



Understanding and using the void type

In this lesson, we'll learn about the 'void' type.

We'll cover the following



- An example
- When to use the void type
- Wrap up

An example

The code below outputs a string to the console.

TS TypeScript

```
1 function logMessage(message: string) {  
2   console.log(message);  
3 }
```



The function doesn't return anything, so what is the return type of the function?





The `void` type can't hold any data; it can only be `undefined` or `null` if the `strictNullChecks` compiler option is off.

The code below shows the TypeScript compiler not happy with a variable of type `void` set to a `string` value:

TS TypeScript

```
let whatCanIHold: void;  
whatCanIHold = undefined;  
whatCanIHold = "something";
```



When to use the `void` type

`void` is only really useful for function return types and it can be explicitly defined on functions after the parameter parentheses like in the example below:

```
function logMessage(message: string): void {  
  console.log(message);  
}
```

Wrap up

So, the `void` type is to define that a function doesn't return anything. TypeScript will correctly infer this type if a function doesn't return anything, so we don't need to define it explicitly.



More information can be found about the `void` type in the [TypeScript handbook](#).



In the next lesson, we will learn about the `never` type, which is often confused with the `void` type.

[← Back](#)[Next →](#)

Avoiding `any` at Any Time Possible

Mutable and Immutable Arrays

 Completed



Report an Issue

