Udemy: Learn and Understand NodeJS

1. Introduction and the Goal of this Course
2. Asdf
3. Asdf
4. Conceptual Aside – The Command Line Interface
5. Command Line References
6. Conceptual Aside – Processors, Machine Language, and C++
   1. When we give the microprocessor on your computer instructions, we give them to it in the specific language it understands. Otherwise it wouldn’t be able to run them. There’s a variety of instruction sets/languages that microprocessors might be designed to speak. The processor’s a physical machine that accepts instructions and carries them out.
   2. Machine code: programming languages spoken by computer processors. Every program you run on your computer has been converted (compiled) into machine code.
   3. So ultimately whenever we write a programming language, there’s another process running that converts the code you write into machine code
   4. The first thing to understand about node: Node is written in C++. The reason why is that V8 (the JavaScript engine) is C++. V8 is the thing that converts JavaScript into machine code.
7. JavaScript Aside – JavaScript Engines and the ECMAScript specification
   1. ECMAScript = the standard that JavaScript is based on. Need a standard since there are many engines. ECMAScript is the name for the core standard that JavaScript is based on. Note: V8 is actually just one of many JavaScript engine
   2. JavaScript Engine: A program that converts JavaScript code into something the computer processor can understand. And it should follow the ECMAScript standard on how the language should work and what features it should have.
8. Adding features to JavaScript
   1. V8 is able to be embedded in other C++ programs, so if you wrote a program you could put V8 inside it and use it. You could write a C++ program where other people can write JavaScript and your C++ program can take it and run it through V8.
   2. Since V8 is just C++ that just takes JavaScript and doing things with it, V8 allows us to write our own C++ code that’s available to JavaScript. Ie. We can make it so that if someone writes something in particular in their JavaScript code, that will cause my C++ code to be run. This allows us to essentially add features to JavaScript by embedding V8 into our C++ program, so our C++ program will read and understand more than what the ECMAScript standard specifies. This is very powerful because C++ has far more features than JavaScript. So, we can write things in C++ that JavaScript can’t do, and then make anything that we write available to JavaScript
   3. All this allows us to attach C++ code to our own custom JavaScript keywords. Essentially we can add features to JavaScript. This is what node.js is – a C++ program with V8 embedded that has added a wealth of features that make it suitable to be a server technology
9. Conceptual Aside: Servers and Clients
   1. Nodejs is a server technology. It’s designed to be able to use JavaScript to write server code.
   2. A **server** is a computer that’s performing services. It’s just a computer performing jobs requested of it. The **client** asks for those services (and may also do some work).
   3. The client sends a request for a service (for some work to be performed) and the server responds, with perhaps verification that the work’s been performed, or error messages, or data. And that request (and the response also) is in a standard format that both the client and server and programmed to understand.
   4. In the case of the internet, we have web servers, and the usual client is a web browser.
10. What does JavaScript need to manage a server?
    1. What are the features we need to have accessible in JavaScript in order to do the kind of things that a web server normally needs to do?
       1. Better ways to organize our code into reusable pieces
       2. Ways to deal with files
       3. Ways to deal with databases
       4. The ability to communicate over the internet
       5. The ability to accept requests and send responses (in the standard format)
       6. A way to deal with work that takes a long time (we want people to continue to be able to access the web server while it deals with a big job)
11. The C++ Core

**Lynda: Up and Running with Node.js**

Module.exports is the object that’s actually returned as the result of a require call.

The exports variable is initially set to that same object, so in the module code you would usually write something like this:

var myFunc1 = function() {…};

exports.myFunc1 = myFunc1;

to export (or “expose”) the internally scoped functions myFunc1 and myFunc2.

And in the calling code you would use:

Var m = require(‘mymodule’);

My.myFunc1();

**Lynda: Node.js Essential Training**

* Every node.js file we create is its own module, and every variable we create is scoped to that module. They aren’t available in the global namespace
* The process object is available to use globally is the process object. It contains functionality that allows us to interact with information about the current process instance.

**Tutorialspoint**

A Node.js application consists of the following three important parts:

* Import required modules – we use require directive to load a Node.js module
* Create server – a server which will listen to a client’s request similar to an Apache HTTP Server
* Read request and return response – the server created in the earlier step will read HTTP requests made by a client (can be a browser or console) and return the response.