Callback Function

A **callback function** is a function passed into another function as an argument, which is then invoked inside the outer function to complete some kind of routine or action.

The consumer of a callback-based API writes a function that is passed into the API. The provider of the API (called the *caller*) takes the function and calls back (or executes) the function at some point inside the caller's body. The caller is responsible for passing the right parameters into the callback function. The caller may also expect a particular return value from the callback function, which is used to instruct further behavior of the caller.

There are two ways in which the callback may be called: *synchronous* and *asynchronous*. Synchronous callbacks are called immediately after the invocation of the outer function, with no intervening asynchronous tasks, while asynchronous callbacks are called at some point later, after an [asynchronous](https://developer.mozilla.org/en-US/docs/Glossary/Asynchronous) operation has completed.

Understanding whether the callback is synchronously or asynchronously called is particularly important when analyzing side effects. Consider the following example:

JS

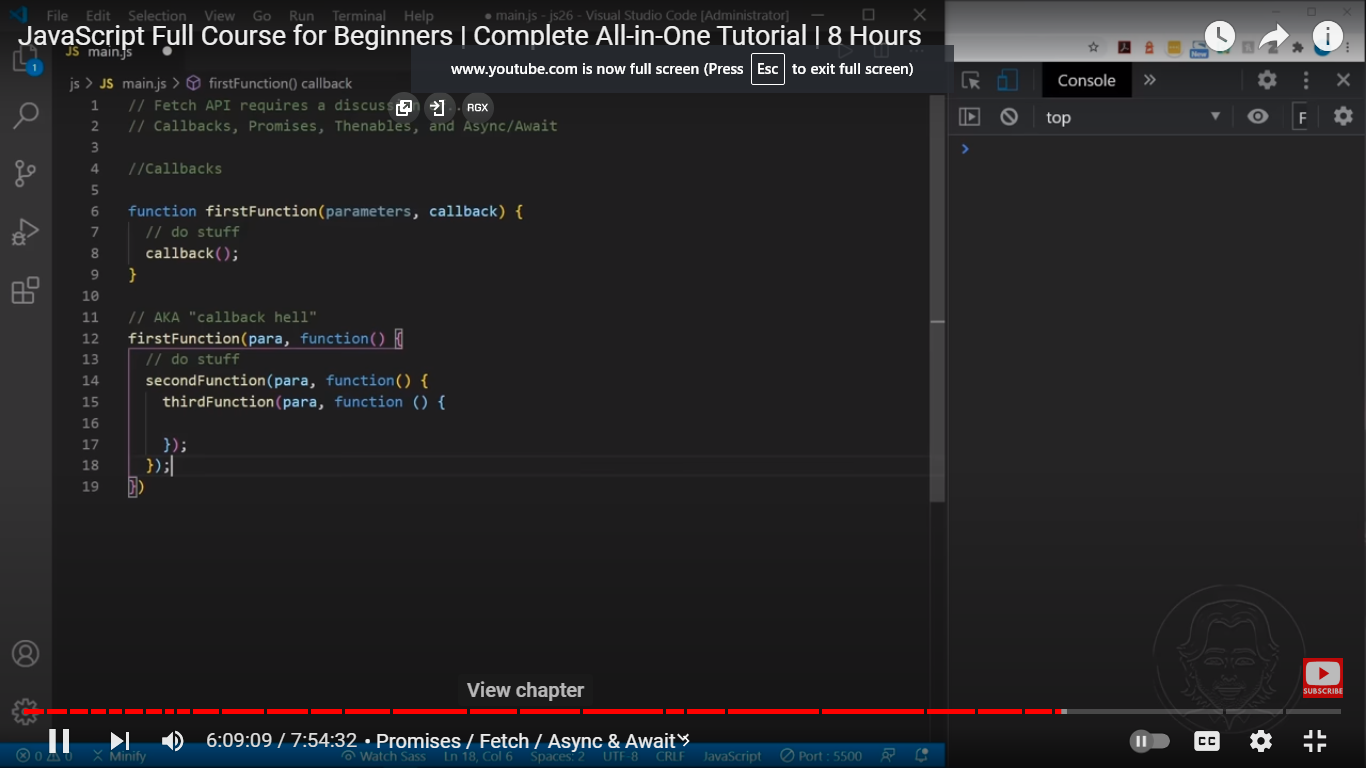
let value = 1;

doSomething(() => {

value = 2;

});

console.log(value);



A JavaScript callback is a function which is to be executed after another function has finished execution. A more formal definition would be - Any function that is passed as an argument to another function so that it can be executed in that other function is called as a callback function.