6 TypeScript Utilities

you should know about !

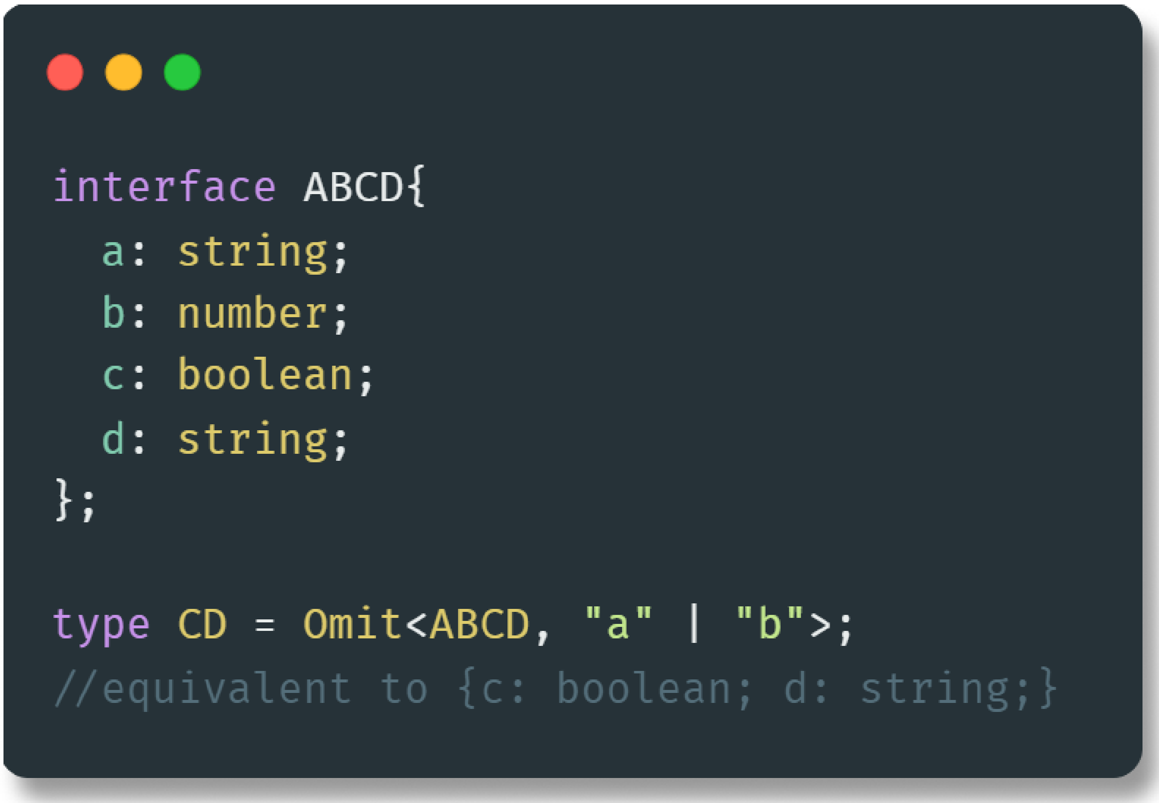
If you are working on a large project, you will find yourself with **a lot of interfaces** for data structures with **duplication of properties.**

TypeScript provides us a variety of **utility functions** to write more **readable**, **non-repititive** and **maintainable code**.

Let’s look at six helpful utility functions in this post...

Omit

This utility function helps you ***create a sub type from an existing type by excluding some fields*** *from the latter one.*



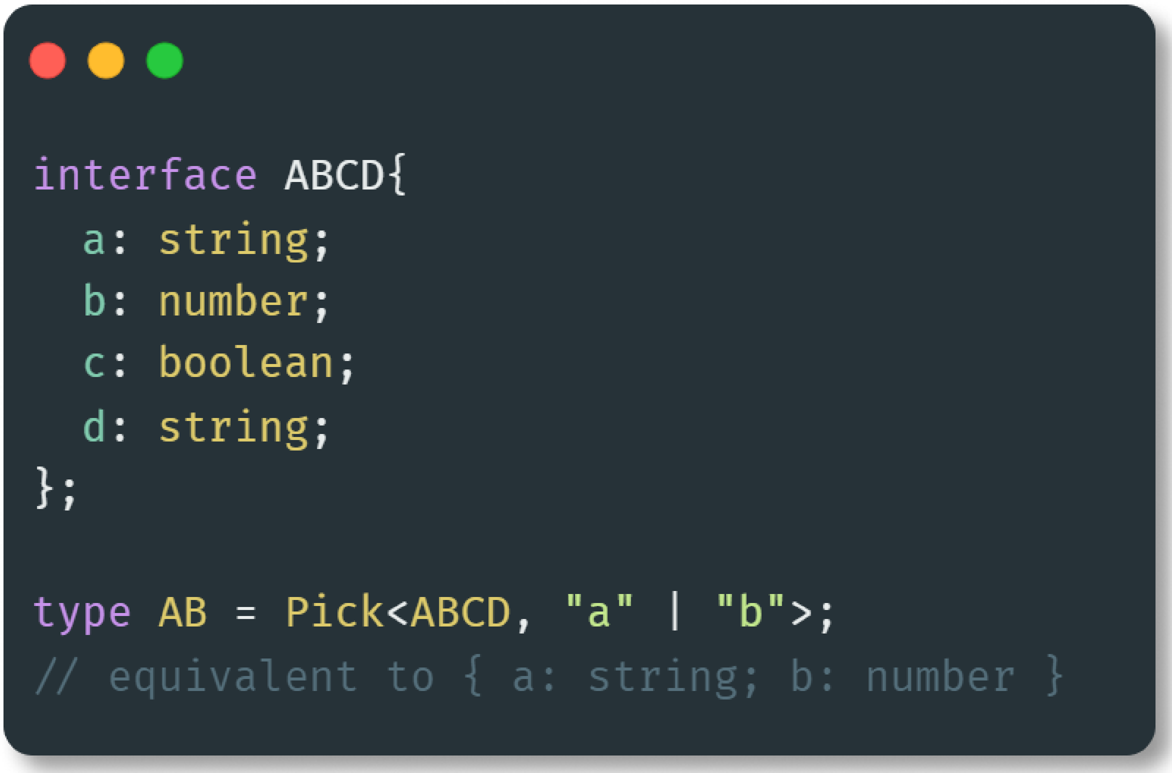
```
interface ABCD{
  a: string;
  b: number;
  c: boolean;
  d: string;
};

type CD = Omit<ABCD, "a" | "b">;
//equivalent to {c: boolean; d: string;}
```

antra verma

2 Pick

Pick is useful for maintaining the type checking ***when we only want a select number of properties*** *from an existing interface*



```
interface ABCD{
  a: string;
  b: number;
  c: boolean;
  d: string;
};

type AB = Pick<ABCD, "a" | "b">;
// equivalent to { a: string; b: number }
```

3

Partial

*Partial creates a new type with **all properties** of the specified Type **set to optional.***

```
interface ABCD{
  a: string;
  b: number;
  c: boolean;
  d: string;
}

type PartialABCD = Partial<ABCD>;
// equivalent to:
// { a?: string; b?: number; c?: boolean; d?: string; }
```

4

NonNullable

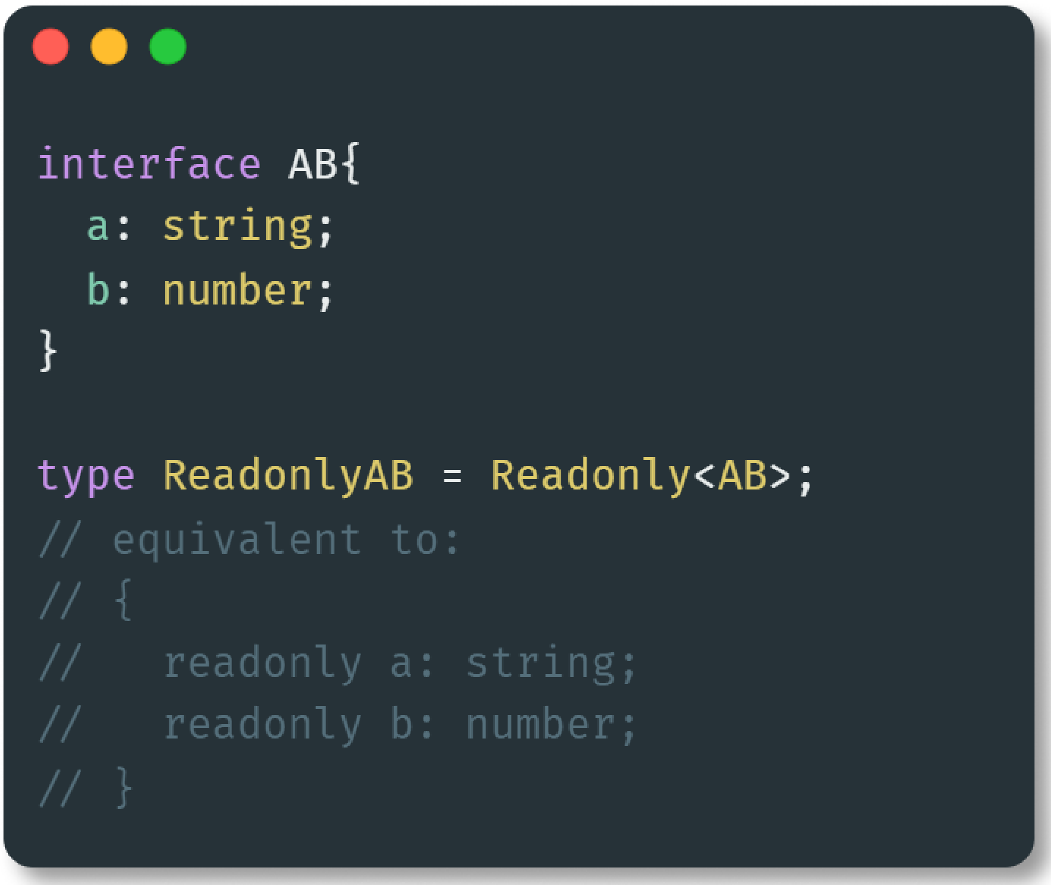
*NonNullable creates a new type by **excluding null and undefined** from Type.*

```
type Type = string | null | undefined;

type NonNullableType = NonNullable<Type>;
```

5 Readonly

*Readonly creates a new type with **all properties of Type set to readonly,** which means that they cannot be reassigned after initialization:*



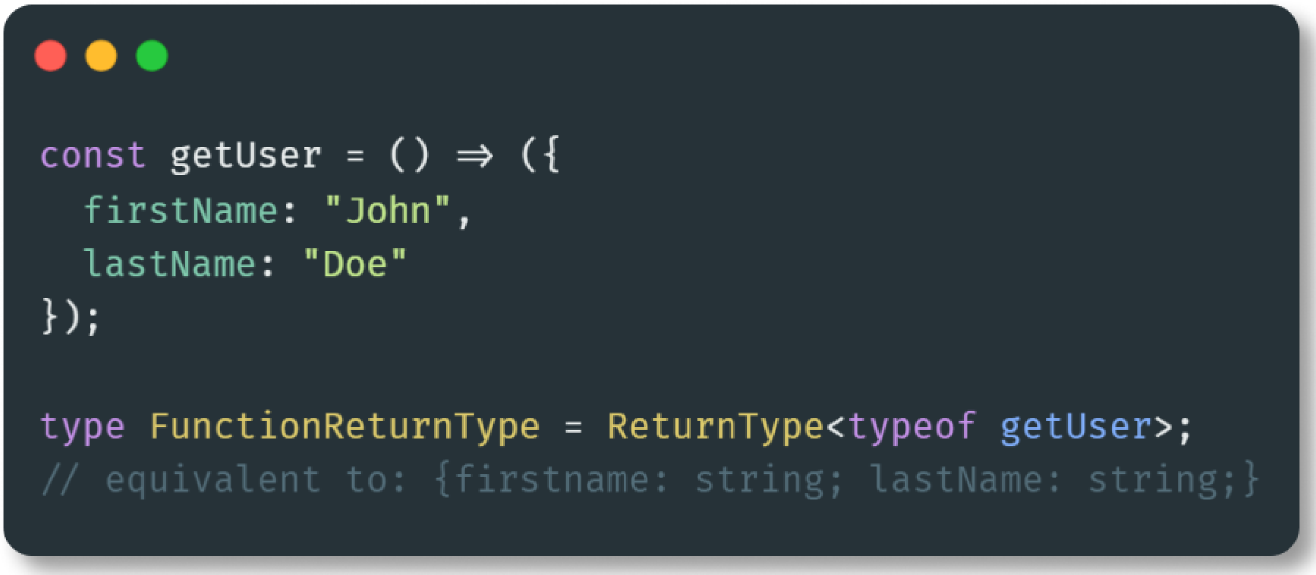
```
interface AB{
  a: string;
  b: number;
}

type ReadonlyAB = Readonly<AB>;
// equivalent to:
// {
//    readonly a: string;
//    readonly b: number;
// }
```

6

ReturnType

ReturnType constructs a ***type of the return type of a function*** *Type*



```
const getUser = () => ({
  firstName: "John",
  lastName: "Doe"
});

type FunctionReturnType = ReturnType<typeof getUser>;
// equivalent to: {firstname: string; lastName: string;}
```

antra verma