테스팅 타입의 함정에 주의하기

이펙티브 타입스크립트

@Leemainsw

타입스크립트 타입 선언 테스트

프로젝트를 공개할 때 테스트 코드 작성은 필수
타입 선언도 테스트를 거쳐야 함

타입스크립트 타입 선언 테스트

실제 반환 타입을 체크하는 것이 더 좋은 테스트 코드

불필요한 타입 선언에 해당하지만 테스트 코드 관점에서는 중요한 역할

```
declare function map<U, V>(array: U[], fn: (u: U) => V): V[];

// 함수를 호출하는 테스트 코드
map(['2017', '2018', '2019'], v => Number(v));

// 반환 타입까지 테스트하는 코드
const lengths: number[] =
map(['2017', '2018', '2019'], v => Number(v));
```

타입스크립트 타입 선언 테스트

테스팅을 위해 할당을 사용하는 방법의
두 가지 근본적인 문제

불필요한 변수 만들기

- 반환값을 할당하는 변수는 샘플 코드처럼 쓰일 수 있지만,
일부 린트 규칙(미사용 변수 경고) 등을 비활성화해야 할 수 있다.

해결법

- 변수를 도입하는 대신 헬퍼 함수 정의하기

```
function assertType<T>(x: T) {}
assertType<number[]>(map(['john', 'paul'], name=>name.length));
```

불필요한 변수 만들기

해결법

- 변수를 도입하는 대신 헬퍼 함수 정의하기

다른 문제점들이 있다.

```
// 변수를 도입하는 대신 헬퍼 함수를 정의 하기
function assertType<T>(x: T) {};
assertType<number []>(map(['apple', 'banana'], name=> name.length));

const n = 12;
assertType<number>(n) // 정상
assertType<string>(n) // 비정상
```

```
const city = ['seoul', 'busan', 'incheon'];

const add = (a: number, b:number) => a + b;
assertType<(a: number, b: number) => number>(add); //정상

const double = (x: number) => 2 * x;
assertType<(a:number, b:number) => number>(double); // 체크X
```

동일체크가 아닌 할당 가능성 체크

- 두 타입이 동일한지 체크하는 것이 아니라, 할당 가능성을 체크하고 있다.

해결법

- Parameters와 ReturnType 제너릭 타입을 이용하여 함수의 매개변수 타입과 반환 타입만 분리하여 테스트한다.

```
const n = 12;
assertType<number>(n);  // 정상
```

```
const beatles = ['john', 'paul', 'george', 'ringo'];
assertType<{name: string}[]>(
  map(beatles, name => ({
    name,
    inYellowSubmarine: name === 'ringo'
  })));  // 정상
```

동일체크가 아닌 할당 가능성 체크

해결법

- Parameters와 ReturnType 제너릭 타입을 이용하여 함수의 매개변수 타입과 반환 타입만 분리하여 테스트한다.

```
const double = (x: number) => 2 * x;
let p: Parameters<typeof double> = null!;
assertType<[number, number]>(p);
//                           ~ '[number]' 형식의 인수는 '[number, number]'
//                             형식의 매개변수에 할당될 수 없습니다.
let r: ReturnType<typeof double> = null!;
assertType<number>(r);  // 정상
```

동일체크가 아닌 할당 가능성 체크

해결법

- 함수 세부사항 테스트하기

```
const beatles = ['john', 'paul', 'george', 'ringo'];
assertType<number[]>(map(
  beatles,
  function(name, i, array) {
// ~~~~~~~~ '(name: any, i: any, array: any) => any' 형식의 인수는
//          '(u: string) => any' 형식의 매개변수에 할당될 수 없습니다.
    assertType<string>(name);
    assertType<number>(i);
    assertType<string[]>(array);
    assertType<string[]>(this);
                        // ~~~~ 'this'에는 암시적으로 'any' 형식이 포함됩니다.
    return name.length;
  }
));
```

결론

타입을 테스트할 때는 함수 타입의 동일성과 할당 가능성의 차이점을 알고 있어야 한다.

타입 관련된 테스트에서 any를 주의해야 하며 엄격한 테스트를 위해 dtslint과 같은 도구를 사용하는 것이 좋다.