TypeScript union Type

Summary: in this tutorial, you will learn about the TypeScript union type that allows you to store a value of one or several types in a variable.

Introduction to TypeScript union type

Sometimes, you will run into a function that expects a parameter that is either a number or a string. For example:

```
function add(a: any, b: any) {
    if (typeof a === 'number' && typeof b === 'number') {
        return a + b;
    }
    if (typeof a === 'string' && typeof b === 'string') {
        return a.concat(b);
    }
    throw new Error('Parameters must be numbers or strings');
}
```

In this example, the `add()` function will calculate the sum of its parameters if they are numbers.

If the parameters are strings, the `add()` function will concatenate them into a single string.

If the parameters are neither numbers nor strings, the `add()` function throws an error.

The problem with the parameters of the `add()` function is that its parameters have the `any` type. It means that you can call the function with arguments that are neither numbers nor strings, the TypeScript will be fine with it.

This code will be compiled successfully but cause an error at runtime:

```
add(true, false);
```

To resolve this, you can use the TypeScript union type. The union type allows you to combine multiple types into one type.

For example, the following variable is of type `number` or `string`:

```
let result: number | string;
result = 10; // OK
result = 'Hi'; // also OK
result = false; // a boolean value, not OK
```

A union type describes a value that can be one of several types, not just two. For example `number | string | boolean` is the type of a value that can be a number, a string, or a boolean.

Back to the `add()` function example, you can change the types of parameters from the `any` to a `union` like this:

```
function add(a: number | string, b: number | string) {
    if (typeof a === 'number' && typeof b === 'number') {
        return a + b;
    }
    if (typeof a === 'string' && typeof b === 'string') {
        return a.concat(b);
    }
    throw new Error('Parameters must be numbers or strings');
}
```

We can specify the union type for the add function:

```
function add(a: number | string, b: number | string) :  number | string {
    if (typeof a === 'number' && typeof b === 'number') {
        return a + b;
    }
    if (typeof a === 'string' && typeof b === 'string') {
        return a.concat(b);
    }
```

```
    throw new Error('Parameters must be numbers or strings');
}
```

Later, you will learn about the generic type to handle this more elegantly.

Summary

- A TypeScript union type allows you to store a value of one or several types in a variable.