item 38

any 타입은 가능한 한 좁은 타입에서만 사용하기

채희수 / 23.07.01

“ 오늘은 any 타입을 어떻게
좁은 범위에서 사용할 수 있는지
배워볼게요. ”

1. 함수에서의 any 사용법

문제 상황

```ts
interface Foo { foo: string; }
interface Bar { bar: string; }
declare function expressionReturningFoo(): Foo;
function processBar(b: Bar) { /* ... */ }

function f() {
  const x = expressionReturningFoo();
  processBar(x);
  //         ~ Argument of type 'Foo' is not assignable to
  //           parameter of type 'Bar'
}
```

반환 타입과 매개변수 타입이 달라 오류가 발생한다.

해결책1

```ts
interface Foo { foo: string; }
interface Bar { bar: string; }
declare function expressionReturningFoo(): Foo;
function processBar(b: Bar) { /* ... */ }

function f1() {
  const x: any = expressionReturningFoo();  // Don't do this
  processBar(x);
}

function f2() {
  const x = expressionReturningFoo();
  processBar(x as any);  // Prefer this
}
```

이런 경우 any 타입으로 선언해 오류를 제거할 수 있다.
두 가지 방법 중 아래의 방법이 더 권장된다.
반환 타입에 any를 사용하면 다른 코드에도 영향을 미치기 때문이다.

?반환 타입에서 any를 사용하면?

```ts
interface Foo { foo: string; }
interface Bar { bar: string; }
declare function expressionReturningFoo(): Foo;
function processBar(b: Bar) { /* ... */ }

function f1() {
  const x: any = expressionReturningFoo();
  processBar(x);
  return x;
}

function g() {
  const foo = f1();  // Type is any
  foo.fooMethod();  // This call is unchecked!
}
```

g 함수 내에서 any 타입을 반환받아 사용하면서 foo의 타입에 영향을 준다.
이렇게 함수에서 any를 반환하면 다른 코드 혹은 프로젝트에 영향을 준다.

=> 반환 타입보다는 매개변수 타입에서 any 사용하자.
그리고 타입스크립트가 타입 추론이 가능하도록 함수의 반환 타입을 명시하자.

1. 함수에서의 any 사용법

해결책2

```
1   interface Foo { foo: string; }
2   interface Bar { bar: string; }
3   declare function expressionReturningFoo(): Foo;
4   function processBar(b: Bar) { /* ... */ }
5
6   function f1() {
7     const x = expressionReturningFoo();
8     // @ts-ignore
9     processBar(x);
10    return x;
11  }
```

충돌나는 곳에서 @ts-ignore 사용할 수도 있다.
@ts-ignore를 사용한 다음 줄의 오류가 무시된다.

그러나 근본적인 원인을 해결한 것이 아니기 때문에 다른 곳에서 더 큰 문제가 발생할 수 있다.

2. 객체에서의 any 사용법

문제 상황

```ts
interface Foo { foo: string; }
interface Bar { bar: string; }
declare function expressionReturningFoo(): Foo;
function processBar(b: Bar) { /* ... */ }

interface Config {
  a: number;
  b: number;
  c: {
    key: Foo;
  };
}
declare const value: Bar;

const config: Config = {
  a: 1,
  b: 2,
  c: {
    key: value
  // ~~~ Property ... missing in type 'Bar' but required in type 'Foo'
  }
};
```

해결책

```ts
const config: Config = {
  a: 1,
  b: 2,  // These properties are still checked
  c: {
    key: value as any
  }
};
```

as any 를 사용해 오류를 제거할 수 있다.
객체 전체를 any 로 단언하면 다른 속성들(a,b)의 타입 체크가 불가하다.
그러니 any 타입의 사용은 최소한의 범위로만 하자!

요약

- **의도치 않은 타입 안전성의 손실을 피하기 위해**
  any의 사용 범위를 최소한으로 좁혀야 한다.

- 함수의 반환 타입이 any는 절대 안 된다.
  매개변수 타입을 as any로 선언하거나 @ts-ignore을 사용하자.

- 객체 전체를 any로 단언하면 다른 속성들의 타입 체크가 되지 않으니,
  최소한의 범위에서 as any 형태를 사용하자.