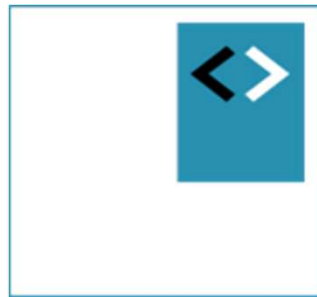


# Training TypeScript

## Module: Generic Function Overloads



Peter Kassenaar

[info@kassenaar.com](mailto:info@kassenaar.com)

# What is an overloaded function?

*Overloaded functions are functions with the same name, but different implementations. The compiler decides which overload is called.*  
*(in C# or Java)*

But: JavaScript **DOESN'T KNOW** overloaded functions

## However, you might want to use overloads

Because TypeScript compiles to plain JavaScript, there are also **no real 'overloads'**.

So anything you do in this area is just an **aid for you**, as developer. And: the **IDE's or Tools** can help you out in showing the different overloads available.

# Step 1. Create the end result/function

- You ALWAYS need to create the **end result** with the **most extensive usage** / implementation / num arguments that you have in mind.
- For instance, you want to build a function that arbitrarily reverses a string, an array or a number
- The you **start with the final implementation**. For instance, like this:

```
function reverse(value: string | number | any[]): string | number | any[] {  
    // Use type guard  
    if (typeof value === 'string') {  
        // Deal with strings...  
    }  
    if (typeof value === 'number') {  
        // Deal with numbers...  
    }  
    // Deal with arrays...  
}
```

../45-overload-functions.ts

# Check the output

```
// 7. implementation
console.log(reverse('Hello World'));
console.log(reverse(-12345));
console.log(reverse(['A', 'B', 'C', 'D', 'E']));
```

```
[nodemon] starting `node .\45-overload-functions.js`
dlroW olleH
-54321
[ 'E', 'D', 'C', 'B', 'A' ]
[nodemon] clean exit - waiting for changes to be
```

It works

```
// 7. implementation
console.log(reverse(value: 'Hello World'));
console.log(reve
console.log(reve
// *****
```

```
function reverse(
  value: string | number | any[]): string | number | any[]
```

But the tooling doesn't give you a clue which 'version' of the function you are working with.

# Create your overloads

- Re-use the name of the function with only 1 argument.
  - NO function body / implementation
- Again: overloads are there for YOU as the developer, and for your IDE / tooling

```
function reverse(value: string): string; // overload 1, accepting a string
function reverse(value: number): number; // overload 2, accepting a number
function reverse(value: any[]): any[]; // overload 3, accepting an array


// Actual implementation
function reverse(value: string | number | any[]): string | number | any[] {
    // Implementation
    // ...
}
```

# Use in your editor

- Now, your editor shows you exactly what type you are dealing with and what overload you use
  - This is much clearer:
- Of course, this mostly comes in handy if you create generic functions or helper libraries that are used by others
  - Or, by yourself in the future!

```
// 7. implementation
console.log(reverse( value: 'Hello World'));
console.log(reverse( value: -12345));
console.log(reverse( value: 12345));

// ***** src/ts/45-overload-functions.ts
```



# Make use of Generics

- You can deal with **specific types** or **Generics** if you want to:

```
function reverse<T>(value: T[]): T[]; // overload 3, accepting a Typed array  
//...  
function reverse<T>(value: string | number | T[]): string | number | T[] {  
  ...  
}
```

```
const myList: number[] = [1, 2, 3, 4, 5]  
console.log(reverse(myList));
```

```
function reverse<number>(  
  value: number[]): number[]
```

```
src/ts/45-overload-functions.ts
```



# Workshop

- Create a helper function that does something for you.
- It must accept multiple arguments of different types.
  - These CAN be unions, if you want to.
- See if the function works if you call it with different arguments, as laid out in the slides
  - Look at your editor. What is the accepted type(s) and what is the return type(s)?
- Create **function overloads** – position them at the **right spot**
- See if, and how, your editor helps you this way.
- Q: can you think of situations where you have seen this, or where this might come in handy?
- Example code: `../45-overload-functions.ts`

