

item 34

부정확한 타입보다는 미완성 타입을 사용하기

일반적으로 타입이 구체적일수록 버그를 더 많이 잡고 타입스크립트에서 제공하는 도구를 활용할 수 있다.

**“ 그러나 타입 선언의 정밀도를 높이는 일엔 주의해야 한다.
실수가 발생하기 쉽고,
잘못된 타입은 차라리 없는 것보다 못할 수 있기 때문이다.”**

구체적인 예시를 통해 알아보까요?

아래의 코드에서 타입을 공통적으로 선언해 볼게요.

```
1 interface Point {
2   type: 'Point';
3   coordinates: number[];
4 }
5 interface LineString {
6   type: 'LineString';
7   coordinates: number[][];
8 }
9 interface Polygon {
10  type: 'Polygon';
11  coordinates: number[][][];
12 }
13 type Geometry = Point | LineString | Polygon; // Also several others
```

Coordinate : 좌표, 경도와 위도 나타내는 값
=> 튜플로 선언하자.

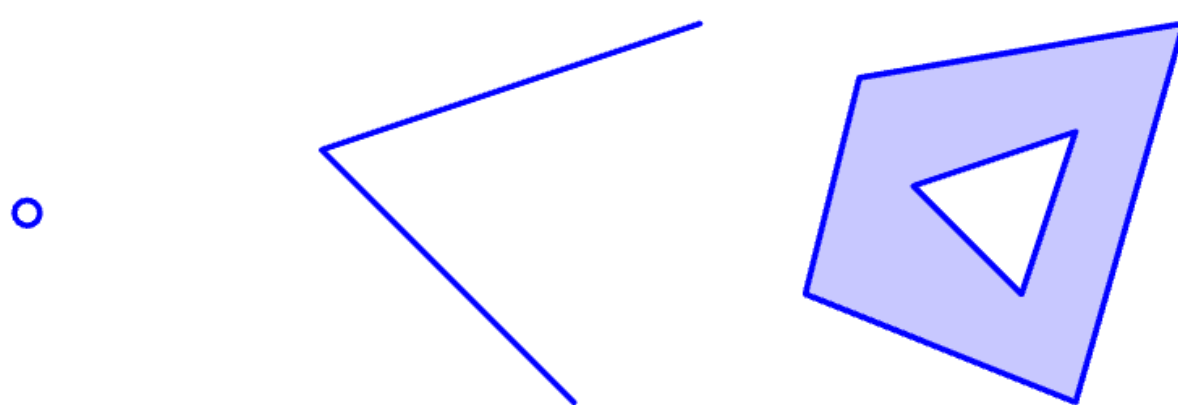
```
1 type GeoPosition = [number, number];
2 interface Point {
3   type: 'Point';
4   coordinates: GeoPosition;
5 }
```

WRONG! 경도와 위도 말고 고도나 다른 정보가 있을 수 있다.
사용하려면 타입 단언문이나 as any를 추가해야 한다.

Point

LineString

Polygon



| Type | WKT example |
|--------------|--|
| "Point" | POINT (30 10) |
| "LineString" | LINESTRING (30 10, 10 30, 40 40) |
| "Polygon" | POLYGON ((35 10, 45 45, 15 40, 10 20, 35 10),(20 30, 35 35, 30 20, 20 30)) |

출처 : <https://geobgu.xyz/py/shapely.html>

이번에는 아래의 값을 타입 선언해 볼게요

```
[
  10,
  "red",
  true,
  ["+", 10, 5],
  ["case", [ ">", 20, 10 ], "red", "blue", "green"],
  ["**", 2, 31],
  ["rgb", 255, 128, 64],
  ["rgb", 255, 128, 64, 73]
];
```

- 맵박스 라이브러리에서 사용하는 입력값
- 모델링해 볼 수 있는 입력값의 전체 종류

1. 모두 허용
2. 문자열, 숫자, 배열 허용
3. 문자열, 숫자, 알려진 함수 이름으로 시작하는 배열 허용
4. 각 함수가 받는 매개변수의 개수가 정확한지 확인
5. 각 함수가 받는 매개변수의 타입이 정확한지 확인



```
1 // 1. 모두 허용
2 type Expression1 = any;
3 // 2. 문자열, 숫자, 배열 허용
4 type Expression2 = number | string | any[];
5
6 const tests: Expression2[] = [
7   10,
8   "red",
9   true,
10  // ~~~ Type 'true' is not assignable to type 'Expression2'
11  ["+", 10, 5],
12  ["case", [ ">", 20, 10 ], "red", "blue", "green"], // Too many values
13  ["**", 2, 31], // Should be an error: no "**" function
14  ["rgb", 255, 128, 64],
15  ["rgb", 255, 0, 127, 0] // Too many values
16 ];
```

```
1 // 3. 문자열, 숫자, 알려진 함수 이름으로 시작하는 배열 허용
2 type FnName = '+' | '-' | '*' | '/' | '>' | '<' | 'case' | 'rgb';
3 type CallExpression = [FnName, ...any[]];
4 type Expression3 = number | string | CallExpression;
5
6 const tests: Expression3[] = [
7   10,
8   "red",
9   true,
10  // ~~~ Type 'true' is not assignable to type 'Expression3'
11  ["+", 10, 5],
12  ["case", [">", 20, 10], "red", "blue", "green"],
13  [**, 2, 31],
14  // ~~~~~~ Type '**' is not assignable to type 'FnName'
15  ["rgb", 255, 128, 64]
16 ];
```

```

1 // 4. 각 함수가 받는 매개변수의 개수가 정확한지 확인
2 // 5. 각 함수가 받는 매개변수의 타입이 정확한지 확인
3 type Expression4 = number | string | CallExpression;
4
5 type CallExpression = MathCall | CaseCall | RGBCall;
6
7 interface MathCall {
8   0: '+' | '-' | '/' | '*' | '>' | '<';
9   1: Expression4;
10  2: Expression4;
11  length: 3;
12 }
13
14 interface CaseCall {
15   0: 'case';
16   1: Expression4;
17   2: Expression4;
18   3: Expression4;
19   length: 4 | 6 | 8 | 10 | 12 | 14 | 16 // etc.
20 }
21
22 interface RGBCall {
23   0: 'rgb';
24   1: Expression4;
25   2: Expression4;
26   3: Expression4;
27   length: 4;
28 }

```

```

29 const tests: Expression4[] = [
30   10,
31   "red",
32   true,
33   // ~~~ Type 'true' is not assignable to type 'Expression4'
34   ["+", 10, 5],
35   ["case", [ ">", 20, 10 ], "red", "blue", "green"],
36   // ~~~~~
37   // Type '["case", [ ">", ... ], ...]' is not assignable to type 'string'
38   ["**", 2, 31],
39   // ~~~~~ Type '["**", number, number]' is not assignable to type 'string'
40   ["rgb", 255, 128, 64],
41   ["rgb", 255, 128, 64, 73]
42   // ~~~~~ Type '["rgb", number, number, number, number]'
43   // is not assignable to type 'string'
44 ];
45

```



```

1 // 4. 각 함수가 받는 매개변수의 개수가 정확한지 확인
2 // 5. 각 함수가 받는 매개변수의 타입이 정확한지 확인
3 type Expression4 = number | string | CallExpression;
4
5 type CallExpression = MathCall | CaseCall | RGBCall;
6
7 interface MathCall {
8     0: '+' | '-' | '/' | '*' | '>' | '<';
9     1: Expression4;
10    2: Expression4;
11    length: 3;
12 }
13
14 interface CaseCall {
15     0: 'case';
16     1: Expression4;
17     2: Expression4;
18     3: Expression4;
19     length: 4 | 6 | 8 | 10 | 12 | 14 | 16 // etc.
20 }
21
22 interface RGBCall {
23     0: 'rgb';
24     1: Expression4;
25     2: Expression4;
26     3: Expression4;
27     length: 4;
28 }

```



만약 FnName에 따라 받을 수 있는 매개변수 수가 달라진다면? 오히려 타입이 너무 복잡해진다.

```

46 const okExpressions: Expression4[] = [
47     ['-', 12],
48     // ~~~~~ Type '["-", number]' is not assignable to type 'string'
49     ['+', 1, 2, 3],
50     // ~~~~~ Type '["+ ", number, ...]' is not assignable to type 'string'
51     ['*', 2, 3, 4],
52     // ~~~~~ Type '["*", number, ...]' is not assignable to type 'string'
53 ];

```

요약

- 타입이 없는 것보다 잘못된 게 더 나쁘다. 정확하게 타입을 모델링할 수 없다면, 부정확하게 모델링하지 말자.
- any와 unknown을 구별해서 사용하자.

| any | Unknown |
|---|---|
| 타입 검사를 우회해 어떤 타입이든 할당할 수 있다. 타입을 알 수 없는 외부 라이브러리나 타입이 일시적으로 필요하지 않는 상황에서 사용 (가능한 any 사용을 피하자) | 타입을 알 수 없는 경우, 동적인 타입 체크가 필요한 경우 사용 any 보다 안전한 타입 체크 unknown을 사용하려면 먼저 해당 값을 검사하고 타입을 좁혀야 한다. |

- 타입 정보를 구체적으로 만들수록 오류 메시지와 자동 완성 기능에 주의를 기울여야 한다.