Creating intersection types

In this lesson, we'll learn what an intersection type is and how to create one.

WE'LL COVER THE FOLLOWING

- Understanding an intersection type
- Intersection of common members
- Wrap up

Understanding an intersection type

Intersection types, like union types, combine existing types to form a new type. An intersection type will have **all** the members from the types it is based on. Note that intersection types don't contain only common members from the types it is based on, as you may first expect.

An intersection type is constructed from existing types using the ampersand (`&`) character:

```
type A_and_B_and_C = A & B & C;
```

The example below combines `Name` and `PhoneNumber` types to create a `Contact` type containing `firstName`, `lastName`, `landline`, and `mobile` properties:

</> TypeScript

```
type Name = {
    firstName: string;
    lastName: string;
}
type PhoneNumber = {
    landline: string;
    mobile: string;
}

type Contact = Name & PhoneNumber;
```

```
const fred: Contact =  {
    firstName: "Fred",
    lastName: "Smith",
    landline: "0116 4238978",
    mobile: "079543 4355435"
}

console.log(fred);
```

Let's say we have an `Email` type as follows:

```
type Email = {
  emailAddress: string;
}
```

How could we add this to the `Contact` type so that it contains the `emailAddress` property? Implement this in the code widget above.

Show Answer

Intersection of common members

The code below creates an `age` variable that is based on the intersection of the `BaseElement` and `TextInput` types.

</> TypeScript

```
type BaseElement = {
    name: string
    kind: "text" | "number" | "email"
}
type TextInput = {
    kind: "text";
}
type Field = BaseElement & TextInput;

const age: Field = {
    name: "Age",
    kind: "number"
}
```

Can you spot the problem with this code?

Show Answer

So, the type of a common member of an intersection type is mathematically intersected.

Now consider the code below where the common member of an intersection type is a function:

</> TypeScript

```
type A = {
    doIt: (a: string) => void;
}
type B = {
    doIt: (a: string, b: string) => void;
}
type A_and_B = B & A;

// does this raise a type error?
const ab_v1: A_and_B = {
    doIt: (a: string) => { }
}

// does this raise a type error?
const ab_v2: A_and_B = {
    doIt: (a: string, b: string) => { }
}
```

Which variable will raise a type error? `ab_v1` or `ab_v2`?

Show Answer

So, the parameters of a common function member of an intersection type are mathematically intersected.

Wrap up

The intersection type is a way of creating useful types from existing types. It is a common misconception that interfaces are more powerful than type aliases

because they can be extended. However, type aliases can extend existing types using the intersection operator.

More information on intersection types can be found in the TypeScript handbook.

In the next lesson, we will see how different types can be compatible.