아이템37 공식 명칭에는 상표 붙이기

아이템37. 공식 명칭에는 상표 붙이기

```ts
interface Vector2D {
    x: number;
    y: number;
}
function calculateNorm(p: Vector2D) {
    return Math.sqrt(p.x*p.x+p.y*p.y);
}

calculateNorm({x:3, y:4});
const vec3D = {x:3, y:4, z:1};
calculateNorm(vec3D);
```

```ts
interface Vector2D{
    _brand: '2d';
    x: number;
    y: number;
}

function vec2D(x: number, y: number): Vector2D {
    return {x, y, _brand: '2d'};
}
function calculateNorm(p: Vector2D) {
    return Math.sqrt(p.x*p.x+p.y*p.y);
}

calculateNorm(vec2D(3,4));
const vec3D = {x:3, y:4, z:1};
calculateNorm(vec3D);
```

아이템37. 공식 명칭에는 상표 붙이기

```
type AbsolutePath = string & {_brand: 'abs'};
function listAbsolutePath(path: AbsolutePath){

}
function isAbsolutePath(path: String): path is AbsolutePath{
    return path.startsWith('/'); -> 절대경로
}


function f(path: String){
    if (isAbsolutePath(path)){
        isAbsolutePath(path);
    }
    listAbsolutePath(path);

}
```

아이템37. 공식 명칭에는 상표 붙이기

```ts
function binarySerach<T>(xs: T[], x: T): boolean{
    let low = 0, high = xs.length - 1;
    while (high >= low) {
        const mid = low + Math.floor((high - low) / 2);
        const v = xs[mid];
        if (v === x) return true;
        [low, high] = x > v ? [mid + 1, high] : [low, mid - 1];
    }
    return false;
}
```

```ts
type SortedLlist<T> = T[] & {_brand: 'sorted'};
function isSorted<T>(xs: T[]) : xs is SortedLlist<T> {
    for (let i = 1; i< xs.length; i++){
        if (xs[i] < xs[i-1]) {
            return false;
        }
    }
    return true;
}
```

아이템37. 공식 명칭에는 상표 붙이기

```ts
type Meters = number & {_brand: 'meters'};
type Seconds = number & {_brand: 'seconds'};

const meters = (m: number) => m as Meters;
const seconds = (s: number) => s as Seconds;

const oneKm = meters(1000);
const oneMin = seconds(60);


const tenKm = oneKm * 10;
const b = oneKm / oneMin ;
```

아이템38 any 타입은 가능한 좁은 범위에만 사용하기

아이템38. any 타입은 가능한 좁은 범위에만 사용하기

```
function processBar(b: Bar) { }

function f(){
    const x = expressionReturningFoo();
    processBar(x);
}

function f1(){
    const x: any = expressionReturningFoo();
    processBar(x);
}

function f2(){
    const x = expressionReturningFoo();
    processBar(x as any);
}
```

아이템38. any 타입은 가능한 좁은 범위에만 사용하기

@ts-ignore

```
function f1(){
    const x: any = expressionReturningFoo();
    // @ts-ignore
    processBar(x);
    return x;
}
```

아이템38. any 타입은 가능한 좁은 범위에만 사용하기

- 타입 안정성 손실을 피하기 위해 any 사용 범위를 최소한으로 좁혀야 한다.
- 함수의 반환 타입이 any인 경우 타입 안정성이 나빠진다.
  따라서 any 타입을 반환하면 절대 안된다.
- 강제로 타입오류를 제거하려면 any 대신 @ts-ignore 사용하는 것이 좋다.

아이템39 any를 구체적으로 변형해서 사용하기

아이템39. any를 구체적으로 변형해서 사용하기

```
function getLengthBad(array: any){
    return array.length
}
```

```
function getLengt(array: any[] ){
    return array.length
}
```

```
getLengthBad(/123/);
getLengt(/123/);
```

아이템39. any를 구체적으로 변형해서 사용하기

```ts
type Fn0 = () => any;
type Fn1 = (arg: any) => any;
type FnN = (...args: any[]) => any;

const numArgsBad = (...args: any) => args.length;
const numArgsGood = (...args: any[]) => args.length;
```