



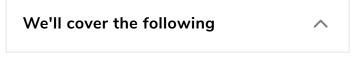






Custom Mapped Type

This lesson explains how to create your own mapped type.



- Creating a "NonNullable" type
- Adding a property conditionally

Creating a "NonNullable" type#

The first custom type shows how to use never to tell TypeScript to not compile if a custom map is not respected. The code map is a generic variable that is neither undefined nor null. In the case that the value is either one, TypeScript does not compile.

Creating a custom mapping requires using the keyword extends which acts

like an if statement. The T is checked against what is writt



extends . If true, it does after the ? . In that example, the value is never on line 2. When TypeScript returns never, it knows that it should never be in that state and stops compilation. Otherwise, it selects the value after the : which is the type \intercal itself.

Adding a property conditionally#

Imagine the scenario where an object has a dateCreated property and you want to add a modifiedDate property automatically.

```
interface Person {
   name: string;
   dateCreated: Date;
interface Animal {
   name: string;
// Create a generic Type that add modifiedDate only if dateCreated is present
type Modified<T> = T extends { dateCreated: Date } ? T & { modifiedDate: Date } : never;
const p: Person = { name: "Pat", dateCreated: new Date() };
const a: Animal = { name: "Jack" };
// ModifiedDate present because "Person" has dateCreated
const p2: Modified<Person> = { ...p, modifiedDate: new Date() };
console.log(p2.modifiedDate)
// Following line do not transpile because Animal does not have dateCreated
// const a2: Modified<Animal> = { ...p, modifiedDate: new Date() };
// console.log(a2.modifiedDate)
                                                                                        נכ
```

In the example above, the code compiles as long as the T type has dateCreated. Line 10 creates a type that accepts a generic type. The generic type is enhanced by a modifiedDate only if the generic type has dateCreated.

Otherwise, the type is never, causing TypeScript to not comp







The commented code on lines 19 and 20 tried to use the ModifiedType but do not have dateCreated, hence it does not compile.

An alternative could have been to return the generic type instead of never. In that case, TypeScript would compile if a check is done to ensure that modifiedDate is present before using the field.

