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, line 1 to 10 defines a type with an interface called `Animal`. However, we have at line 17 to 21 a function that requires a subset of all the field 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 less field needs to be specified: only the one not desired. Line 13 has a function that removes from `Animal` three fields.

```
interface Animal {
    age: number;
    name: string;

    maximumDeepness: number;

    numberOfLegs: number;
    canSwim: boolean;
    runningSpeed: number;
}

// Parameter using Omit to remove three fields
function buyAFish(fishEntity: Omit<Animal, "numberOfLegs" | "canSwim" | "runningSpeed" >) {
    console.log(fishEntity);
}

buyAFish({
    age: 1,
    name: "Clown Fish",
    maximumDeepness: 10,
});
```

`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