CS4220 Node.js & Vue.js

Cydney Auman
Albert Cervantes
CSULA

Callbacks, Promises and Async

Synchronous vs Asynchronous

In a ***synchronous*** programming model, things happen one at a time. When you call a function that performs a long-running action, it returns only when the action has finished and it can return the result. This stops your program for the time the action takes.

An ***asynchronous*** model allows multiple things to happen at the same time. When you start an action, your program continues to run. When the action finishes, the program is informed and gets access to the result.

Timing Methods

`setTimeout(`*`fn`*`, `*`number`*`)`

Calls a function or executes a code after specified delay.

`setInterval(`*`fn`*`, `*`number`*`)`

Calls a function or executes a code repeatedly, with a fixed time delay between each call to that function.

`clearInterval(`*`fn`*`)`

Cancels repeated action which was set up using setInterval().

Callbacks

One approach to asynchronous programming is to make functions that perform a slow action take an extra argument, a ***callback function***. The action is started, and when it finishes, the callback function is called with the result.

Callbacks are an important feature of asynchronous programming. It enables the function that receives the callback to call our code when it finishes a long running task, while allowing us to continue the execution of other code.

Simply put a callback is a function to be executed after another function is done executing.

Promises

A Promise is an object which takes a callback. A promise specifies some code to be executed later (as with callbacks) and also explicitly indicates whether the code succeeded or failed at its job. You can chain promises together based on success or failure.

A **Promise** is in one of these states:

- *pending*: initial state, not fulfilled or rejected.
- *fulfilled*: meaning that the operation completed successfully.
- *rejected*: meaning that the operation failed.

- The **`Promise.all(iterable)`** method returns a promise that resolves when all of the promises in the iterable argument have resolved, or rejects with the reason of the first passed promise that rejects.

Async Functions

An async function is marked by the word ***async*** before the **function** keyword. When such a function or method is called, it returns a ***Promise***. As soon as the body returns something, then the promise is resolved. If it throws an error, then the promise is rejected.

The word ***await*** can be put in front of a function in order to wait for a ***Promise*** to resolve and only then continue the execution of the function.

References & Readings

Eloquent Javascript

Chapter 11 – https://eloquentjavascript.net/11_async.html

Timers

https://developer.mozilla.org/en-US/Add-ons/Code_snippets/Timers

Promises, Callbacks, Async

https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Statements/async_function

https://developer.mozilla.org/en-US/docs/Glossary/Callback_function

https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Promise