JavaScript Callbacks

A **callback** is simply a **function passed as an argument to another function**, to be **called later**.

This might sound confusing at first, but once you see it in action, it becomes very easy to understand.

Why Do We Need Callbacks?

Sometimes, you don't want a function to run immediately.
Instead, you want it to **run later**, **when something else happens** — for example:

- When a timer finishes
- When a user clicks a button
- When some data is ready

Callbacks help us do that.

Basic Example of a Callback

```
function greet(name) {
  console.log("Hello, " + name);
}

function processUser(callback) {
  const userName = "Harry";
  callback(userName);
}
```

```
processUser(greet);
```

What’s Happening Here?

- `greet` is a function that prints a greeting.
- `processUser` is another function that receives a function (callback) and **calls it later**.
- We pass `greet` as an argument to `processUser`.

So `processUser(greet)` ends up calling: `greet("Harry")`

Callbacks in Asynchronous Code

Callbacks are often used with async functions like `setTimeout`.

```
function showMessage() {
  console.log("This runs after 2 seconds");
}

setTimeout(showMessage, 2000);

console.log("This runs first");
```

Output:

```
This runs first
This runs after 2 seconds
```

Even though `showMessage` is written before the timer, it **runs later** — after 2 seconds. That’s because we passed it as a callback to `setTimeout`.

Writing Inline Callback Functions

Instead of defining the function first, we can also write it directly:

```
setTimeout(function () {
  console.log("Hello after 1 second");
}, 1000);
```

This is still a callback — just written directly where it's used.

Another Example with User Input (Browser Only)

If you're in a browser and use something like this:

```
document.getElementById("btn").addEventListener("click", function () {
  console.log("Button clicked");
});
```

The second argument to `addEventListener` is a callback. It runs **only when the user clicks** the button.

Summary

- A **callback** is just a function passed to another function.
- It can be used to run code **later**.
- Callbacks are commonly used in:
  - `setTimeout`
  - Event listeners
  - Many asynchronous operations

Callbacks are the foundation for working with asynchronous JavaScript. Once you’re comfortable with them, you’re ready to learn more advanced things like **Promises** and `async/await`.