Help to complete the tasks of this exercise can be found on the chapters from ch. 0 ” Setting Up Node.js and the JavaScript Engine” to ch. 2 “Running a Node.js Application” of our course book “Get Programming with Node.js” by Jonathan Wexler. The aims of the exercise are to teach to create a Node.js development environment, give you basic understanding of Node.js, and to help you to run your first Node.js application.

Embed your theory answers, drawings, codes and screenshots directly into this document. Always immediately after the relevant question. Return the document into your return box in itsLearning by the deadline.

It’s also recommendable to use Internet sources to supplement the information provided by the course book.

The maximum number of points you can earn from this exercise is 10.

Tasks:

1. Prepare a Node.js development environment for yourself. (2 points)
   * If necessary, install the latest version of Oracle VirtualBox virtual machine on your computer.
   * Install the latest version of Node.js.

A black background with white text

Description automatically generated

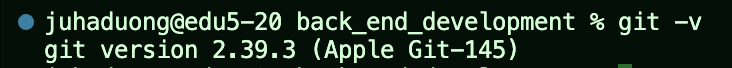
* + If necessary, add node.js to the operating system’s PATH environmental variable.
  + Install Visual Studio Code editor.

Already installed (check picture below)

A screenshot of a computer

Description automatically generated

* + Install Git



Please note that you can also prepare the development environment directly on your own computer without installing Oracle VirtualBox and without downloading the virtual image.

1. Explain with your own words what Node.js is. (1 point)

Node.js is an open-source platform that allows you to build fast and scalable network applications. It's based on the JavaScript runtime engine V8, which was originally developed by Google for its Chrome browser. This means you can use the same language (JavaScript) to build both front-end and back-end components of your web applications.

1. Explain shortly the concept of asynchronous programming. (1 point)

Instead of waiting for one task to finish before starting the next, a computer can work on multiple tasks at once. This makes it faster and more efficient, especially when dealing with lots of tasks. This is a concept of asynchronous programming.

1. Node.js and its event loop (4 \* 0,5 = 2 points, you can leave one unanswered)
   * Explain with your own words what Node.js’ event loop is.

The event loop is a mechanism that allows Node.js to handle multiple tasks efficiently by delegating complex tasks to the operating system and using callbacks to be notified when they're finished.

* + What is executed in the event loop, what is delegated?

Simple tasks like calculations, data manipulation and small functions are executed and more complex tasks in example, reading and writing files, network request (http requests), database interactions are delegated to operating system

* + What does it mean that Node.js is event-driven?

It means that the application's execution flow is determined by events that occur. Instead of following a linear path, the code waits for events to happen and then responds to them.

* + What does the event loop do when a delegated task is finished?

The event loop continuously checks if the delegated task has been completed by the operating system. Once the task is finished, the event loop adds the corresponding callback function to the event queue. This callback function was specified when the task was originally delegated. When it’s callback’s turn, the event loop removes it from the queue and executes it.

* + What does it mean that Node.js is single threaded?

While Node.js has only one main thread, it delegates time-consuming tasks like I/O operations (e.g., reading files, making network requests) to the operating system. This allows the main thread to continue processing other tasks without being blocked.

1. Explain the types of application where Node.js is efficient. When it might not be efficient at all? (2 \* 0,5 = 1 point)

Node.js is a great tool for building applications that need to update information quickly, like online games or chat apps. It's also good for handling tasks that involve waiting for input or output, such as reading from files or making network requests. And it can handle a lot of requests at once, making it suitable for websites with many visitors.

However, Node.js might not be the best choice for applications that require complex calculations or need to pause and wait for a long time. In these cases, other tools might be more efficient.

1. Run your first Node.js application “Hello World!”. Write the necessary code into a file. Display the greeting on the terminal window. (1 point)

A screen shot of a computer

Description automatically generated

A close up of a sign

Description automatically generated

1. Create an array containing the names of some of your Favorite songs. The length of the array is six. Create a small Node.js application that randomly selects one of the songs and then displays it at the terminal window. (2 points)

A screen shot of a computer program

Description automatically generated

A screenshot of a computer program

Description automatically generated