아이템 52 테스팅 타입의 함정에 주의하기

아이템 52 테스팅 타입의 함정에 주의하기

```
test('square a number', () =>{
    square(1);
    square(2);
});
```

```
const lengths: number[] = map(['john', 'paul', name => name.length]);
```

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

아이템 52 테스팅 타입의 함정에 주의하기

lodash map(array, (name, index, array) => {});

```
const double = (x:number)=> 2 * x;

let p: Parameters<typeof double> = null;
assertType<[number, number]> (p);

let r: ReturnType<typeof double> = null;
assertType<number>(r);
```

아이템 52 테스팅 타입의 함정에 주의하기

```
const beatles = ['john','paul','george','ringo'];
assertType<number[]>(map(
    beatles,
    function(name,i,array){
        assertType<string>(name);
        assertType<number>(i);
        assertType<string[ ]>(array);
        assertType<string[ ]>(this);
        return name.length;
    }
));

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


declare module 'overbar';
```

아이템 52 테스팅 타입의 함정에 주의하기

```
const beatles = ['john','paul','george','ringo'];
assertType<number[]>(map(
  beatles,
  function(name,i,array){
    assertType<string>(name);
    assertType<number>(i);
    assertType<string[ ]>(array);
    assertType<string[ ]>(this);
    return name.length;
  }
));
```

dtslint

```
const beatles = ['john','paul','george','ringo'];
map(beatles, function (
    name,  //$ExpectType string
    i,     //$ExpectType number
    array  //$ExpectType string[]
){
    this   //$ExpectType string[]
    return name.length;
});        //$ExpectType number[]
```

string | number

number | string

7장 코드를 작성하고 실행하기

아이템53 타입스크립트 기능보다 ECMAScript 기능 사용하기

1. 열거형 enum

- 숫자 열거형에 0,1,2 외 다른 숫자가 할당되면 매우 위험하다.
- 상수 열거형은 보통의 열거형과 달리 런타임에 완전히 제거된다.
  문자열 열거형과 숫자 열거형은 완전히 다른 동작이다.
- preserveConstEnums 플래그를 설정한 상태의 상수 열거형은 런타임 코드에 상수 열거형 정보를 유지한다.
- 문자열 열거형은 런타임의 타입 안정성과 투명성을 제공하지만 타입스크립트의 다른 타입과 달리 구조적 타이핑이 아닌 명목적 타이핑을 사용한다.

*구조적 타이핑은 구조가 같으면 할당이 허용되지만 명목적 타이핑은 타입 이름이 같아야 할당이 허용된다.

아이템53 타입스크립트 기능보다 ECMAScript 기능 사용하기

```
enum Flavor {
    VANILLA = 'vanilla',
    CHOCOLATE = 'chocolate',
    STRAWBERRY = 'strawberry',
}
let flavor = Flavor.CHOCOLATE;
flavor = 'strawberry';
// Type '"strawberry"' is not assignable to type 'Flavor'.


function scoop(flavor: Flavor) {

}

scoop('vanilla'); //자바스크립트에서는 정상

import {Flavor} from 'ice-cream';
scoop(Flavor.VANILLA);
```

아이템53 타입스크립트 기능보다 ECMAScript 기능 사용하기

```
type Flavor = 'vanilla' | 'chocolate' | 'strawberry';

let flavor: Flavor = 'chocolate';
    flavor = 'mint chip';
```

```
type Flavor = 'vanilla' | 'chocolate' | 'strawberry';

let flavor: Flavor = 'chocolate';
    flavot = 'mint chip';

function scoop(flavor: Flavor){
    if(flavor === 'v')
}
```

vanilla

2. 매개변수 속성

```
class Person1 {
    name: string;
    constructor(name:string){
        this.name = name;
    }
}


class Person2 {
    constructor(public name : string) {

    }
}
```

public name -> 매개변수 속성

2. 매개변수 속성

- 일반적으로 타입스크립트 컴파일은 타입제거가 이루어지므로 코드가 줄어들지만 매개변수 속성은 코드가 늘어나는 문법이다.
- 매개변수 속성이 런타임에는 실제로 사용되나 타입스크립트 관점에서는 사용되지 않는 것처럼 보인다.
- 매개변수 속성과 일반 속성을 섞어 사용하면 클래스 설계가 혼란스러워진다.

2. 매개변수 속성

```
class Person1 {
    first: string;
    last: string;
    constructor(public name: string){
        [this.first, this.last] = name.split(' ');
    }
}
```

```
class Person1 {
    constructor(public name: string){}
}
const p: Person1 = {name: 'Jed Bartlet'};
```

3. Namespace 와 /// import

4. Decorater 데코레이터

@logged

7장 코드를 작성하고 실행하기

아이템54 객체를 순회하는 노하우

아이템 54 객체를 순회하는 노하우

```
const obj = {
    one: 'uno',
    two: 'dos',
    three: 'tres',
};
for (const k in obj){
    const v = obj[k];
    //obj 내에 인덱스 시그니처가 없기 때문에
    //엘리먼트는 암시적으로 any 타입입니다.
}
```

```
let k: keyof typeof obj;
for(k in obj){
    const v = obj[k];
}
```

아이템 54 객체를 순회하는 노하우

```ts
interface ABC{
    a: string;
    b: string;
    c: number;
}
function foo(abc: ABC){
    for (const k in abc){
        const v = abc[k];
    }
}

const x = {a: 'a', b: 'b', c: 'c', d: new Date()};
foo(x);
```

```ts
function foo(abc: ABC){
    let k: keyof ABC;
    for ( k in abc){
        const v = abc[k]; //v의 타입은 string | number
    }
}
```

아이템 54 객체를 순회하는 노하우

```
Object.entries

let k: keyof T
```