**CS 628 Modern Full-Stack Development**

**HOS02: Server-Side Action: Node and NPM**

January 10, 2023 - Developed by Shingo Kise

School of Technology & Computing (STC)

City University of Seattle (CityU)

Icon

Description automatically generatedA blue and white flag

Description automatically generated with medium confidence

**Before You Start**

* This tutorial targets Windows users and MacOS users.
* There might be subtle discrepancies along the steps. Please use your best judgment while going through this cookbook-style tutorial to complete each step.
* For your working directory, use your course number. This tutorial may use a different course number as an example.
* The directory path shown in the screenshots may be different from yours.
* If you are not sure what to do or confused with any steps:
  + Consult the resources listed below.
  + If you cannot solve the problem after a few tries, ask a TA for help.

**Learning Outcomes**

Students will be able to:

* Understand the basics usage of Node and NPM
* Know how to write a basic web server
* Know how to execute JavaScript code, how to create an NPM project, and how to add dependencies.

**Resources**

* Zammetti, F. (2022). *Modern Full-Stack Development: Using TypeScript, React, Node.js, Webpack, Python, Django, and Docker* (2nd ed.). Apress.
  + Ch. 1: Server-Side Action: Node and NPM
  + Ch. 2: A Few More Words: Advanced Node and NPM
* Visual Studio Code - <https://code.visualstudio.com/>
* Node.js - <https://nodejs.org/en/>
* npm - <https://www.npmjs.com/>

**Introduction**

In this course, as we learn “full-stack” development, we’ll be learning about coding clients as well as the server code they make use of to form a cohesive, whole application. In the next few chapters, we will focus on the development of the backend side, more specifically the server. In this chapter, we will learn about two extremely popular tools for developing servers; Node.js and NPM (Node Package Manager).

**Of JavaScript Runtimes and Building (Mostly) Servers**

Node is a platform for running primarily, though not exclusively, server-side code that has high performance and is capable of handling large request loads with ease.

* Node is based on the most widely used language on the planet today: JavaScript
* In Node, almost everything you do is nonblocking, meaning code won’t hold up the processing of other request threads.
* To use Node as a server, you must write some code that then runs on the Node “runtime.”
  + You can write your own web server (Node.js can “create” servers ), but not write a web server from scratch like in many other languages
* Node uses Google’s popular and highly tuned V8 JavaScript engine
  + makes it very high-performance and it can handle a large request load
* Node keeps its core functionality to an absolute minimum and provides extended functionality by way of APIs (initially very simple, yet functional)
  + We can customize and add more functions by importing libraries

**Node Installation**

If you have not already,  please visit the website and install the LTS version of node

* Node.js - <https://nodejs.org/en/>

If you have successfully installed Node, let’s interact with it.

Open your command and type “node” and press enter.

Text

Description automatically generated

You should see a “>” prompt

Node is now listening for your commands in interactive mode

Now let’s type in console.log(”Hello World”)

Text

Description automatically generated

***Please take a screenshot of your output.***

Since Node is a JavaScript runtime, your JavaScript (or compile-to-JavaScript language such as TypeScript) program can be executed. Node can not only execute a line of JavaScript code as we did earlier, but it can also run JavaScript files as well.

**Node’s Partner in Crime: NPM**

NPM, which stands for Node Package Manager, is a companion app that installs alongside Node.

Much of Node’s utility comes from its large package library, which is accessible from the npm command. The NPM registry hosts more than 1.3 million packages of free, reusable Node.js code.

Why would you want to use an NPM package?

In many cases, installing a package via the NPM command line is the fastest and most convenient to get the latest stable version of a module running in your environment, and is typically less work than cloning the source repository and building an installation from the repository.

**Initializing a New NPM/Node Project**

Let’s open Visual Studio Code 3.

Create a file called HOS01\_*yourname*

Open a new terminal and type “npm init”



This will walk you through an interactive, step-by-step process wherein you can enter whichever information is relevant to your project if any. In this HOS, you can just hit Enter on each option to use the default.

Text

Description automatically generated

If you take a look at your files, you can see that the package.json file was created.

**Text

Description automatically generated**

**Graphical user interface, text

Description automatically generated**

In the package.json file, you can see the default information about this project.

As of now, there we have no installed any NPM packages yet for the project. But later, we will see a list of packages, or so-called dependencies (a library that a project needs to function effectively) and important metadata about your project.

Package.json file is powerful, especially when sharing projects with others. As the file holds important metadata about a project and dependencies, others can set up their environment the same as yours.

**Adding Dependencies**

Now let’s add dependencies.

In the terminal, type “npm install express --save”



If you look at the package.json file, you can see that express was added as one of the dependencies

**Text

Description automatically generated**

Also, if you look at the files, you can see that the node\_module file was created

**Graphical user interface, text, application

Description automatically generated**

What is node\_module file?

You do not need to worry too much about the file and what it contains. But it is essentially a directory created by NPM and a way of tracking each package you install locally via package.json.

NPM fetched the Express package from the central repository and determined all the dependencies it needs, and downloaded and installed all of them in the node\_modules directory.

***Open a new terminal, enter following command to commit the section 1 work to git.***

$ git add --all

$ git commit -m "Section 1"

**Let’s Built A Node-Based Web Server**

Now let’s write some server-side code!

Let’s create a file named server.js in your GitHub directly.

Type the following code.

**Graphical user interface, text

Description automatically generated**

* http is a module
  + To import the Node core modules, you just require( ) them by name.
  + One of the methods the object returned is createServer( ). This method creates a web server instance and returns a reference to it.
* The require( ) function returns an object that is essentially the API provided by the module. This object can include methods, attributes, or whatever you want.
* This function handles all incoming HTTP requests. You can do anything you need to there, including such things as the following:
  + Interrogate the incoming request to determine the HTTP method.
  + Parse the request path.
  + Examine header values
* In this code, the createServer() method returns a reference to the web server instance
* The instance contains the .listen( ) method, which the method accepts a port number on which the server should listen to
  + The server will begin listening for requests, and for each request that comes in, it will call the anonymous function passed to .createServer ( ) method.
    - inRequest and inResponse functions (representing the HTTP request and response)
    - All this callback function does is call the .end( ) method on the response object, passing the response you want to send back

Now, if you type “node server.js” in the command, you can start up the server on port 80.

Graphical user interface

Description automatically generated<http://localhost/80>

***Please take a screenshot of your output.***

Let’s define a custom command for NPM to start up the server!

Open the generated package.json file, and in the scripts section, add a new attribute to the object: "start": "node server.js"

**Graphical user interface, text, application, chat or text message

Description automatically generated**

This effectively defines a custom command for NPM.

NPM will look for that start key, take its value, and execute whatever the command is that you provide in it

You can first stop the node server.js by terminating the command.

Now you can spin up the server by typing “npm run start” in the new terminal

**Text

Description automatically generated**

***Please take a screenshot of your output.***

Now visit the localhost port 80 to see the same result. <http://localhost/80>

***Open a new terminal, enter following command to commit the section 1 work to git.***

$ git add --all

$ git commit -m "Section 2"

**Bonus Example**

Let’s install another dependency

Type “npm install request --save”

* The request module will provide our server with an elementary HTTP client for it to use to make remote calls

****

It’s the same code as before, but now with a bit more inside the callback function provided to .createServer ( ) function.

* Firstly, import the request module and give it the name requestModule
* requestModule takes the API and a callback function
  + The URL here is to the World Time API
  + The API passes a URL to the constructor, plus a callback, and a call will be made to the URL, and the provided callback will be executed when the response to that call comes back
  + This form of the URL takes in a time zone, America/New\_York

**Pushing your work to GitHub**

Run the following commands to push your work to the GitHub repository:

Open the terminal from the VSCode by hitting the control + ~ key and type the following command:

>>> git add .

>>> git commit -m “Submission for Module 1--yourname”

>>> git push