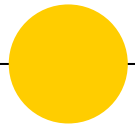
CS4220 Node.js & Angular.js

Cydney Auman
Albert Cervantes
CSULA



Callbacks, Promises and Intro to Node.js

Timing Methods

`setTimeout()`

Calls a function or executes a code after specified delay.

`setInterval()`

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

`clearInterval()`

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

Callbacks

Callbacks are functions that are passed as arguments to other functions. This is 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.

What is Node.js ?

Node.js is a platform. The creators of Node.js took JavaScript and allowed it to run on your machine. It’s built on Chrome's V8 JavaScript runtime.

The main idea is that Node.js uses an event-driven, non-blocking I/O model to remain lightweight and efficient

What Node Achieves

One of the more difficult problems in writing systems that communicate over a network is managing input and output — that is, the reading and writing of data to and from the network, the hard drive, DBs and other such devices.

Node was initially conceived for the purpose of making ***asynchronous*** I/O easy and convenient.

Where Node Excels

Any place where we can perform asynchronous I/O - reading and writing to network connections, reading/writing to the filesystem, and reading/writing to the database.

All of these are common tasks in web apps and execute very fast in Node.

- JSON APIs
- Web Apps
- Command Line Utilities/Apps
- Chat Apps

Node Modules & require

Node.js puts little functionality in the global scope.

If you want to access other built-in functionality, you have to ask the module system for it.

Node.js follows the CommonJS module system, and the built in require function is the easiest way to include modules that exist in separate files.

The basic functionality of require is that it reads a javascript file, executes the file, and then proceeds to return the exports object.

Node Core Modules

File System

- Handles File I/O

```
const fs = require('fs')
```

HTTP/HTTPS

- Interfaces designed to support features of the http or https protocol.

```
const http = require('http')
```

```
const https = require('https')
```

Event Loop

JavaScript engines are built on the concept of a single-threaded event loop (one piece of code is ever executed at a time). Java or C++ can allow multiple different pieces of code to execute at the same time.

The event loop is a process inside the JavaScript engine that monitors code execution and manages the jobs.

https://www.youtube.com/watch?v=8aGhZQkoFbQ

References & Readings

Asynchronous Programming, Promises, Callback, & Event Loop

https://leanpub.com/understandinges6/read#leanpub-auto-promises-and-asynchronous-programming

Promises

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

Event Loop

https://developer.mozilla.org/en-US/docs/Web/JavaScript/EventLoop

Timers

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