

Item 16.

number 인덱스 시그니처보다는
Array, **튜플**, **ArrayLike**를 사용하기

Effective TypeScript

목 차

1. 자바스크립트의 암묵적 타입 변환
2. 인덱스 시그니처에 number 타입 사용하기
3. 결론

O'REILLY®

Effective TypeScript

62 Specific Ways to Improve Your TypeScript



Dan Vanderkam

1. 자바스크립트의 암묵적 타입 변환

자바스크립트에서 배열 요소에 접근하기

자바스크립트에서 배열은 숫자 인덱스, 문자열 키를 통해 배열의 요소에 접근 가능



```
1  const array = [1, 2, 3];  
2  
3  // 숫자 인덱스로 array에 접근  
4  console.log(array[0]);  
5  > 1  
6  
7  // 문자열 키로 array에 접근  
8  console.log(array['0']);  
9  > 1
```

[질문] 문자열 키를 사용해 배열의 요소에 접근할 수 있는 이유?

⇒ 자바스크립트의 배열은 객체 타입이기 때문

1. 자바스크립트의 암묵적 타입 변환

자바스크립트에서 객체란?

- 키/값 쌍의 모음
- 키 : 문자열 또는 심벌 타입만 가능, 숫자는 사용 불가
- 만약 숫자로 된 키를 사용한다면?

⇒ 자바스크립트 엔진이 런타임에 **암묵적 타입 변환**을 통해 숫자 인덱스를 문자열로 변환



```
1 // Object.keys를 이용하여 배열의 키를 나열하면 키가 문자열로 출력
2 Object.keys(array);
3 > ['0', '1', '2']
```

2. 인덱스 시그니처에 number 타입 사용하기

타입스크립트는 숫자 키를 허용

Array의 타입 정의

```
1 interface Array<T> {
2     length: number;
3     toString(): string;
4     toLocaleString(): string;
5     pop(): T | undefined;
6     push(...items: T[]): number;
7     concat(...items: ConcatArray<T>[]): T[];
8     concat(...items: (T | ConcatArray<T>[]): T[]);
9     join(separator?: string): string;
10    reverse(): T[];
11    shift(): T | undefined;
12    slice(start?: number, end?: number): T[];
13    sort(compareFn?: (a: T, b: T) => number): this;
14    splice(start: number, deleteCount?: number): T[];
15    splice(start: number, deleteCount: number, ...items: T[]): T[];
16    unshift(...items: T[]): number;
17    indexOf(searchElement: T, fromIndex?: number): number;
18    lastIndexOf(searchElement: T, fromIndex?: number): number;
19    every<S extends T>(predicate: (value: T, index: number, array: T[]) => value is S, thisArg?: any): this is S[];
20    every(predicate: (value: T, index: number, array: T[]) => unknown, thisArg?: any): boolean;
21    some(predicate: (value: T, index: number, array: T[]) => unknown, thisArg?: any): boolean;
22    forEach(callbackfn: (value: T, index: number, array: T[]) => void, thisArg?: any): void;
23    map<U>(callbackfn: (value: T, index: number, array: T[]) => U, thisArg?: any): U[];
24    filter<S extends T>(predicate: (value: T, index: number, array: T[]) => value is S, thisArg?: any): S[];
25    filter(predicate: (value: T, index: number, array: T[]) => unknown, thisArg?: any): T[];
26    reduce(callbackfn: (previousValue: T, currentValue: T, currentIndex: number, array: T[]) => T): T;
27    reduce(callbackfn: (previousValue: T, currentValue: T, currentIndex: number, array: T[]) => T, initialValue: T): T;
28    reduce<U>(callbackfn: (previousValue: U, currentValue: T, currentIndex: number, array: T[]) => U, initialValue: U): U;
29    reduceRight(callbackfn: (previousValue: T, currentValue: T, currentIndex: number, array: T[]) => T): T;
30    reduceRight(callbackfn: (previousValue: T, currentValue: T, currentIndex: number, array: T[]) => T, initialValue: T): T;
31    reduceRight<U>(callbackfn: (previousValue: U, currentValue: T, currentIndex: number, array: T[]) => U, initialValue: U): U;
32
33    [n: number]: T;
34 }
```

⇒ 숫자 타입의 키 허용

타입 체크 시점에 오류 탐지

But!

실제로는 런타임에

- 타입스크립트의 타입 제거됨
- 키는 문자열로 암묵적 타입 변환

2. 인덱스 시그니처에 number 타입 사용하기

number 타입 인덱스 시그니처

- 인덱스 시그니처가 숫자 타입으로 되어있더라도 실제 런타임에 사용되는 키는 문자열
- 만약 숫자를 인덱스 타입으로 사용해야 한다면?
 - ⇒ 이미 정의되어 있는 Array, 튜플, ArrayLike 타입을 사용

[Array와 ArrayLike 타입의 차이점]

- Array 타입은 배열 메서드를 사용 가능
- ArrayLike 타입은 배열 메서드 사용 불가
 - ⇒ 배열 메서드가 필요하지 않은 경우 ArrayLike 사용

```
1 interface ArrayLike<T> {  
2     readonly length: number;  
3     readonly [n: number]: T;  
4 }
```

3. 결론

자바스크립트의 배열은 객체이므로 키는 숫자가 아닌 문자열입니다.

인덱스 시그니처로 사용된 숫자 타입은 타입 체크 시점에 버그를 잡기 위한 순수 타입스크립트 코드입니다. 그러므로 실제로 런타임에는 키가 문자열로 변환된다는 것을 이해하고 사용해야 합니다.

숫자를 인덱스 타입으로 사용한다면 숫자 속성이 특별한 의미를 가진다는 오해가 발생할 수 있으므로 인덱스 시그니처에 숫자 타입을 사용하는 것보다 Array, 튜플, ArrayLike 타입을 사용하는 것이 좋습니다.

감사합니다 :))

Effective TypeScript

@Bori-github(이보리)