Trainer guide
Examples and exercises
JS-5 – Callbacks

Contents

JS-5a | Exercise: Callbacks

Link to environment

This exercise gives learners practice on passing a function as an argument to another function.

index.js

- *Explain the existing code in the file:*
    - First, we have a function `greet`, which takes in one parameter, `name`, and writes a greeting message in the console.
    - We have another function, `forEach`, which takes in two parameters: the first parameter is an array, and the second parameter is a function. Remember, functions that are passed as arguments to other functions are called "callback functions".
        - `forEach` will loop through each element of the array and execute the given callback function on each element.
    - We also have an array of names.
    - Finally, we execute the `forEach` function.
        - We pass in the array of names as the first argument.
        - For the second argument, we pass in the `greet` function.
            - Notice that we don't put brackets after `greet`. If we did this, the function would execute immediately on this line – but we don't want that. We just want to pass the function itself so that it can be called at a later time.
- Now look at the console.
- You can see that the `greet` function has been called for every element of the `names` array.

Exercise

- Your task is to write some very similar code.
    - Using the `forEach` function, instead of saying hello to each person in the array, write some code to say **goodbye** to each person in the array.

Solution

```
function sayGoodbye(name) {
  console.log(`Goodbye, ${name}!`)
}

forEach(names, sayGoodbye)
```

Trainer demo | Arrow functions as callbacks

[Link to environment]

In this demo, you will show that arrow functions can be passed as callbacks.

index.js

- We've looked at arrow functions, and we've looked at callbacks, so let's now combine these two concepts.
- We have the same example as before, with the `greet` and `forEach` functions.
- Let's turn `greet` into an arrow function.
- *Rewrite `greet`:*

```
const greet = (name) => console.log(`Hello, ${name}!`)
```

- Now, when we call `forEach`, instead of passing `greet` as a named function, we could just grab the arrow function itself and pass that.
- *Cut the arrow function expression:*

```
(name) => console.log(`Hello, ${name}!`)
```

- *Paste this as the second argument to `forEach`:*

```
forEach(names, (name) => console.log(`Hello, ${name}!`))
```

- We've now achieved the same thing as before, but this time we've done it by passing an unnamed (anonymous) arrow function as an argument. Our code is very concise, and very flexible, as we can easily perform whatever behaviour we want on the names array.
- For example, we could change it to behave like the `sayGoodbye` function:

```
forEach(names, (name) => console.log(`Goodbye, ${name}!`))
```

- Using arrow functions as callbacks is a very common pattern in modern JavaScript. This concept will be used in the next topic.