20221226_토스_타입호환성

내용 요약

- 타입 호환성(type compatibility)

  : TypeScript의 타입 호환성은 구조적 서브타이핑(structural subtyping)을 기반으로 합니다. 구조적 타이핑이란 오직 멤버만으로 타입을 관계시키는 방식입니다. 명목적 타이핑(nominal typing)과는 대조적입니다. TypeScript의 구조적 타입 시스템의 기본 규칙은 y가 최소한 x와 동일한 멤버를 가지고 있다면 x는 y와 호환된다는 것입니다.

  : 이렇듯 타입에 있어서 엄격한 타입스크립트이지만 올바른 코드라면 개발자의 의도에 맞게 유연하게 대응하는 것도 필요하다

  ⇒ 이러한 개발자의 필요에 맞춰 TypeScript 타입 시스템은 부분적으로 타입 호환을 지원함

- TypeScript의 타입 호환성 규칙

  : 개발자의 의도를 파악하여 유연함을 제공하는 것은 좋지만 과도한 자유도는 타입의 안정성을 해칠수 있다.

  : 때문에 호환 허용에 대한 명확한 규칙이 필요함

  1) 명목적 서브타이핑(nominal subtyping)

     : 타입 정의 시에 **상속관계임을 명확히 명시한 경우**에만 타입 호환을 허용

  2) 구조적 서브타이핑(structural subtyping)

     : 덕 타이핑(duck typing)이라고도 함

     : 상속 관계가 명시되어 있지 않더라도 객체의 프로퍼티를 기반으로 사용처에서 사용함에 문제가 없다면 타입 호환을 허용하는 방식

     : 명목적 서브타이핑와 동일한 효과를 내면서도 개발자의 수고를 덜어준다

     *예외조건 - 신선도(freshness)

     : fresh object를 함수에 인자로 전달한 경우 유연함에 대한 이점보다는 위험성이 크기 때문에 구조적 서브타이핑을 지원하지 않는다

     : 만약 fresh object 에 대해 타입 호환을 허용하고자 한다면

     (1)함수 매개변수 타입에 index signature를 포함시켜두어 명시적으로 타입을 허용

     (2) tsconfig상에 suppressExcessPropertyErrors를 true로 설정하면 된다

- 모든 경우에 대해 타입호환을 허용하지 않고 싶다면?

  : Branded type(=Branding type) 기법을 사용하면 된다

주요 어휘

- 타입 시스템

  : 타입 시스템은 프로그램 내에 타입을 이용해 기술된 프로그래머의 여러 의도가 말이 되는지, 서로 모순을 일으키지는 않는지 검사하는 검사원

  1) 강한 타입

  : 타입을 엄격하게 검사하여 올바르지 않은 타입 정보를 가진다면 프로그램의 실행을 막는다

  : 높은 가독성과 코드 품질을 제공한다

  2) 약한 타입

  : 변수 타입을 별도로 명시하지 않더라도 묵시적으로 형변환이 이루어진다. 자바스크립트가 대표적인 예시

  : 높은 자유도와 유연성을 보장한다는 특징을 가진다

원본 글 링크

TypeScript 타입 시스템 뜯어보기: 타입 호환성

토스 Node.js 챕터에서는 높은 코드 가독성과 품질을 위해 TypeScript의 타입 시스템을 적극적으로 활용하고 있고 이에 대한 이해도를 높이기 위해 스터디를 꾸준히 진행하고 있습니다. TypeScript

https://toss.tech/article/typescript-type-compatibility

참고자료

TypeScript: Documentation - Type Compatibility (typescriptlang.org)

타입과 타입 시스템 : 연재를 시작하며 (ahnheejong.name)

타입과 타입 시스템 : 기본 개념 (ahnheejong.name)