Omit

This lesson explains the omit mapped type.

The mapped type `Omit` is the inverse of `Pick`. While `Pick` is used to select a member of a type, `Omit` takes everything except the member selected. Before going any further, here is the exact same example from `Pick`, except using the `Omit` mapping instead.

In the following code, **lines 1** to **10** define a type with an interface called `Animal`. However, in **lines 17** to **21**, we have a function that requires a subset of all the fields of an animal. We could `Pick` the field needed, however in the scenario where only a few fields need to be removed, `Omit` is preferred. The reason is that fewer fields need to be specified: only the one not desired. **Line 13** has a function that removes three fields from `Animal`.

```
1  interface Animal {
2      age: number;
3      name: string;
4
5      maximumDeepness: number;
6
7      numberOfLegs: number;
8      canSwim: boolean;
9      runningSpeed: number;
10 }
11
12 // Parameter using Omit to remove three fields
13 function buyAFish(fishEntity: Omit<Animal, "numberOfLegs" | "canSwim" | "runningSpee
14     console.log(fishEntity);
15 }
16
17 buyAFish({
18     age: 1,
19     name: "Clown Fish",
20     maximumDeepness: 10,
21 });
```

`Omit` is actually a combination of two mapped types: `Pick` and `Exclude`.

```
type Omit<T, K> = Pick<T, Exclude<keyof T, K>>
```

How to build your own omit with pick and exclude

Back

Pick

Next

Record

Mark as Completed

Report an Issue