JavaScript Basics

CallBack Functions

Mark as done

CallBack Functions

➤ A callback function is a function that is passed as an argument to another function, to be "called back" at a later time.

➤ A “higher-order function” is a function that accepts functions as parameters and/or returns a function. It contains the logic for when the callback function gets executed.

➤ It's the combination of these two that allow us to extend our functionality.

➤ Callbacks are used to execute code after another function or an asynchronous operation has finished executing.

➤ In JavaScript, a callback is easier to create. That is, we simply have to pass the callback function as a parameter to another function and call it right after the completion of the task.

➤ Callbacks are mainly used to handle the asynchronous operations such as the registering event listeners, fetching or inserting some data into/from the file, and many more.

> event listeners topic will be covered later on.

➤ The following defines a `pickNumbers()` function that accepts an array of numbers and returns a new array of odd numbers:

```
function pickNumbers(numbers) {
  let results = [];
  for (const number of numbers) {
    if (number % 2 != 0) {
      results.push(number);
    }
  }
  return results;
}
let numbers = [1, 2, 4, 7, 3, 5, 6];
console.log(pickNumbers(numbers)); // [1, 7, 3, 5]
```

How it works.

- First, define the `pickNumbers()` function that accepts an array of numbers and returns a new array of the odd numbers.
- Second, define the `numbers` array that has both odd and even numbers.
- Third, call the `pickNumbers()` function to get the odd numbers out of the numbers array and output the result.

➤ If you want to return an array that contains even numbers, you need to modify the `pickNumbers()` function. To make the `pickNumbers()` function more generic and reusable, you can:

- First, extract the logic in the `if` block and wrap it in a separate function.
- Second, pass the function to the `pickNumbers()` function as an argument.

Here’s the updated code:

```
function isOdd(number) { // this function is invoked as callback
  return number % 2 != 0;
}

// for even values
function isEven(number) { // this function is invoked as callback
  return number % 2 == 0;
}

function pickNumbers(numbers, fn) {
  let results = [];
  for (const number of numbers) {
    if (fn(number)) {
      results.push(number);
    }
  }
  return results;
}
let numbers = [1, 2, 4, 7, 3, 5, 6];
console.log(pickNumbers(numbers, isOdd)); // [1, 7, 3, 5]
console.log(pickNumbers(numbers, isEven)); // [2, 4, 6]
```

The result is the same. However, you can pass any function that accepts an argument and returns a boolean value to the second argument of the `pickNumbers()` function.

When you pass a function as an argument, remember not to use parenthesis.

- A callback is a function passed into another function as an argument to be executed later.
- A high-order function is a function that accepts another function as an argument.
- Callback functions can be synchronous or asynchronous.

Why do we need Callback Functions?

➤ JavaScript runs code sequentially in top-down order. However, there are some cases that code runs (or must run) after something else happens and also not sequentially. This is called asynchronous programming.

➤ Callbacks make sure that a function is not going to run before a task is completed but will run right after the task has completed. It helps us develop asynchronous JavaScript code and keeps us safe from problems and errors.

➤ In JavaScript, the way to create a callback function is to pass it as a parameter to another function, and then to call it back right after something has happened or some task is completed.

Next

You have completed 0% of the lesson

Jump to...

Reset user tour on this page