
    _isScalar: boolean
    constructor();
    static create<T>(
      subscribe?: (subscriber: Subscriber<T>) => TeardownLogic
    ): Observable<T>
}
```

- declarations 플래그를 활성화한 다음 TSC로 컴파일하면 .d.ts 타입 정의가 생성된다.
- declare 키워드 체킹 (타입선언에서는 클래스를 직접 정의할 수는 없지만, 그 대신 .d.ts파일에 대응하는 자바스크립트 파일 안에 정의했음을 선언(declare)할 수 있다)
- 구현부는 포함하지 않고 타입만 표시

- 타입 선언 -1. 타입스크립트로 부터 생성된 자바스크립트 파일에 대응하는 .d.ts파일을 검색한다. -2. 타입스크립트를 지원하는 코드 편집기는 이 .d.ts 파일들을 읽어해석한 다음 타입 힌트를 제공 -3. 타입스크립트 코드의 불필요한 재컴파일을 막아주어 컴파일 시간을 크게 줄여준다.

11.1.1 앰비언트 변수 선언

엠비언트 변수 선언은 한 프로젝트의 모든 .ts와 .d.ts 파일에서 임포트 없이 사용할 수 있는 전역 변수의 존재를 타입스크립트에 알리는 수단이다.

```
declare const process: {
  env: {
    NODE_ENV: 'development' | 'production'
  }
}

process = {
   env: {
    NODE_ENV: 'production'
   }
}
```

11.1.2 앰비언트 타입 선언

앰비언트 타입 선언은 앰비언트 변수 선언과 같은 규칙을 사용한다.

```
type ToArray<T> = T extends unknown[] ? T : T[]

function toArray<T>(a: T): ToArray<T> {
 // ...
}
```

11.1.3 앰비언트 모듈 선언

일부 타입을 빠르게 선언하고 안전하게 사용하고 싶을 때 앰비언트 모듈 선언을 사용하자.

```
declare module 'module-name' {
    export type MyType = number
    export type MyDefaultType = { a: string }
    export let myExport: MyType
    let myDefaultExport: MyDefaultType
    export default myDefaultExport
}
```

11.2 자바스크립트를 타입스크립트로 천천히 마이그레이션하기

타입스크립트는 태생부터 자바스크립트와 상호 운용될 수 있도록 만들어졌다.

- TSC를 프로젝트에 추가한다.

```
// tsconfig.json
{
  "compilerOptions": {
    "allowJs": true
```

```
    }
  }
```

이처럼 설정 하나만 바꾸면 TSC가 자바스크립트 파일도 컴파일한다.

- 기존 자바스크립트 코드에 타입 확인을 시작한다.

```
// tsconfig.json
{
  "compilerOptions": {
    "allowJs": true,
    "checkJs": true
  }
}
```

검사하려는 자바스크립트 파일 맨 위에 일반 주석 형태로 // @ts-check 지시어를 추가할 수도 있다. // @ts-nocheck 지시어도 있다.

- 한 번에 한 파일씩 자바스크립트를 타입스크립트로 마이그레이션한다.
- 의존하는 외부 코드용 타입 선언을 설치한다. 방법은 두 가지다. 아직 타입이 없는 외부 코드용 타입의 스텁(stub)을 만들거나, 타입 선언을 만들어서서 DefinitelType에 기여하면 된다.3
- 코드베이스에 strict 모드를 적용한다.

11.4 서드 파티 자바스크립트 사용

npm install로 서드 파티 자바스크립트 코드를 프로젝트에 설치할 때 다음처럼 세 가지 상황이 일어날 수 있다.

- 코드를 설치할 때 타입 선언이 함께 제공됨
- 코드를 설치할 때 타입 선언은 제공되지 않지만 DefinitelyTyped에서 선언을 구할 수 있음
- 코드를 설치할 때 타입 선언이 제공되지 않으며 DefinitelyTyped에서도 구할 수 없음

11.4.1 타입 선언을 포함하는 자바스크립트

NPM 패키지에 타입 선언이 내장되어 있는 몇 가지 예

```
npm install rxjs
npm install ava
npm install @angular/cli
```

11.4.2 DefinitelyTyped에서 타입 선언을 제공하는 자바스크립트

import하는 서드 파티 코드가 자체적으로 타입 선언을 포함하지 않더라도 타입스크립트 커뮤니티를 관리하는 앰비언트 모듈 선언 중앙 저장소인 DefinitelyTyped에서 타입 선언을 제공하기도 한다.

```
npm install lodash --save # Lodash 설치
npm install @types/lodash --save-dev # Lodash의 타입 선언 설치
```

11.4.3 DefinitelyTyped에서 타입 선언을 제공하지 않는 자바스크립트

세 가지 상황 중 가장 드문 상황이다. 이런 경우라면 가장 빠르지만 안전성이 떨어지는 방법에서 시작해 아주 안전하지만 시간이 많이 소비되는 방법까지 다양한 선택지가 있다.

- // @ts-ignore 지시어 추가

- 빈 타입 선언 파일을 하나 만들어서 화이트리스트를 처리할 모듈을 적어놓는다.
- 앰비언트 모듈 선언을 만든다.
- 타입 선언을 만들고 NPM에 제공한다.