✓ 100 XP

Exercise - Create functions

5 minutes

In this exercise, you'll create two named functions with strongly typed parameters and return values.

1. Open the Playground and remove any existing code.

2. Copy the following JavaScript function into the Playground. Notice that the parameter `message` implicitly has an `any` type.

   JavaScript

   ```javascript
   function displayAlert(message) {
       alert('The message is ' + message);
   }
   ```

3. Now, assign the `string` type to the `message` parameter. This named function doesn't return a value so you can leave off the return type (you can also pass back `void` as the return type, but it isn't necessary to do so.)

   TypeScript

   ```typescript
   function displayAlert(message: string) {
       alert('The message is ' + message);
   }
   ```

4. Try calling the function, passing in a `string` as a parameter. Now, try passing in a `number`. TypeScript type checks the parameter and notifies you of the conflict. Depending on what you're trying to accomplish with this function, you can either put the number in quotes, expand the types of values accepted by the parameter with a union type (for example `msg: string | number`), or add some logic to your function to handle the different types of values passed into it.

5. Here's another example. The `sum` function totals the numbers in an array and returns the result. Copy the JavaScript code into the Playground.

JavaScript

```
function sum(input) {
    let total =  0;
    for(let count = 0; count < input.length; count++) {
        if(isNaN(input[count])){
            continue;
        }
        total += Number(input[count]);
    }
    return total;
}
```

6. Try calling the function with a single number as a parameter, for example, `sum(5)`. It accepts the value but doesn't return the correct result because the parameter isn't passed as an array.

7. Set the type of the `input` parameter to an array of `number` values, set the return type of the function to `number`, and set the type of the `total` variable to `number`.

TypeScript

```
function sum(input: number[]): number {
    let total: number =  0;
    for(let count = 0; count < input.length; count++) {
        if(isNaN(input[count])){
            continue;
        }
        total += Number(input[count]);
    }
    return total;
}
```

8. Now, if you call the function with `sum(5)`, TypeScript flags the type issue with the parameter.

9. Try calling the function with an array of values that have mixed types, for example, `sum([1, 'two', 3])`. The values inside the array are type checked, and TypeScript returns the error: `Type 'string' is not assignable to type 'number'.`

Next unit: Fun with parameters

Continue >

How are we doing? ☆ ☆ ☆ ☆ ☆