1. Introduction to Node.js?

* Node.js is an open source, cross platform runtime environment and library of JS.
* Used for server-side programming
* Primarily developed for non-blocking, event-driven servers
* It is built on Google Chrome’s V8 JS engine
  1. What is node.js?
* Node.js app runs in a single process, without creating a new thread for every request
* It provides set of asynchronous I/O (primitives = ancient) in its standard library to prevent JS code from blocking
* engine
  1. Why use node.js?
* Due to huge community
* V8 runs JS code lightning-fast
* Node.js applications are very scalable and maintainable
* We can create desktop app, software as a service, mobile application due to cross platform support
* Node.js is perfect for real-time and data-intensive application
* Huge collection of node packages (npm)
  1. History of node.js
* Written by Ryan Dahl in 2009,
* Developed for Linux and macOS
* Later sponsored by Joyent
  1. Node.js vs browser
* We mostly interact in Browser with DOM and cookies, which don’t exist in Node.js
* In Browser we don’t have nice APIs that Node.js provides through its modules
* In Browser we cannot choose the version but in Nod.js we can control the environment
* Browser uses Babel to transform JS into latest JS code but Node.js supports all modern JS code (ES6+ ,2015>)
* Node.js supports commonJS and ES6 module system
* We use import in Browser and require and import in Node.js
  1. Running node.js code

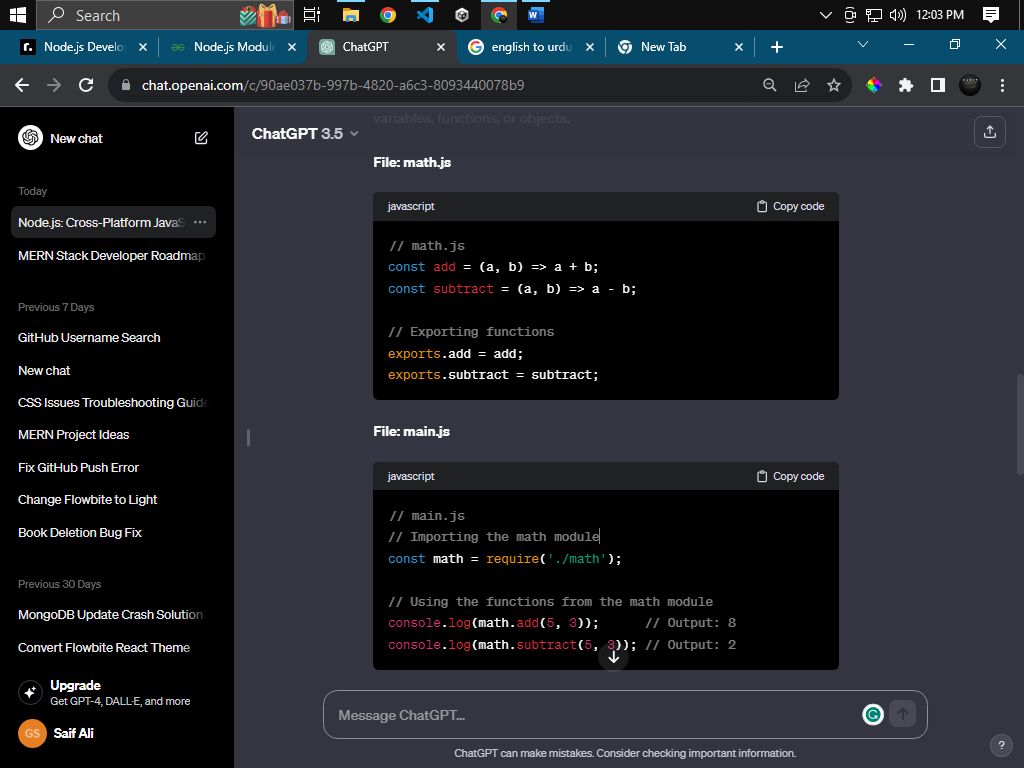
At the same directory open command prompt and run the file with this code

“node app.js” if the file name is app

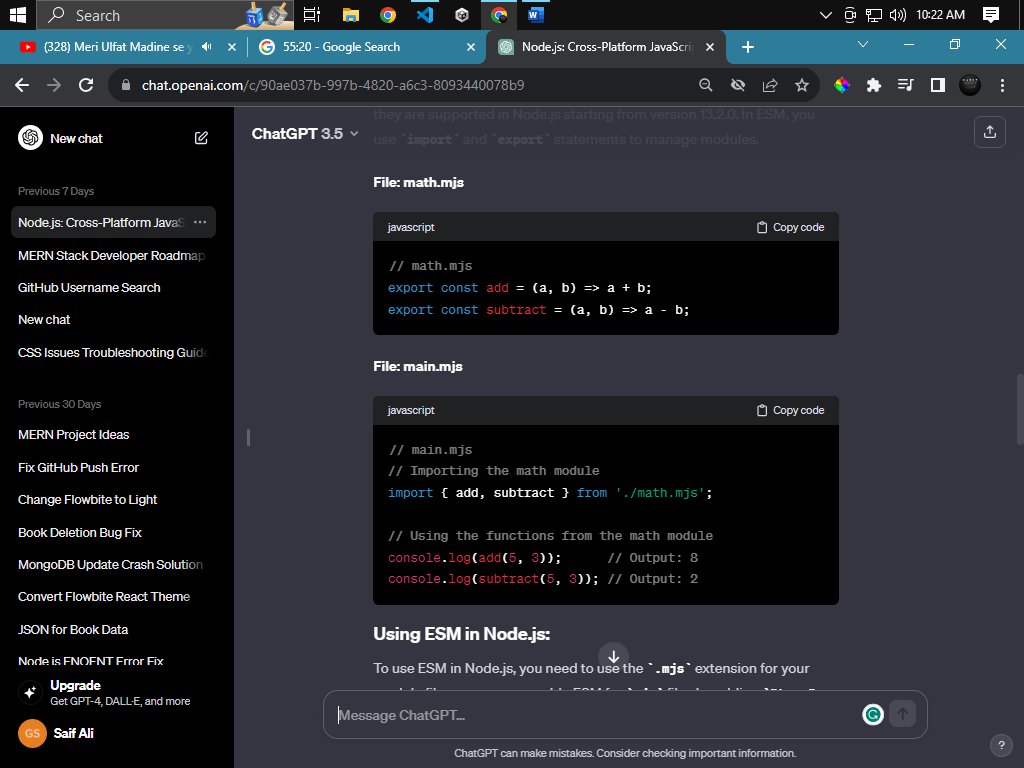
And can write some code and run at the command prompt using “-e”

“node -e “console.log(‘Code is running using -e command in command prompt’)””

1. Modules

* Chunk of codes to maintain, organize and reuse as needed
* They are JS files written by other developers
* Some built-in modules that are part of platform and come with Node installation (http, fs, path, querystring, process, os and path)
  1. Common JS
* Traditional modules those are exported by exports.module name and imported by using require()
* Not accessible to others by default

* 1. ESM
* Advanced modules, introduced in version 13
* Instead of exports.module we use export keywords
* And import in another file like react (import)
* We can assign any name if export as default
* Import and export name must be the same



* 1. Creating Custom Modules

Have done in a and b.

* 1. Keywords [global]
     1. Global objects

Global, console, process, buffer, setTimeout, setInterval, clearTimeout, clearIntervel

* + 1. Global Keywords

Var, let, const, function, return, if, else, switch, case, default, while, for, do-while, module, exports, import, require, try, catch, finally, async, await, new, this, typeof

1. NPM
   1. npx

* Node Package Manager
* Online repos for publishing open-source Node.js projects
* Worlds largest software registry
* Run packages without downloading using npx.
* It is a command line utility to interact with the installed package
* Use to manage and share software packages or libraries in JS code.
* It simplifies the process of installing, updating, and managing dependencies in node.js project
* Developer uses npm to install packages (libraries or modules)
* Npm ensures that the correct version of package is installed to avoid the conflict
* “npm install” is used to install dependencies and “npm init” to create for “package.json” file
* We can set the version in “package.json” to allow the specific version
* We can create custom script in “script” to do specific tasks
  1. Global Installation
  2. Local Installation
  3. Updating Packages
  4. Using installed Packages
  5. Running Scripts
  6. npm workspace (advanced)
  7. Private Packages(advanced)
  8. Public Packages (advanced)

1. Error Handling
   1. Uncaught Exceptions
   2. Call Stacks and Stack Trace
   3. Types of Errors
      1. JS Errors (intermediate)
      2. System Errors (intermediate)
      3. User Specified Errors (intermediate)
      4. Assertion Error (intermediate)
      5. Handling Async Errors
2. Asynchronous Programming
   * 1. Promises (intermediate)
     2. Async/await (intermediate)
     3. Callbacks (intermediate)
     4. setTimeout (intermediate)
     5. setInterval (intermediate)
     6. setImmediate (intermediate)
     7. process.nextTick (intermediate)
3. working With Files:
   1. fs module
   2. path module
   3. process.cwd()
   4. \_\_dirname
   5. \_\_filename
   6. Glob
   7. Globby
   8. Fs-extra
   9. Chokidar
4. Command Line Apps
   1. Process.env
   2. Dotenv package
   3. Environment Variables
   4. Existing and Exit Codes
   5. Printing Output
   6. Taking Input
   7. Command Line Args
5. Working With APIs
   1. HTTP server
   2. Making API calls
   3. Authentication
6. Keeping Application Running
   1. Nodemon package
7. Templating Engines
   1. Ejs
   2. Pug
   3. Marko
8. Working With Database
   1. Mongoose package
   2. Prisma package
   3. Knex package
   4. typeORM package
   5. sequelize package
   6. native driver
9. Testing
   1. Jest
   2. Mocha
   3. Cypress
10. Logging
    1. Winston
    2. Margon
11. Keeping App Running (intermediate)
    1. Pm2 package
    2. Forever package
    3. Nohup
12. Threads (advanced)
    1. Child process
    2. Cluster
    3. Worker threads
13. Streams (advanced)
14. More Debugging (advanced)
    1. Garbage Collection
    2. Memory leaks
    3. Node-inspect
    4. Using APM
15. Common Build-in Modules (intermediate)
    1. Fs
    2. Os
    3. Net
    4. Path
    5. Url
    6. Events
    7. http
    8. console
    9. assert
    10. process
    11. cluster
    12. perf\_hooks
    13. crypto
    14. buffer