CST-391 Activity 0 Guide

Contents

[NodeJS Installation, Tools, and First Applications 2](#_Toc123275110)

[Part 1: Tools Installation and Hello World 2](#_Toc123275111)

[Overview 2](#_Toc123275112)

[Tools Install 2](#_Toc123275113)

[NodeJS "Hello World" Application 2](#_Toc123275114)

[Express "Hello World" Application 3](#_Toc123275115)

[Part 2: Node.js with TypeScript 4](#_Toc123275116)

[Overview 4](#_Toc123275117)

[Execution 4](#_Toc123275118)

[Submission 6](#_Toc123275119)

# NodeJS Installation, Tools, and First Applications

## Part 1: Tools Installation and Hello World

### Overview

In this activity, students will install the development tool and validate installation with a simple "Hello World" application running on Express/NodeJS.

### Tools Install

Execute this assignment according to the following guidelines:

1. Install NodeJS and NPM:

1. Go to the NodeJS download Page located in Class Resources.
2. Click on the Downloads link from the main menu.
3. Download and install the 64-bit version for your operating system.
4. Validate the installation by opening a Terminal window and entering the following commands:
   * Run: node -v   
     Validate the version number that is displayed with the version that was just installed. Take a screenshot.
   * Run: npm -v   
     Validate the version number that is displayed with the version that was just installed. Take a screenshot.

2. Install Microsoft Visual Studio Code:

1. Go to Microsoft Visual Studio Code Download Page located in Class Resources.
2. Click the Download button, download the version for your operating system, and then install the VS Code per the installation instructions.

### NodeJS "Hello World" Application

1. Create a console application similar to the first application you wrote in Java.
2. Create a 'hello' folder in a location on your computer where your projects for this course will be stored.
3. Open the 'hello' folder in VS Code. These instructions are general because there are several ways to manage folders and workspaces in VS Code. Find a workspace management technique that works for you.
4. In the 'hello' folder, create a file named 'app.js'. Enter the following JavaScript code:

console.log('Hello World!');

1. Open a 'Terminal' window in VS Code. This terminal should open in the same folder as 'app.js'
2. Run the following commands in the integrated Terminal to start the NodeJS server:

node app.js.

1. Take a screenshot.

### Express "Hello World" Application

1. Our first Express/NodeJS task will be the classic "Hello World" application. This will send a greeting to a browser, or any HTTP client.

1. Express is a NodeJS library that allows us to create NodeJS web servers. During this course, we will be developing a real server in the next activity and connecting to that server from both an Angular and a ReactJS application. For this activity, we will develop a simple server that sends a client a message.
2. Create a 'helloex' folder and open that folder in VS Code.
3. NodeJS applications that use libraries need to use npm (node package manager) to manage those libraries. These steps will initialize the application and install the Express library.
   1. Open a terminal in VS Code, when prompted by npm init, accept all the defaults:

npm init

npm install express

1. Create a file 'app.js'. Enter the following JS code:

const express = require('express');

const app = express();

const port = 3000;

app.get('/', (req, res) => res.send('Hello World!'));

app.listen(port, () => console.log(`Example app listening on port ${port}!`));

1. Run the following commands in the integrated Terminal to start the NodeJS server:

node app.js.

1. Enter Ctrl-C in the terminal to stop the server.
2. Change the response string in app.js and start the NodeJS server.
3. Open a browser at http://localhost:3000/ to view the Hello World Express page. Take a screenshot.
4. Stop the server.

#### Node Monitor (nodemon) A Convenient Utility

The nodemon utility will rerun your NodeJS code when it detects a change in the source code.

1. Install *nodemon* using the following command in the Terminal (**Note:** on a Mac, you may need to run as sudo):

npm install -g nodemon

The -g means to install the library globally, making this utility available to all NodeJS projects. The default is to install in 'node\_modules', making the utility or library available only to the current project.

1. Start the NodeJS server using the following command:

nodemon app.js

1. Change the response string in app.js, refresh your browser, and validate that the code changes took effect.
2. Take a screenshot.

## Part 2: Node.js with TypeScript

### Overview

JavaScript was used in Part 1 of this activity. In Part 2 of this activity, you will learn how to create a node application using TypeScript. The activity will form the starter code for the next activity where we will build a TypeScript Node.js MusicAPI application. We will use this API for the rest of the course.

### Execution

Execute this assignment according to the following guidelines:

1. Make a directory called MusicAPI. Open that directory in VS Code.
2. Run
   1. npm init
      1. Accept all the defaults.
3. Run
   1. npm i express.
      1. 'i' is short for 'install'. We use both in this exercise because you will see examples of both in tutorials. You will also see --save in tutorials. This switch does no harm, but it is no longer necessary.
4. Run
   1. npm i --save-dev typescript @types/express
   2. The --save-dev switch indicates that the code you are installing is a development tool. The code is not shipped with the software. Instead, it is a tool used in the development cycle. Without --save-dev, the developer tool will be bundled with the application.
   3. The specific tool @types/express helps add types to traditional JavaScript libraries with no concept of strong types.
5. Create a folder called 'src' in the root of your application folder. The 'src' folder is the root of your source code. The pattern:
   1. Configuration files in the root.
   2. Source code in 'src'
6. Create an 'app.ts' file (notice the TypeScript file extension) in the MusicAPI\src folder. Copy the following code into that file:

import express, { Request, Response } from 'express';

const app = express();

const port = 3000;

// Make sure you understand the following line of code.

app.get('/', (req: Request, res: Response) => {

res.send('Hello World from TypeScript!');

});

app.listen(port, () => {

console.log(`Example app listening at http://localhost:${port}`)

});

1. Install the TypeScript compiler globally:

npm install typescript@latest -g

1. TypeScript should always be run with a configuration file; it wakes up in stupid mode. Generate a tsconfig.json file in the root of your application, next to package.json:

tsc --init

1. You can run the TypeScript compiler directly with the 'tsc' command. This will generate the needed app.js file. However, we aren't going to take that path. Instead, we are going to use ts-node, which does two steps for us:
   1. Compiles TypeScript files to JavaScript
   2. Executes node with the JavaScript version of the file
2. Install ts-node:

npm install -g ts-node

1. Run your application:

ts-node app.ts

**Note:** Some Windows users cannot run 'tsc' by default. To fix this:

* 1. Close VS Code
  2. Run PowerShell as an admin:
     1. In PowerShell run
     2. Set-ExecutionPolicy -ExecutionPolicy RemoteSigned
  3. Open VS Code.

1. Open a browser at http://localhost:3000/ to view the Hello World Express page. Take a screenshot.
2. The nodemon utility is smart enough to recognize TypeScript. You can still run
   1. nodemon app.ts
3. Believe it! You have a Node.js web service written in TypeScript. In the next activity, we will take this starter code and expand it into the MusicAPI.
4. Take a screenshot of your running application.
   1. Add a descriptive comment to every line of app.ts
      1. Take a screenshot.

## Submission

Submit the following as directed by the instructor in a Microsoft Word document:

1. Caption all screenshots with respective summaries for items listed below. Available points will be equally applicable to all screenshots.
   1. node -v
   2. npm -v
   3. Hello World console application
   4. Hello World in the browser.
   5. Hello World in the browser with nodemon utility
   6. Hello World Typescript in the browser.
   7. Commented app.ts