100 XP

Exercise - Fun with parameters

5 minutes

In this exercise, you'll create functions that have required, optional, and default parameters.

Required parameters

1. Open the Playground and remove any existing code.
2. Enter the following arrow function, which accepts three required parameters.

```
TypeScript
let addThreeNumbers = (x: number, y: number, z: number): number => x + y + z;
```

3. Try calling the function by entering `addThreeNumbers(10, 20)`. TypeScript raises the error **Expected 3 arguments but got 2. An argument for 'z' was not provided.** When it runs, the function returns `NaN` because the third argument was passed as `undefined`, making the calculation invalid.
4. What happens when you enter `addThreeNumbers(10, 20, 30, 40)`? TypeScript raises the error **Expected 3 arguments but got 4.** When it runs, the fourth argument drops off and the function returns `60`.

Optional parameters

1. In the function, try making the `y` parameter optional. What happens?

```
TypeScript
let addThreeNumbers = (x: number, y?: number, z: number): number => x + y +
z;
```

2. TypeScript **raises an error** because the position of the optional parameters matter. In the

parameter list, optional parameters must follow all required parameters. Instead of the `y` parameter, try making the `z` parameter optional. Also, for this function to return the correct value, you must also update it to address the possibility that `z` may now be passed as `undefined`. You should now be able to call the function using `addThreeNumbers(10, 20)` or `addThreeNumbers(10, 20, 30)`.

TypeScript

```
let addThreeNumbers = (x: number, y: number, z?: number): number => {
    if((z === undefined)) {
        return x + y;
    } else {
        return x + y + z;
    }
};
```

Default parameters

1. Enter the following arrow function, which accepts three required parameters.

TypeScript

```
let subtractThreeNumbers = (x: number, y: number, z: number): number => x - y
- z;
```

2. Assign a default value of `100` to the `z` parameter by replacing `z: number` with `z = 100`.

TypeScript

```
let subtractThreeNumbers = (x: number, y: number, z = 100): number => x - y -
z;
```

3. Try calling the function with two and three arguments to test the result.

TypeScript

```
subtractThreeNumbers(10, 20);       // returns -110 because 'z' has been as-
signed the value 100
subtractThreeNumbers(10, 20, 15);   // returns -25
```

Next unit: Exercise - Define function types

Continue >

How are we doing? ☆ ☆ ☆ ☆ ☆