JS Callback Functions

Learning Objectives

- Understanding the concept of callback functions
- Using an anonymous callback function
- Using a named function as a callback function
- Knowing what a higher order function is

Callback Functions

A callback function is a function that is passed **as an argument** into another function.

The outer function can execute this callback function at the correct moment or multiple times, for example:

- when an event is triggered
- when the fetched data arrived on your computer
- for each element in an array.

Callback functions are used, whenever the program itself needs to figure out **when** or **how many times** the function needs to be executed. We already used callback functions in **event listeners**:

```
button.addEventListener("click", () => {
  console.log("Inside the callback function.");
});
```

Here the structure is as follows:

- outer function: `addEventListener()`
- first argument: `'click'`
- second argument: callback function

```
() => {
  console.log("Inside the callback function.");
};
```

This type of function is called **anonymous function**, since it is declared without giving it a name.

Named Callback Functions

Any function can be used as a callback function. It just needs to be passed to another function. You can declare a normal function and then use the **name of the function** to pass it into another function:

```
function sayHello() {
  console.log("Hey Dude!");
}

button.addEventListener("click", sayHello);
```

> ❗ Note that we do not call the function here (we wrote `sayHello` instead of `sayHello()`). We only pass the function to the event listener. The function is only called when the event happens.

Higher Order Functions

A higher order function is a function that takes a **callback function as an argument** and **calls the callback function** inside their body, e.g. the `addEventListener` method.

```
// this function calls its callback function 3 times!
function myHigherOrderFunction(callback) {
  callback();
  callback();
  callback();
}
```

We will encounter these higher order functions in future sessions:

- `.then`
- `.forEach`
- `.map`
- `.filter`

> 💡 Don't worry, you don't have to write higher order functions yourself, you only apply them to solve certain problems.

Parameters in Callback Functions

A callback function can accept parameters. The values for the parameters are provided by the function, that calls the callback function (the "higher order function").

In this example the callback function can accept a parameter to retrieve information about the occurred event:

```
button.addEventListener("click", (event) => {
  console.log("This button was clicked:", event.target);
});
```

Resources

- MDN Callback Functions