Introduction to NodeJS

Week 1 - Day 3

Topics Covered

1. Installing node.js
2. Starting a node application
3. A simple node module
4. Callbacks and error handling

Installing node.js

➔ Linux Ubuntu
➔ macOS
➔ Windows

Starting a node application

1. Create a folder named `*node-examples*` and then move into this folder.
2. At the prompt, type the following to initialize a `*package.json*` file in the node-examples folder:
   $ npm init
3. Accept the standard defaults suggested.
4. You will observe that a file `*package.json*` has been initialized. A package.json file:
   a. lists the packages your project depends on
   b. specifies versions of a package that your project can use using semantic versioning rules
   c. makes your build reproducible, and therefore easier to share with other developers

More about package.json

What is a node module?

In Node.js, a module is a collection of JavaScript functions and objects that can be used by external applications. Describing a piece of code as a module refers less to what the code is and more to what it does—any Node.js file or collection of files can be considered a module if its functions and data are made usable to external programs.

Because modules provide units of functionality that can be reused in many larger programs, they enable you to create loosely coupled applications that scale with complexity, and open the door for you to share your code with other developers.

Creating a simple node module

Resources

- Video mode
- Reading mode
  - Resource 1: Part 1, Part 2, Part 3
  - Resource 2

Callbacks and Error Handling

What is a callback?

A callback is an asynchronous equivalent for a function. A callback function is called at the completion of a given task. Node makes heavy use of callbacks. All the APIs of Node are written in such a way that they support callbacks.

For example, a function to read a file may start reading the file and return the control to the execution environment immediately so that the next instruction can be executed. Once file I/O is complete, it will call the callback function while passing the callback function, the content of the file as a parameter. So there is no blocking or wait for File I/O. This makes Node.js highly scalable, as it can process a high number of requests without waiting for any function to return results.

Resources

- Video mode
- Reading mode
    - Resource 1
    - Resource 2