Item 41. any의 진화 이해하기

'23. 02. 18. (SAT) 신현호(@SWARVY)

- TS에서 일반적으로 타입은 변수를 선언할 때 결정된다.
  이후에는 정제를 수는 있지만 (ex. null check ...),
  새로운 값이 추가되도록 확장은 불가

Ex 1)

type number

number타입에서 특정 값을 걸러낸다던지..

Ex 2)

type string

string타입에서 undefined를 걸러낸다던가..

- 물론 당연히 예외가 존재한다. any에서는 예외사항이 존재.

```
function range(start, limit) {
  const out = [];
  for (let i = start; i < limit; i++) {
    out.push(i);
  }
  return out;
}
```

동작은 잘되는데...?

```
function range(start: number, limit: number) {
  const out = [];
  for (let i = start; i < limit; i++) {
    out.push(i);
  }
  return out;  // 반환 타입이 number[]로 추론됨.
}
```

const out = []; → any 타입 []로 초기화.

↳추론은 number[];

중요.

- out의 타입은 any[]였지만 number 타입의 값을 넣는순간부터 타입은 number[]로 진화한다.

any는 이브이마냥 진화하게된다

```
const result = [];  // 타입이 any[]
result.push('a');
result              // 타입이 string[]
result.push(1);
result              // 타입이 (string | number)[]
```

- 타입의 진화는 타입 좁히기와는 다르다. 배열에 다양한 타입의 요소를 넣으면 배열의 타입이 확장되며 진화한다.

```
let val;   // 타입이 any
if (Math.random() < 0.5) {
  val = /hello/;
  val      // 타입이 RegExp
} else {
  val = 12;
  val      // 타입이 number
}
val        // 타입이 number | RegExp
```

- 조건문 분기에 따라 타입이 변할 수도 있다.

```
let val = null;  // 타입이 any
try {
  somethingDangerous();
  val = 12;
  val            // 타입이 number
} catch (e) {
  console.warn('alas!');
}
val              // 타입이 number | null
```

- 변수의 초기값이 null인 경우도 any의 장점이 사라진다.
  보통은 try/catch에서 쓸 수 있다.

쓰고싶다면 noImplicitAny 설정하세요.

any 타입의 전파는 noImplicitAny가 설정된 상태에서 변수의 암시적 타입이 any일 경우에만 일어난다.

```
let val: any;  // 타입이 any
if (Math.random() < 0.5) {
  val = /hello/;
  val          // 타입이 any
} else {
  val = 12;
  val          // 타입이 any
}
val            // 타입이 any
```

← 이경우로 명시적 any를 지정해도 안좋다

```ts
function range(start: number, limit: number) {
  const out = [];
  //    ~~~ 'out' 변수는 형식을 확인할 수 없는 경우
  //        일부 위치에서 암시적으로 'any[]' 형식입니다.
  if (start === limit) {
    return out;
    //     ~~~ 'out' 변수에는 암시적으로 'any[]' 형식이 포함됩니다.
  }
  for (let i = start; i < limit; i++) {
    out.push(i);
  }
  return out;
}
```

any인 상태에서 바로 사용하려하면 암시적 any오류 발생.

막 쓸수 있는건 아니고,

추가할 때만 진화하는거라서 그냥 쓰면 오류

```ts
function makeSquares(start: number, limit: number) {
  const out = [];
    // ~~~ 'out' 변수는 일부 위치에서 암시적으로 'any[]' 형식입니다.
  range(start, limit).forEach(i => {
    out.push(i * i);
  });
  return out;
    // ~~~ 'out' 변수에는 암시적으로 'any[]' 형식이 포함됩니다.
}
```

함수로는 이브이가 진화하지 않음.

결론

Any는 포켓몬의 이브이처럼 타입의 진화가 가능하다 (단, 값을 추가할 때만, 함수로 추가하는거로는 안된다)

하지만, 타입이 진화한다는건 타입의 안정성이 떨어진다는 말이기에, 그냥 명시적 타입 지정을 쓰는걸 권장.

Item 41. any의 진화를 이해하기

- TS에서 일반적으로 타입은 변수를 선언할 때 결정된다. 이후에는 정제를 할 수 있지만 (ex. null check ...), 새로운 값이 추가되듯 확장은 불가

- 물론 당연히 예외가 존재한다. any에서는 예외사항이 존재.

```
function range(start, limit) {
  const out = [];
  for (let i = start; i < limit; i++) {
    out.push(i);
  }
  return out;
}
```

좌우는 잘 되는데 ...?

```
function range(start: number, limit: number) {
  const out = [];  ⟶ any 타입 []로 초기화
  for (let i = start; i < limit; i++) {
    out.push(i);
  }
  return out;  // 반환 타입이 number[]로 추론됨.
}
```

↳ 추론은 number[];

- out의 타입은 any[]였지만 number 타입의 값을 넣는순간부터 타입은 number[]로 진화한다.

```
const result = [];  // 타입이 any[]
result.push('a');
result              // 타입이 string[]
result.push(1);
result              // 타입이 (string | number)[]
```

- 타입의 진화는 타입 좁히기와는 다르다. 배열에 다양한 타입의 요소를 넣으면 배열의 타입이 확장되며 진화한다.

```
let val;   // 타입이 any
if (Math.random() < 0.5) {
  val = /hello/;
  val      // 타입이 RegExp
} else {
  val = 12;
  val      // 타입이 number
}
val        // 타입이 number | RegExp
```

- 조건문 분기에 따라 타입이 변할 수도 있다.

```
let val = null;  // 타입이 any
try {
  somethingDangerous();
  val = 12;
  val            // 타입이 number
} catch (e) {
  console.warn('alas!');
}
val              // 타입이 number | null
```

- 변수의 초기값이 null인 경우도 any의 진화가 일어난다. 보통은 try/catch에서 볼 수 있다.

※ any 타입의 진화는 noImplicitAny가 설정된 상태에서 변수의 암시적 타입이 any일 경우에만 일어난다.

```
let val: any;  // 타입이 any
if (Math.random() < 0.5) {
  val = /hello/;
  val          // 타입이 any
} else {
  val = 12;
  val          // 타입이 any
}
val            // 타입이 any
```

← 이경우로 명시적 any를 지정해도 안좋다

```
function range(start: number, limit: number) {
  const out = [];
  //    ~~~ 'out' 변수는 형식을 확인할 수 없는 경우
  //        일부 위치에서 암시적으로 'any[]' 형식입니다.
  if (start === limit) {
    return out;
    //     ~~~ 'out' 변수에는 암시적으로 'any[]' 형식이 포함됩니다.
  }
  for (let i = start; i < limit; i++) {
    out.push(i);
  }
  return out;
}
```

any인 상태에서 바로 사용하려하면 암시적 any오류 발생.

```
function makeSquares(start: number, limit: number) {
  const out = [];
    // ~~~ 'out' 변수는 일부 위치에서 암시적으로 'any[]' 형식입니다.
  range(start, limit).forEach(i => {
    out.push(i * i);  ⟶ 함수콜백코드는 진화 안함.
  });
  return out;
    // ~~~ 'out' 변수에는 암시적으로 'any[]' 형식이 포함됩니다.
}
```

↳ any 타입의 진화는 any타입에 어떤 값을 할당할 때만 발생.

근데 그냥 명시적 타입 지정 쓰세요.