잉여 속성 체크의 한계 인지하기

잉여속성 체크란

잉여 속성 체크의 한계 인지하기

- 타입에 선언된 속성 외에 속성이 있는지 체크하는 것
- 객체 리터럴에서만 동작하기에 엄격한 객체 리터럴 체크라고도 불린다.
- 일반적인 구조적 할당 가능성 체크와는 역할이 다르다.

잉여속성 체크란 - 예제로 알아보기

잉여 속성 체크의 한계 인지하기

```ts
interface Person {
  name: string;
  age: number;
}

const jung: Person = {
  name: "jung",
  age: 100,
  gender: "M"
}
```

```ts
const hyun = {
  name: "hyun",
  age: 101,
  gender: "F"
}

const hoon: Person = hyun;
```

잉여 속성 체크와 할당 가능 검사는 별도의 과정이다.

잉여 속성 체크를 하지 않는 경우

잉여 속성 체크의 한계 인지하기

- 객체 리터럴이 아닌 경우
- 타입 단언문을 사용할 경우
- 인덱스 시그니처를 사용할 경우

객체 리터럴이 아닌 경우

잉여 속성 체크를 하지 않는 경우

```ts
interface Options {
  title: string;
  darkMode?: boolean;
}

function createWindow(options: Options) {
  if (options.darkMode) {
    setDarkMode();
  }
}

function setDarkMode () {}

// 객체 리터럴을 사용할 경우
const o1: Options = { darkmode: true, title: 'Ski Free' };

// 타입 구문 없는 임시 변수를 사용할 경우
const intermediate = { darkmode: true, title: 'Ski Free' };
const o2: Options = intermediate;
```

타입 단언문을 사용할 경우

잉여 속성 체크를 하지 않는 경우

```
// 타입 단언문 사용하기
const o3 = {darkmode: true, title: 'Ski Free'} as Options;
```

인덱스 시그니처를 사용할 경우

잉여 속성 체크를 하지 않는 경우

```ts
interface Options {
  darkMode?: boolean;
  [otherOptions: string]: unknown;
}

const o: Options = {darkmode: true};
```

결론

잉여 속성 체크의 한계 인지하기

- 잉여 속성 체크는 구조적 타이핑 시스템에서 허용하는 속성 이름의 오타 같은 실수를 잡는 데 효과적입니다
- 적용 범위가 매우 제한적이며 오직 객체 리터럴에만 적용됩니다