아이템 22

# 타입 좁히기

채희수 / 23.05.21

# 타입 좁히기

**타입스크립트가 넓은 타입으로부터 좁은 타입으로 진행하는 과정**

- null 체크
- 분기문에서 예외를 던지거나 함수를 반환하여 블록의 나머지 부분에서 변수의 타입을 좁힐 수 있다.
- instanceof 사용
- 속성 체크
- Array.isArray 같은 일부 내장 함수
- 명시적 '태그' 붙이기 = '태그된 유니온' '구별된 유니온'
- 식별을 돕기 위한 커스텀 함수 사용자 정의 타입 가드

# 설명하기 좋게 나눴습니다.

- null 체크 // HTMLElement
- 분기문에서 예외를 던지거나 함수를 반환하여 블록의 나머지 부분에서 변수의 타입을 좁힐 수 있다.
- 식별을 돕기 위한 커스텀 함수 사용자 정의 타입 가드

- 속성 체크 // ‘a’ in ab
- 명시적 ‘태그’ 붙이기 = ‘태그된 유니온‘ ‘구별된 유니온‘ // e.type

- instanceof 사용 // RegExp
- Array.isArray 같은 일부 내장 함수 // Array

# HTMLElement은 타입이 HTMLElement | null 이다.

```
const el = document.getElementById('foo'); // Type is HTMLElement | null
if (!el) throw new Error('Unable to find #foo');
el; // Now type is HTMLElement
el.innerHTML = 'Party Time'.blink();
```

분기문에서 예외를 던지거나 함수를 변환하여
블록의 나머지 부분에서 변수의 타입을 좁히기

```
function isInputElement(el: HTMLElement): el is HTMLInputElement {
  return 'value' in el;
}

function getElementContent(el: HTMLElement) {
  if (isInputElement(el)) {
    el; // Type is HTMLInputElement
    return el.value;
  }
  el; // Type is HTMLElement
  return el.textContent;
}
```

식별을 돕기 위한 커스텀 함수
‘ 사용자 정의 타입 가드 ’

# type/interface 안에 정의되어 있는가

```ts
interface A { a: number }
interface B { b: number }
function pickAB(ab: A | B) {
  if ('a' in ab) {
    ab // Type is A
  } else {
    ab // Type is B
  }
  ab // Type is A | B
}
```

속성 체크로

```ts
interface UploadEvent { type: 'upload'; filename: string; contents: string }
interface DownloadEvent { type: 'download'; filename: string; }
type AppEvent = UploadEvent | DownloadEvent;

function handleEvent(e: AppEvent) {
  switch (e.type) {
    case 'download':
      e  // Type is DownloadEvent
      break;
    case 'upload':
      e;  // Type is UploadEvent
      break;
  }
}
```

명시적 ‘ 태그 ’ 붙이기
= 태그된 유니온 / 구별된 유니온

# RegExp / Array 자료형별 맞춤 분기처리

```
function contains(text: string, search: string|RegExp) {
  if (search instanceof RegExp) {
    search  // Type is RegExp
    return !!search.exec(text);
  }
  search  // Type is string
  return text.includes(search);
}
```

instanceof 사용

Array.isArray 내장 함수 사용

```
function contains(text: string, terms: string|string[]) {
  const termList = Array.isArray(terms) ? terms : [terms];
  termList // Type is string[]
  // ...
}
```

# 타입을 좁힐 때 주의사항

- typeof null === 'object'

```
const el = document.getElementById('foo'); // type is HTMLElement | null
if (typeof el === 'object') {
  el;  // Type is HTMLElement | null
}
```

- 빈 문자열 '' 과 0 모두 false

```
function foo(x?: number|string|null) {
  if (!x) {
    x;  // Type is string | number | null | undefined
  }
}
```

# 요약

- 분기문 외에도 여러 종류의 제어 흐름을 살펴보며
  타입스크립트가 타입을 좁히는 과정을 이해해야 한다.
  -> 타입 추론에 대한 개념을 잡을 수 있고,
  오류 발생의 원인을 알 수 있고,
  타입 체커를 더 효율적으로 이용할 수 있다.

- 태그된/구별된 유니온과 사용자 정의 타입 가드를 사용하여
  타입 좁히기 과정을 원활하게 만들 수 있다.