# Creating union types

In this lesson, we'll learn what a union type is, how to create one and some cases where they are useful.

WE'LL COVER THE FOLLOWING

- Understanding a union type
- String literal union types
- Object union types
- Wrap up

## Understanding a union type

As the name suggests, union types are types that we can combine together to form a new type. A union type is constructed from existing types using the pipe (`|`) character:

```
type A_or_B_or_C = A | B | C;
```

Let's go through an example to make this clear. Let's say we have an `age` variable that can be `null` or numeric. How could we create a type for this? Well we can combine the `number` and `null` types in a union type as shown below:

</> TypeScript

```
let age: number | null;
age = null;      // okay
age = 30;        // okay
age = "30";      // error
```

Unioning a type with `null` is a common use case because data can often be `null` or a specific type.

How could we change the type of `age` so that it can be `undefined` as well as a `number` or `null`? Make the change in the code widget above.

Show Answer

## String literal union types

Specific string values can be unioned together to form what is called **string literal union types**.

What would the string literal union type be to represent the name of a fruit where it can only be *Banana*, *Apple*, or *Pear*?

Show Answer

These types are really useful when you want a more specific and narrower type than `string`.

## Object union types

Objects can be unioned together as well. For example:

```
type Actions = { type: "loading" } | { type: "loaded", data: {name: string } }
```

Can you think of an area in React code where this might be useful?

Show Answer

## Wrap up

The union type is a way to create useful types from existing types. If a type can be something *or* something else, then a union type can be used to represent that type.

*String literal unions* are a way of creating narrower typed strings than a plain string. Unioning with `null` or `undefined` is a neat way to handle data that has a specific type and is empty.

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

Excellent; we are getting to grips with creating more advanced types now!

In the next lesson, we will learn about a slightly different way of combining existing types to construct a new type.