item 54

객체를 순회하는 노하우

채희수 / 23.08.06

아래의 코드는 왜 오류가 날까?

```
const obj = {
  one: 'uno',
  two: 'dos',
  three: 'tres',
};
for (const k in obj) {
  const v = obj[k];
        // ~~~~~~ Element implicitly has an 'any' type
        //        because type ... has no index signature
}
```

string　'one' | 'two' | 'three'

타입을 다르게 추론해서 오류가 난다.

타입을 구체적으로 명시하면 오류가 사라진다.

```
let k: keyof typeof obj;  // Type is "one" | "two" | "three"
for (k in obj) {
  const v = obj[k];  // OK
}
```

그럼 왜 k 는 string 으로 추론될까?

```
interface ABC {
  a: string;
  b: string;
  c: number;
}
```

foo 함수의 매개변수는 a,b,c 속성 외에 다른 속성을 가진 객체로 호출이 가능하다.

```
function foo(abc: ABC) {
  for (const k in abc) {  // const k: string
    const v = abc[k];
          // ~~~~~~ Element implicitly has an 'any' type
          //        because type 'ABC' has no index signature
  }
}
const x = {a: 'a', b: 'b', c: 2, d: new Date()};
foo(x);  // OK
```

다른 속성을 가질 수 있기 때문에 키를 string 타입으로 선택한다.

아까와 같이 keyof 로 타입을 구체적으로 명시하면 오류는 해결하지만, 문제점이 있다.

```ts
interface ABC {
  a: string;
  b: string;
  c: number;
}
function foo(abc: ABC) {
  let k: keyof ABC;
  for (k in abc) {  // let k: "a" | "b" | "c"
    const v = abc[k];  // Type is string | number
  }
}

const x = {a: 'a', b: 'b', c: 2, d: new Date()};
foo(x);  // OK
```

다른 속성을 가질 수 있기 때문에 k와 v의 타입이 한정되어 런타임 동작을 예상하기 어렵다.

타입 문제 없이 객체를 순회하고 싶다면! Object.entries 를 사용하자.

```
interface ABC {
  a: string;
  b: string;
  c: number;
}
function foo(abc: ABC) {
  for (const [k, v] of Object.entries(abc)) {
    k  // Type is string
    v  // Type is any
  }
}

const x = {a: 'a', b: 'b', c: 2, d: new Date()};
foo(x);  // OK
```

요약

- 객체를 순회할 때, 키가 어떤 타입인지 정확히 파악하고 있다면
  let k: keyof T 와 for-in 루프를 사용하자.
  함수의 매개변수로 쓰이는 객체에는 추가적인 키가 존재할 수 있으니 조심.

- 객체를 순회하며 키와 값을 얻는 가장 일반적인 방법은
  Object.entries 를 사용하는 것이다.