TypeScript 설정 이해하기

ts compiler 는 매우 많은 설정을 가지고 있다.

현 시점에는 설정이 거의 100개에 이른다.

이 설정들은 ***command line*** 과

```
tsc --noImpLicitAny program.ts
```

`tsconfig.json`

```
{
 "compilerOptions": {
     "noImplicitAny": true
 }
}
```

에서 사용할 수 있다.

가급적 tsconfig.json 파일 권장

그래야 ts 를 어떻게 사용할 계획인지 동료나 다른 도구들이 알 수 있다.(`tsc --init`을 통해 생성)

ts 설정구성

- 소스파일의 위치 (대부분)
- 출력 종류 제어 (대부분)
- 언어 자체의 핵심 요소들을 제어하는 설정
    - 대부분의 언어에서는 허용하지 않는 고수준 설계의 설정

핵심

- `noImplicitAny`

  변수들이 미리 정의된 타입을 가져야 하는지 여부를 제어

```
function add(a, b) {
    return a + b;
}
```

위 코드는 noImplicitAny 가 해제 되어있을 때에는 유효

편집기에서 add 부분에 마우스를 올려보면, ts 가 추론한 함수의 타입을 알 수 있다.

```
function add1(a: any, b: any): any
```

any 를 코드에 명시하지 않았지만 any 타입으로 간주되므로 이를 **implicit any** 라고 부른다.

`any` 타입을 매개변수에 사용하면 타입 체커는 무력해진다.

> **Note:**
>
> `any`는 유용하지만 매우 주의해서 사용해야한다.
>
> item 5 와 3장에서 자세히 다룸

그런데 같은 코드임에도 noImplicitAny 가 설정되어있다면 오류가 된다.

이 오류들은 명시적으로 ***`: any` 라고 선언해 주거나 더 분명한 타입***을 사용하여 해결해야한다.

```
function add(a: number, b: number) {
    return a + b;
}
```

ts 는 타입 정보를 가질 때 가장 효과적이기 때문에 되도록 `noImplicitAny` 를 true 로 설정해야한다.

기존 js 를 ts 로 마이그레이션 하는 상황에만 false 로 설정하는 것을 권장

> **장점**
>
> - 가독성 증가
> - 생산성 향상

- `strictNullChecks` 기본값 true

  위 속성은 `null` **과** `undefined` **가 모든 타입에서 허용**되는지 확인하는 설정이다.

  ```
  const x: number = null;
  ```

  `strictNullChecks` 가 해제 되어있다면 위 코드는 유효하다.

  ```
  Type 'null' is not assignable to type 'number'.
  ```

  true 로 설정되어있는 경우 위와 같은 오류가 발생한다.

  `null` 대신 `undefined` 를 써도 같은 오류

  ```
  {
      strictNullChecks: true
  }
  ```

  로 설정한 경우에도

  ```
  const x: number | null = null;
  ```

  위와 같이 `null` type을 명시적으로 허용하여 사용할 수 있다.

  => 그러니 `strictNullChecks: true` 로 써라

  그러나 `strictNullChecks: true` 로 설정한 경우 코드를 작성을 어렵게 만들 수 있다.

  하지만 `null` 과 `undefined` 관련 대부분의 오류를 잡기 위해서는 설정을 권장한다.

  > `undefined`는 객체가 아닙니다.

  그렇지 않으면 위와 같은 끔찍한 런타임 오류를 보게 되고 결국 이 오류를 잡기 위해 엄격한 체크를 설정할 수밖에 없을것이다.

  프로젝트가 거대해질 수록 설정 변경은 어려워질 것이므로 가능한 초반에 설정하자

다음과 같이 사용빈도가 높은 설정들도 있지만 위 두개만큼 중요한 것은 없다.

이 모든 체크를 설정하고 싶다면 `strict` 를 설정하면 된다.

```
{
  "strictNullChecks": true, //null, undefined 타입에 이상한 짓 할시 에러내기
  "strictFunctionTypes": true, //함수파라미터 타입체크 강하게
  "strictPropertyInitialization": true, //class constructor 작성시 타입체크 강하게
  "noImplicitThis": true, //this 키워드가 any 타입일 경우 에러내기
  "alwaysStrict": true, //자바스크립트 "use strict" 모드 켜기

  "noUnusedLocals": true, //쓰지않는 지역변수 있으면 에러내기
  "noUnusedParameters": true, //쓰지않는 파라미터 있으면 에러내기
  "noImplicitReturns": true, //함수에서 return 빼먹으면 에러내기
  "noFallthroughCasesInSwitch": true, //switch문 이상하면 에러내기
}
```

tsconfig Reference

```
function add(a, b) {
    return a + b;
}
add(10, null);
```

위 코드는 타입 체크를 통과할 수 있을까