CST-391 Activity 0: Tools Installation and Initial Applications

Kaya Nelson

College of Science, Engineering, and Technology, Grand Canyon

University Course Number: CST-91

Professor Estey

3/8/2025

GitHub Link:

Kdeshun/CST-391

📂 Project Structure

This repository is organized into three sub-projects as part of Activity 0, each serving a distinct purpose:

1. **hello** → This is a fundamental Node.js console application that demonstrates the basic functionality of Node.js by printing "Hello World" to the terminal. It serves as an introductory example for those new to Node.js, showcasing how to set up a simple project and execute scripts.
2. **helloex** → This sub-project features an Express.js application that brings the "Hello World" concept to the web. It runs a lightweight server that responds to HTTP requests, displaying "Hello World" in a web browser. This project highlights the use of Express.js for creating web applications and handling routes effectively.
3. **MusicAPI** → A more advanced project, this is a web service built using TypeScript and Node.js. It provides a robust framework for developing music-related functionalities, such as managing albums and artists. This project emphasizes the use of TypeScript for type safety and enhanced development experience, making it suitable for building scalable and maintainable applications.

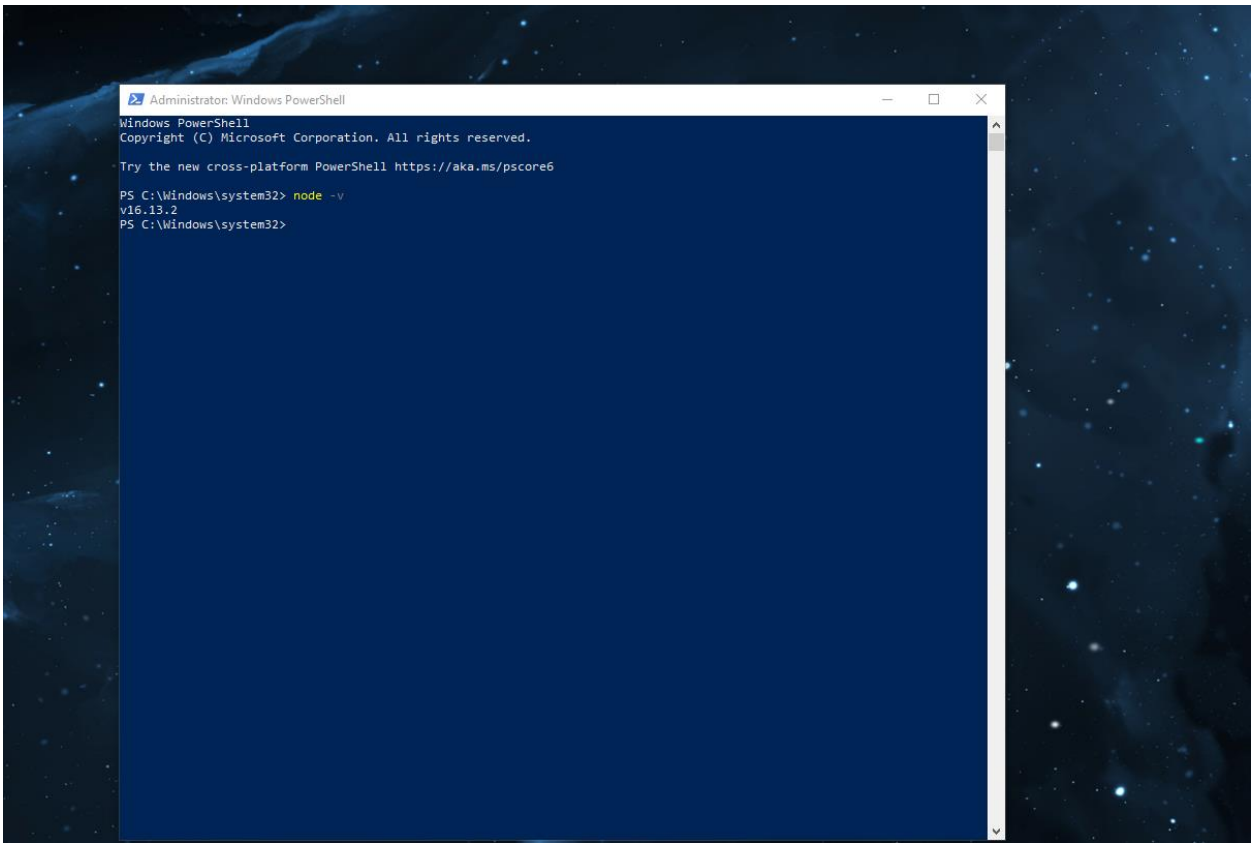
NodeJS and NPM Installation

Verify Node.js Installation

Run the command:

```
node -v
```

Screenshot:

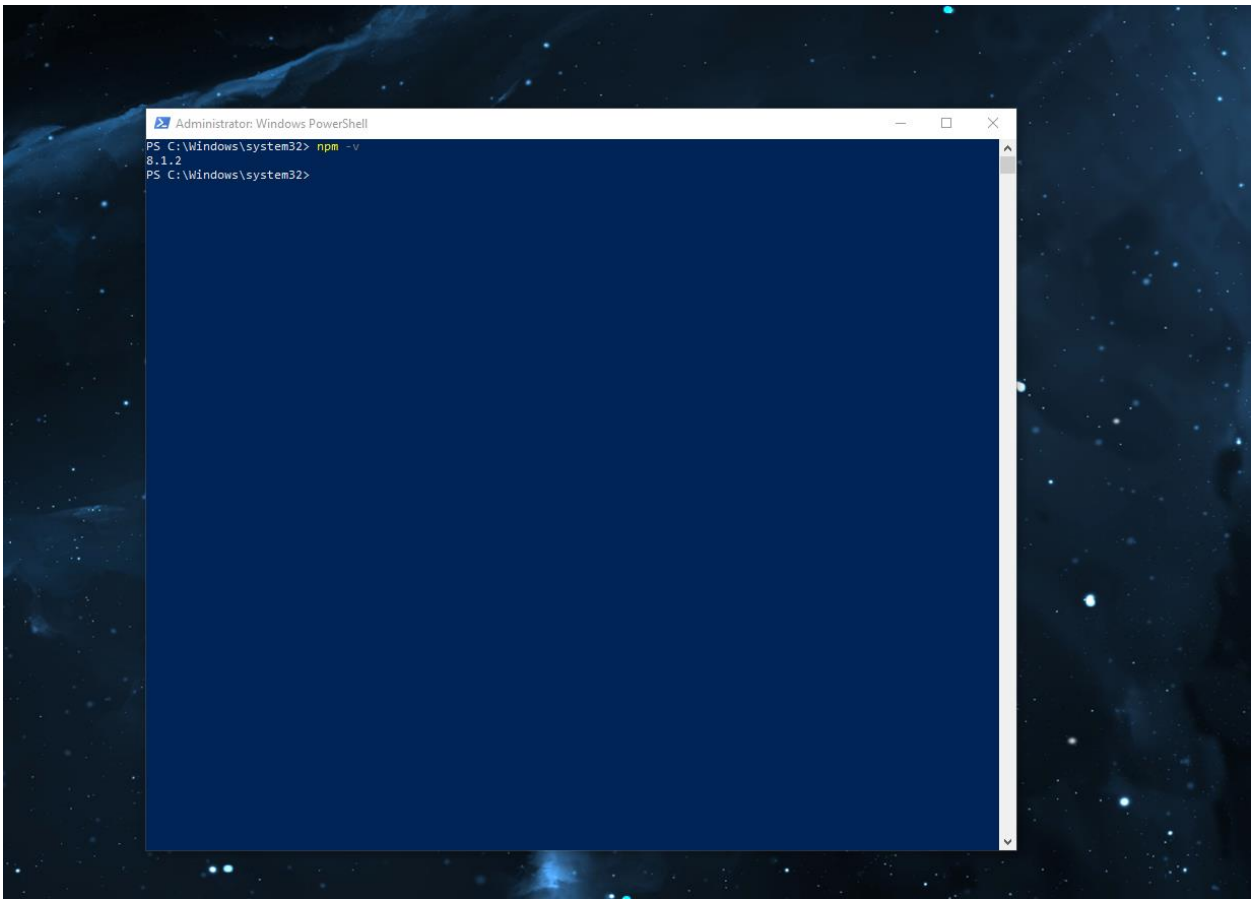


Verify NPM Installation

Run the command:

npm -v

Screenshot:



Hello World Console Application (hello/)

app.js Code

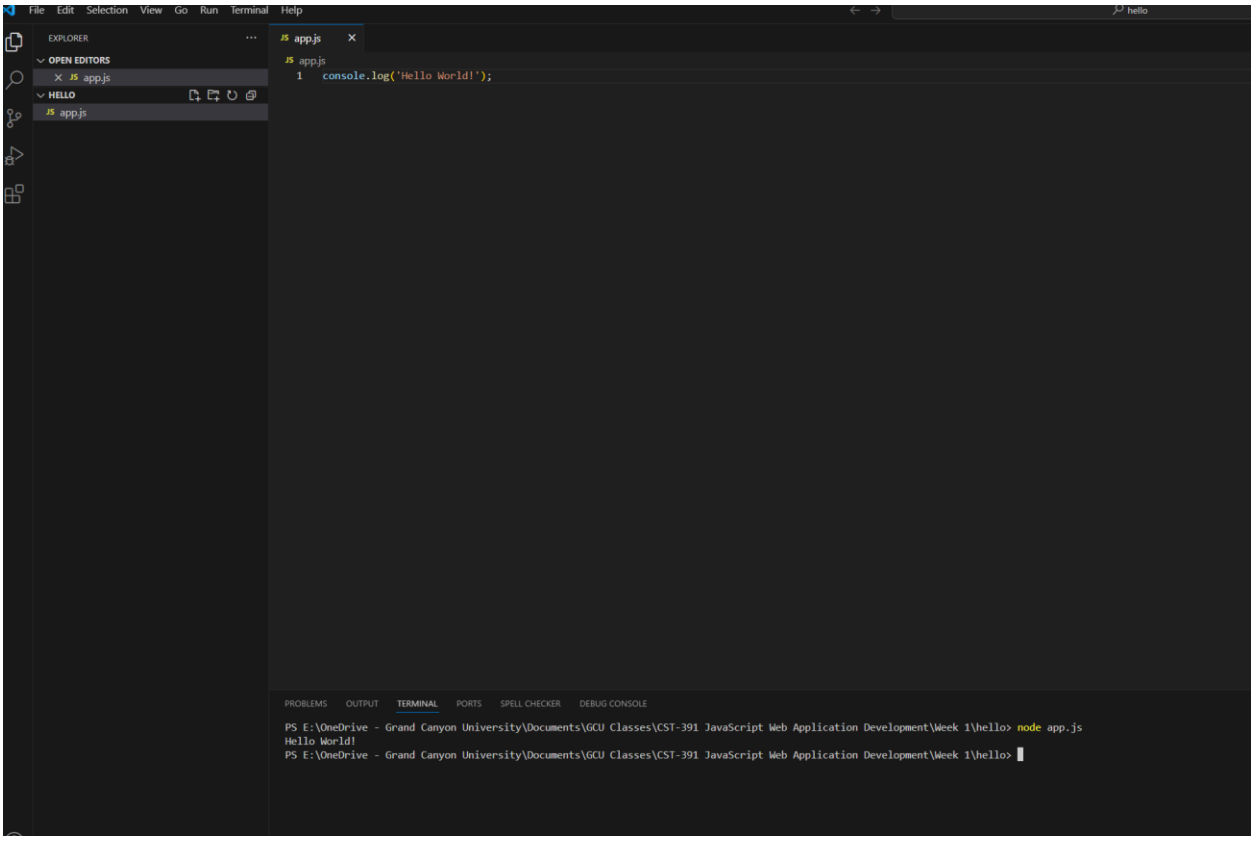
Create a file named app.js inside the hello/ folder with the following content:

console.log('Hello World!');

Run the Application

node app.js

Screenshot:



Express "Hello World" Application (helloex/)

Install Express

Navigate to the helloex/ folder and run:

npm init -y

npm install express

app.js Code

Create app.js inside helloex/:

const express = require('express');

const app = express();

const port = 3000;

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

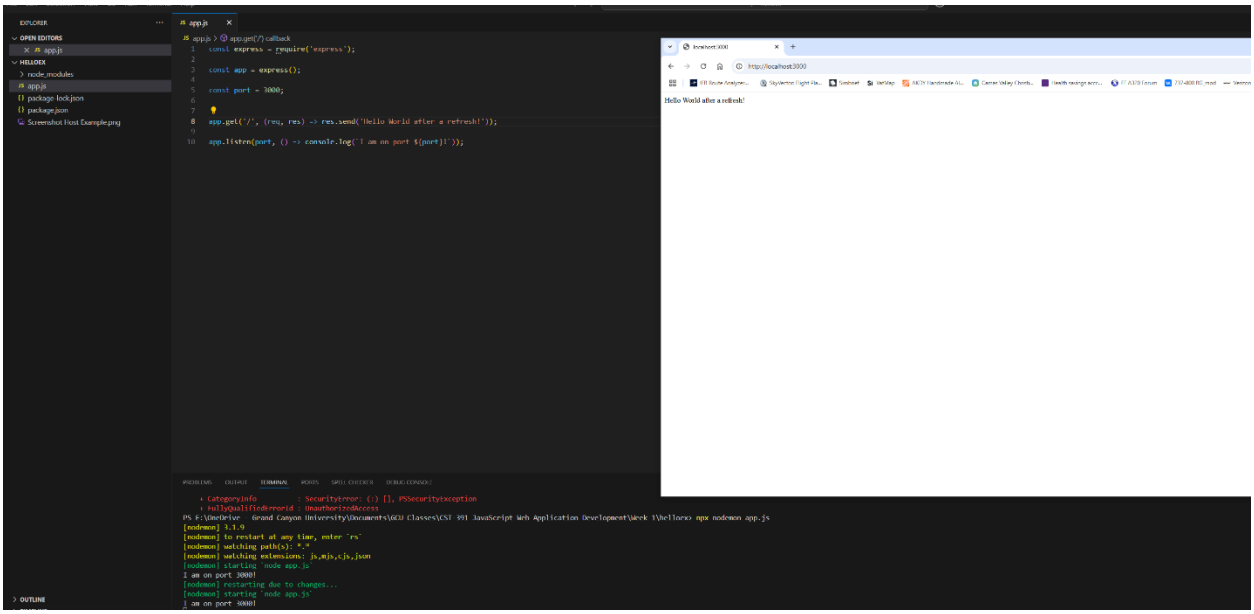
app.listen(port, () => console.log(` Example app listening on port ${port}! `));

Run the Server

node app.js

Open a browser and visit: http://localhost:3000

Screenshot:



Nodemon Utility

Install nodemon

npm install -g nodemon

Run Server with nodemon

nodemon app.js

Modify app.js, refresh the browser, and check if it updates automatically.

Screenshot:

TypeScript "Hello World" API (MusicAPI/)

Set Up the Project

Navigate to MusicAPI/ and run:

```
npm init -y
npm install express
npm install --save-dev typescript @types/express
```

Create src/app.ts

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

const app = express();
const port = 3000;

app.get('/', (req: Request, res: Response) => {
  res.send('Hello World from TypeScript!');
});

app.listen(port, () => {
  console.log(` Example app listening at http://localhost:${port}` );
});
```

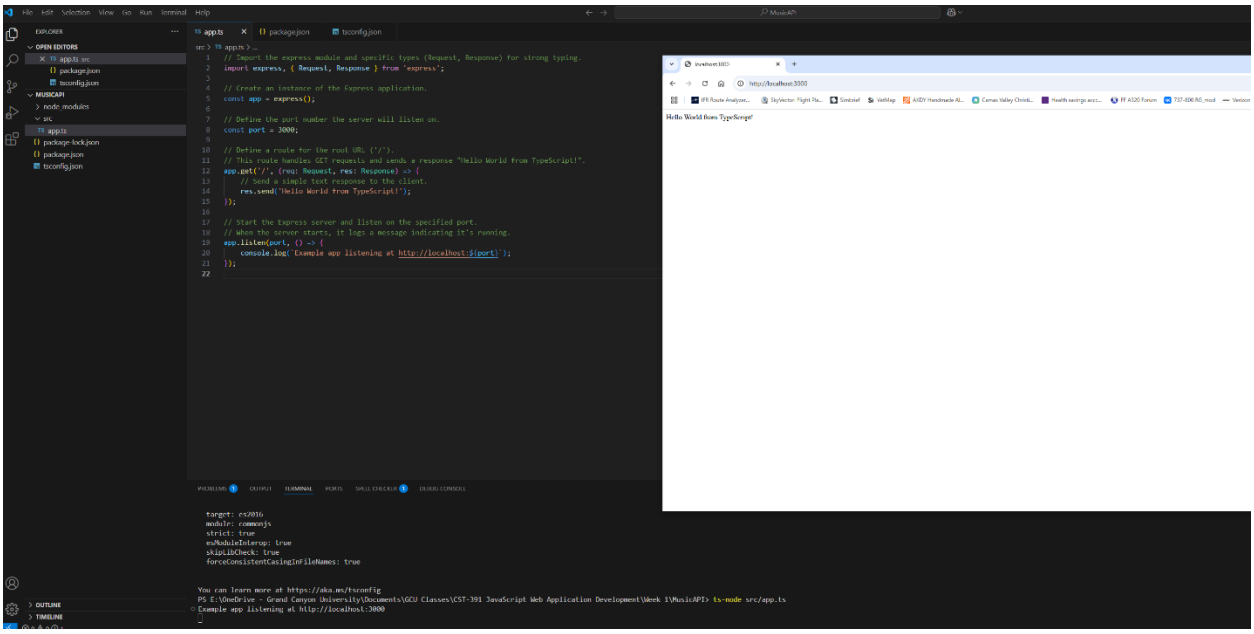
Install TypeScript Compiler

```
npm install typescript@latest -g
tsc --init  # Generates tsconfig.json
```

Run the TypeScript Application

```
ts-node src/app.ts
```

Screenshot:



Commented app.ts File

Here's the **fully documented TypeScript app**:

```
// Import Express framework and Request, Response types for strong typing
import express, { Request, Response } from 'express';

// Create an Express app instance
const app = express();

// Define the port the server will listen on
const port = 3000;

// Define a GET endpoint that responds with "Hello World from TypeScript!"
app.get('/', (req: Request, res: Response) => {
  // Send a response back to the client
```

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


// Start the Express server and listen on the defined port
app.listen(port, () => {
  console.log(` Example app listening at http://localhost:${port}` );
});
```

Screenshot: