

아이템 15

동적 데이터에 인덱스 시그니처 사용하기

채희수 / 23.05.07

동적 데이터?

프로그램 실행 중에 생성되고 변경될 수 있는 데이터

- 서버에서 실시간으로 받아온 데이터 -> 프론트에서 변환해서 사용
- 사용자가 양식을 작성할 때, 입력된 정보

동적 데이터의 타입을 유연하게 표현하려면 **인덱스 시그니처!**

- 완전 간단하게 key-value의 타입 지정

```
type Study = {[property:string]:string}
const study : Study = {
  name: 'effectiveTS',
  frequency: 'weekly',
}
```

```
type Study = {[property:string]:string | number | undefined}
const study : Study = {
  name: 'effectiveTS',
  frequency: 'weekly',
  numberOfMembers: 6,
  date: undefined,
}
```

여러 타입이 올 수 있다면 유니온타입 사용

그런데,, 인덱스 시그니처의 단점이 있었으니,,

- 완전 간단하게 key-value의 타입 지정

```
type Study = {[property:string]:string}
const study : Study = {
  name: 'effectiveTS',
  frequency: 'weekly',
}

type Study = {[property:string]:string | number | undefined}
const study : Study = {
  name: 'effectiveTS',
  frequency: 'weekly',
  numberOfMembers: 6,
  date: undefined,
}
```

단점)

- key에 오타가 있는 경우 ide가 오류 표시를 안함
- key마다 특정 타입 체크가 필요한 경우

인덱스 시그니처보다 더 나은 방법은 인터페이스

```
interface Study {  
  name: string;  
  frequency: string;  
  numberOfMembers: number;  
  date: number | undefined;  
}  
  
const study : Study = {  
  name: 'effectiveTS',  
  frequency: 'weekly',  
  numberOfMembers: 6,  
  date: 1683455400  
}
```

interface Study { 필드 뒤에 ? 을 붙이면 선택적 속성임을 나타낸다.

```
  name: string;  
  frequency: string;  
  numberOfMembers?: number;  
  date?: number;  
}  
  
const study : Study = { ?이 붙은 필드는 생략가능하다.  
  name: 'effectiveTS',  
  frequency: 'weekly'  
}
```

Record

키 타입에 유연성을 제공하는 제너릭 타입

보통 `Record`는 문자열 또는 숫자로 된 속성 이름과, 해당 속성에 대한 값의 타입을 명시적으로 지정해야 하는 경우에 사용됩니다. 예를 들어, 다음과 같은 코드에서는 `Record`를 사용하여 `colors` 객체를 생성하고 있습니다.

typescript


 Copy code

```
type Color = 'red' | 'blue' | 'green';

const colors: Record<Color, string> = {
  red: '#ff0000',
  blue: '#0000ff',
  green: '#00ff00'
};
```

위의 코드에서 `Record`는 `Color` 타입의 속성 이름과, 이에 대한 값의 타입인 `string`을 명시적으로 지정하고 있습니다. 이를 통해, `colors` 객체의 속성 이름이 `Color` 타입으로 제한되고, 각 속성의 값이 `string` 타입이 되도록 보장할 수 있습니다.

typescript

 Copy code

```
const users: Record<string, { name: string, age: number }> = {
  'user1': { name: 'Alice', age: 30 },
  'user2': { name: 'Bob', age: 25 },
  'user3': { name: 'Charlie', age: 35 },
};
```

매핑된 타입


중복을 피하기 위해 다른 타입을 바탕으로 새로운 타입을 생성

```
type OptionsFlags<Type> = {
  [Property in keyof Type]: boolean;
};

type FeatureFlags = {
  darkMode: () => void;
  newUserProfile: () => void;
};

type FeatureOptions = OptionsFlags<FeatureFlags>;
/** {
  darkMode: boolean;
  newUserProfile: boolean
} */
```

typescript

 Copy code

```
interface Person {
  name: string;
  age: number;
}

type OptionalPerson = { [K in keyof Person]?: Person[K] };

const person: Person = { name: 'Alice', age: 30 };
const optionalPerson: OptionalPerson = { name: 'Alice' };
```


동적 데이터는 타입을 어떻게 모델링하면 좋을까?

Case	Best Way
필드의 이름을 모를 경우	인덱스 시그니처
필드에 사용가능한 타입이 있는 경우	선택적 필드 또는 유니온 타입
string 타입이 너무 광범위해서 인덱스 시그니처 사용이 어려운 경우	Record 또는 매핑된 타입

가능하다면 위의 방법보다는 정확한 타입을 사용하는 것이 좋다.