Lekcja 39: Filtrowanie z użyciem never

Właściwość dystrybucyjna typów warunkowych pozwala na kilka interesujących przypadków użycia w połączeniu z never. W naszych aplikacjach można tworzyć przydatne typy filtrów jako bloki konstrukcyjne dla zaawansowanych, samoobsługujących się typów. Pamiętaj o ostatecznym celu: chcemy modelować dane i zachowanie, ale nigdy nie utrzymujemy naszych typów poza modelem.

Model

Nasza aplikacja do handlu elektronicznego ma kolejną funkcję. Chcemy tworzyć płyty CD i LP z funkcją createMedium. Tak wygląda nasz model. Typ Medium zawiera nasze podstawowe właściwości:

type Medium = {

id: number,

title: string,

artist: string,

}

TrackInfo przechowuje liczbę utworów i całkowity czas trwania.

type TrackInfo = {

duration: number,

tracks: number

}

Płyta CD to połączenie Medium i TrackInfo. Dodajemy również kind tworzenia dyskryminowanych związków.

type CD = Medium & TrackInfo & {

kind: ‘cd’

}

LP pochodzi również z podstawowej klasy Medium. Zawiera dwie strony, z których każda przechowuje TrackInfo:

type LP = Medium & {

sides: {

a: TrackInfo,

b: TrackInfo

},

kind: ‘lp’

}

Łączymy wszystkie możliwe media w typ unii AllMedia. Definiujemy również unię kluczy multimedialnych.

type AllMedia = CD | LP

type MediaKinds = AllMedia[‘kind’] // ‘lp’ | ‘cd’

To są nasze typy. Funkcja createMedium powinna działać w następujący sposób:

1. Pierwszym argumentem jest typem, który chcemy utworzyć, albo LP, albo CD.

2. Drugi argument to wszystkie brakujące informacje, których potrzebujemy, aby pomyślnie utworzyć to medium. Nie potrzebujemy typu właściwości, który zdefiniowaliśmy w naszym pierwszym argumencie, ani identyfikatora, ponieważ zostanie on wygenerowany przez funkcję.

3. Funkcja zwraca nowo utworzony nośnik.

Do realizacji.

Wybierz gałęzie związków

Absolutne minimum nagłówka funkcji definiuje dwa typy dla argumentów i AllMedia dla typu zwracanego. Alternatywnie może to być Medium, aby wskazać typ podstawowy.

declare function createMedium(kind, info): AllMedia

Pierwszy typ, jaki możemy zdefiniować, to rodzaj, który chcemy wybrać. Musi być typu MediaKinds.

declare function createMedium(

kind: MediaKinds, info

): AllMedia

Używamy generycznych do wiązania rzeczywistego typu wartości, jeśli używamy literału.

declare function createMedium<

Kin extends MediaKinds

>(

kind: Kin, info

): AllMedia

Teraz, gdy wiemy, jaki rodzaj nośnika chcemy stworzyć, możemy skupić się na oczekiwanym wyniku. AllMedia jest zdecydowanie za szeroka, ale jak wybrać konkretną gałąź w naszym związku? Pamiętaj, że typy warunkowe są podzielone na typy związków, co oznacza, że warunek związków jest jak suma warunków. Możemy użyć tego zachowania, aby utworzyć typ warunkowy, który sprawdza, czy każdy składnik unii jest podtypem tego rodzaju, dla którego filtrujemy. Jeśli tak, zwracamy składnik. Jeśli nie, zwraca never.

type SelectBranch<Brnch, Kin> =

Brnch extends { kind: Kin } ? Brnch : never

Zwróć uwagę na nagi typ Brnch! Zobaczmy, co się stanie, jeśli uruchomimy przez niego llMedia i wybierzemy gałąź dla cd.

// We create a type where we want to select the

```
// cd branch of the AllMedia Union

type SelectCD = SelectBranch<AllMedia, 'cd'>

// This equals

type SelectCD = SelectBranch<CD | LP, 'cd'>

// A conditional of unions is like a union of

// conditionals

type SelectCD =

SelectBranch<CD, 'cd'> |

SelectBranch<LP, 'cd'>

// Substitute for the implementation

type SelectCD =

(CD extends { kind: 'cd' } ? CD : never) |

(LP extends { kind: 'cd' } ? LP : never)

// Evaluate!

type SelectCD =

// This is true! Awesome! Let's return CD

(CD extends { kind: 'cd' } ? CD : never) |

// This is false. let's return never

(LP extends { kind: 'cd' } ? LP : never)

// Equal to

type SelectCD = CD | never
```

Kończymy połączeniem CD | never. Ponownie, dla każdego składnika związku otrzymujemy odpowiedni typ. Jednak never nie jest to niemożliwe. Jak mówi nazwa, to nigdy się nie zdarzy! Dlatego TypeScript usuwa ze unii wszystko, co stanowi, że never nie ma, jeśli są dostępne inne składniki. Tak więc SelectBranch <AllMedia, 'cd'> ostatecznie przekształca się w CD. Zaktualizujmy nasz typ zwrotu.

```
declare function createMedium<

Kin extends MediaKinds

>(

kind: Kin, info

): SelectBranch<AllMedia, Kin>
```

Wiążą Kin z typem wartości, otrzymujemy poprawną gałąź naszego typu unii. Poręczny! Ponadto, jeśli użyjesz wzorca dodawania właściwości kind, aby często tworzyć związki dyskryminowane, typ SelectBranch stanie się typem pomocniczym wielokrotnego użytku w Twoim arsenale typów.

Extract

O wiele bardziej ogólnym typem jest wbudowany typ pomocnika Extract <A, B>. Extract <A, B> definiuje się jako.

type Extract<A, B> = A extends B ? A : never

Dokumentacja mówi, że wyodrębnia z A te typy, które można przypisać do B. Może to być zestaw kluczy lub (w naszym przypadku) obiekty

// Przekłada się na LP

type SelectLP = Extract <AllMedia, {kind: 'lp'}>

W momencie dodania kolejnego medium do unii typu AllMedia, wszystkie nasze typy są aktualizowane automatycznie. Mamy nowe rodzaje, które możemy przekazać do createMedium, ale wiemy też, że otrzymujemy z powrotem inny nośnik. Brak konserwacji z naszej strony. Po prostu dodajemy coś do modelu.