

12.6 TypeScript - Type Alias

타입 앨리어스



타입 앨리어스는 새로운 타입을 정의한다. 타입으로 사용할 수 있다는 점에서 타입 앨리어스는 인터페이스와 유사하다.

인터페이스는 아래와 같이 타입으로 사용할 수 있다.

TYPESCRIPT

```
interface Person {  
  name: string,  
  age?: number  
}  
  
// 빈 객체를 Person 타입으로 지정  
const person = {} as Person;  
person.name = 'Lee';
```

```
person.age = 20;  
person.address = 'Seoul'; // Error
```

타입 앨리어스도 인터페이스와 마찬가지로 타입으로 사용할 수 있다.

TYPESCRIPT

```
// 타입 앨리어스  
type Person = {  
  name: string,  
  age?: number  
}  
  
// 빈 객체를 Person 타입으로 지정  
const person = {} as Person;  
person.name = 'Lee';  
person.age = 20;  
person.address = 'Seoul'; // Error
```

하지만 타입 앨리어스는 원시값, 유니온 타입, 튜플 등도 타입으로 지정할 수 있다.

TYPESCRIPT

```
// 문자열 리터럴로 타입 지정  
type Str = 'Lee';  
  
// 유니온 타입으로 타입 지정  
type Union = string | null;  
  
// 문자열 유니온 타입으로 타입 지정  
type Name = 'Lee' | 'Kim';  
  
// 숫자 리터럴 유니온 타입으로 타입 지정  
type Num = 1 | 2 | 3 | 4 | 5;  
  
// 객체 리터럴 유니온 타입으로 타입 지정  
type Obj = {a: 1} | {b: 2};  
  
// 함수 유니온 타입으로 타입 지정  
type Func = (() => string) | (() => void);
```

```
// 인터페이스 유니온 타입으로 타입 지정  
type Shape = Square | Rectangle | Circle;  
  
// 튜플로 타입 지정  
type Tuple = [string, boolean];  
const t: Tuple = ['', '']; // Error
```

인터페이스는 extends 또는 implements될 수 있지만 타입 앨리어스는 extends 또는 implements될 수 없다. 즉, 상속을 통해 확장이 필요하다면 타입 앨리어스보다는 인터페이스가 유리하다. 하지만 인터페이스로 표현할 수 없거나 유니온 또는 튜플을 사용해야한다면 타입 앨리어스를 사용하는 편이 유리하다.