01 - Introduction to Node.js - PT

Node.js is an open-source, cross-platform environment for developing server-side web applications. Node.js applications are written in JavaScript and can run on a wide variety of operating systems. Node.js is based on an event-driven architecture and a non-blocking input/output API designed to optimize throughput and scalability for real-time web applications. For a long time, available web development frameworks have all been based on a stateless model. A stateless model is where data generated in a session (such as information about user settings and events that have occurred) is not persisted for use in the next session by that same user.

With Node.js there is finally a way for web applications to have real-time bidirectional connections, where both the client and server can initiate communication, allowing them to exchange data freely.

WHY USE NODE.JS

The real value of Node.js will be explored in the next chapters, but what makes this framework so popular?

Over the years, most applications have been based on a stateless request-response framework. In such applications, it is up to the developer to ensure that the correct code is implemented to ensure that the state of the web session is maintained while the user is working with the system. But with Node.js web applications, it is now possible to work in real-time and have two-way communication. State is maintained and either the client or the server can initiate communication.

NODE.JS FEATURES

Some of the main features of Node.js are:

- Event-driven asynchronous I/O helps in handling concurrent requests – this is probably the strongest suit of Node.js. This feature basically means that if a request is received by Node for some I/O operation, it will execute the operation in the background and continue processing other requests. This is quite different from other programming languages.

Let's look at the following code:

```
var fs = require('fs');

fs.readFile("teste.txt",function(error,data)
{
    console.log("A leitura do ficheiro foi terminada com sucesso.");
});
```

The code snippet above analyzes the reading of a file called test.txt. In other languages, the next line of processing would only happen when the entire file has been read. In the case of Node.js, the important part of the code to note is the function declaration ('function(error,data)'). This is known as a “callback” function. What happens is that the file reading operation will start in the background while other processing can occur simultaneously. When the read operation is complete, this anonymous function will be called and the text “The file reading was completed successfully” will be written to the console log.

- Node uses the JavaScript Runtime V8, which is also used by Google Chrome. Node has a wrapper over the JavaScript engine that makes it much faster, and therefore processing requests in Node also becomes faster.

- Handling concurrent requests – Another important feature of Node is the ability to handle concurrent connections with minimal overhead on a single process.
- Node.js library used JavaScript – This is another important aspect of Node.js development. A large part of the development community is already experienced in JavaScript and hence Node.js development becomes easier for programmers familiar with the language.

WHO USES NODE.JS?

Node.js is used by several large companies. Below is a list of some of them:

- Paypal - Several Paypal sites have begun transitioning to Node.js. LinkedIn -
- LinkedIn is using Node.js to power its mobile servers, which power its iPhone, Android, and Mobile Web products.
- Mozilla has implemented Node.js to support APIs with over half a billion installations.
- Ebay has implemented hosting of its HTTP API service in Node.js

WHEN TO USE NODE.JS?

Node.js is suitable for use in streaming or event-driven real-time applications, such as:

- Chat applications
- Game Servers - Fast, high-performance servers that need to process thousands of requests at the same time.
- It is suitable for collaborative environments – for example, document management environments. In a document management environment, we may have multiple people publishing documents and making constant changes, checking documents in and out. Therefore, Node.js is good for such environments because the event loop in Node.js can be triggered whenever documents are changed.
- Ad Servers – In these systems we can have thousands of requests to pull ads from the central server and Node.js can be an ideal framework to handle these requests.
- Streaming Servers – Another ideal scenario to use Node is for streaming media servers where clients have requests to pull different media content from that server. Node.js is good when you need high levels of concurrency but less dedicated CPU time.

WHEN NOT TO USE NODE.JS?
Node.js can be used for various applications with various purposes, the only scenario where it should not be used is if there are long processing times required by the application.

Node is structured to be single threaded. If some application is required to perform some long running computations in the background and if the server is performing some computation then it cannot process any other request.

As mentioned earlier, Node.js is better when processing needs less dedicated CPU time.

IS NODE.JS A “RUNTIME SYSTEM”? WHAT DOES THIS MEAN?

A runtime system is a platform on which a software program runs. It is essentially an environment that encompasses the collection of software and hardware that allows an application to run. Node.Js is a runtime system because it provides the environment necessary for applications to run on it, and no additional code or hardware is required.

Because Node.Js uses JavaScript, it is a “runtime system” that provides a framework in which you can use JavaScript to create and run your programs. The JavaScript programming language (which is quite popular) is then automatically translated into machine code so that the hardware can execute the program in real time.

It is an efficient system, making it the preferred choice for many software and technology companies.

WHAT IS A MICRO-SERVICES ARCHITECTURE AND WHY USE NODE.JS IN THIS CONTEXT?

Microservices architecture is a style or method of software development where single-function modules predominate. In essence, it is a way of developing modules that can be repurposed from one program or application to another.

When we build an application, it is a collection of modules that have been thoroughly tested and well maintained. Modules are typically built around different features and functionalities and are then loosely coupled with other modules when deployed as part of a program. We can use Node.Js in a microservices architecture as the language of choice for one, some, or all of the microservices (or modules).

The advantage of microservices is that we can use the best language for the specific microservice, and where highly scalable and fast programs are needed, Node.Js is one of the main competitors.

DOWNLOAD AND INSTALL NODE.JS?

To start building your Node.js applications, the first step is to install the Node.js framework. The Node.js framework is available for a variety of operating systems, from Windows to Ubuntu and OS X. Once the Node.js framework is installed, you can start building your first applications.

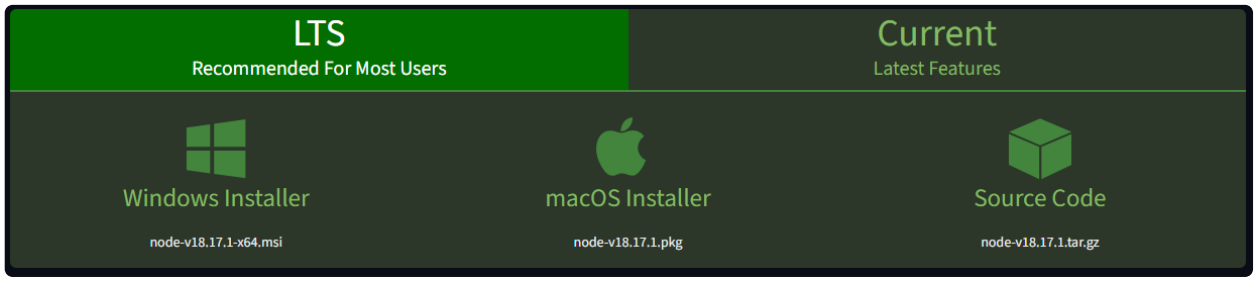
Node.js also has the ability to incorporate external functionality or extended functionality using custom modules. These modules must be installed separately. An example of a module is the MongoDB module that allows you to work with MongoDB databases from within your Node.js application.

HOW TO INSTALL NODE.JS?

The first steps to using Node.js involve installing its libraries.

To install Node.js, follow the steps below:

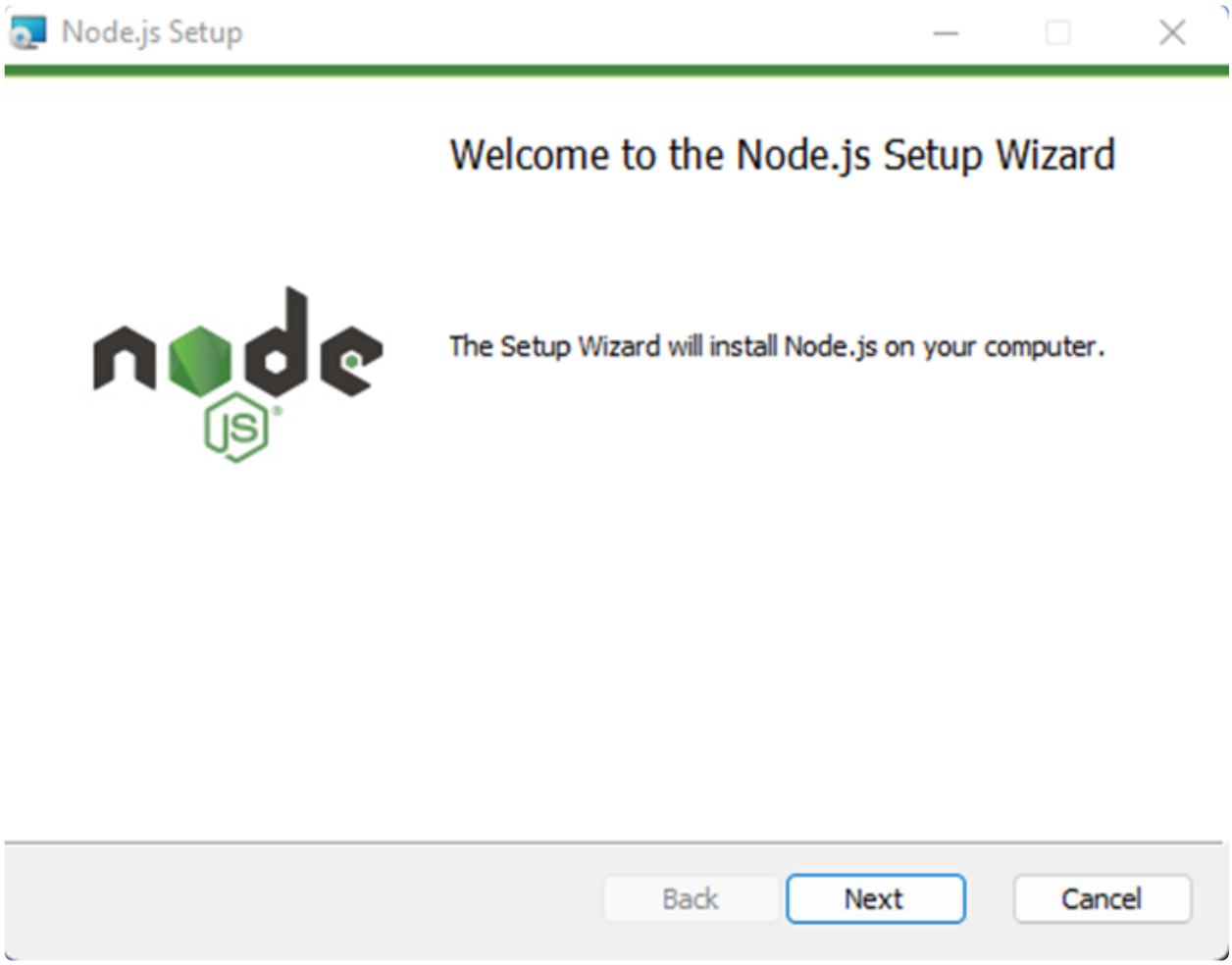
1.Access the websitehttps://nodejs.org/en/download/and download the necessary files. In our case, we will download the 64-bit configuration files for Node.js.



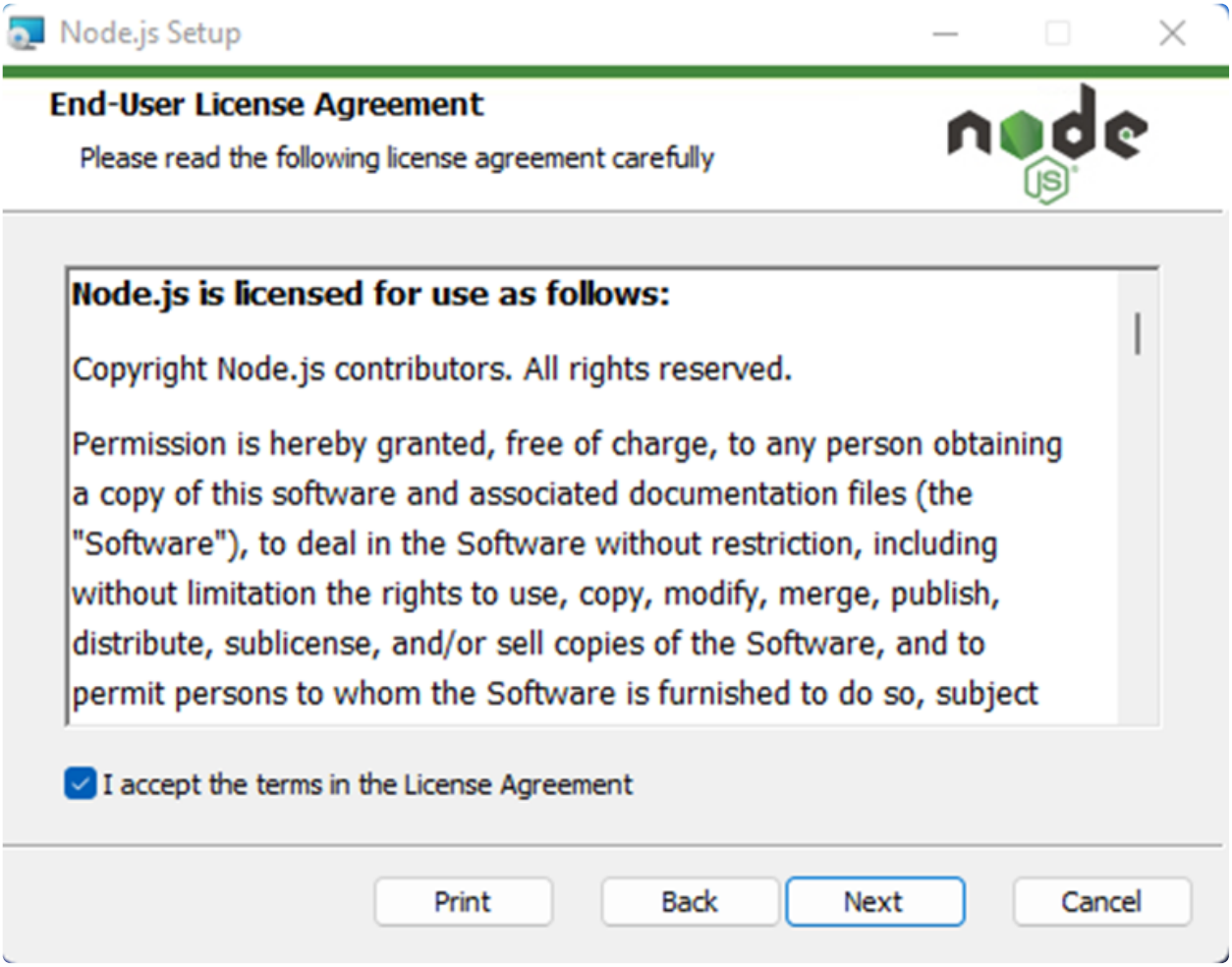
The first steps to using Node.js involve installing its libraries.

To install Node.js, follow the steps below:

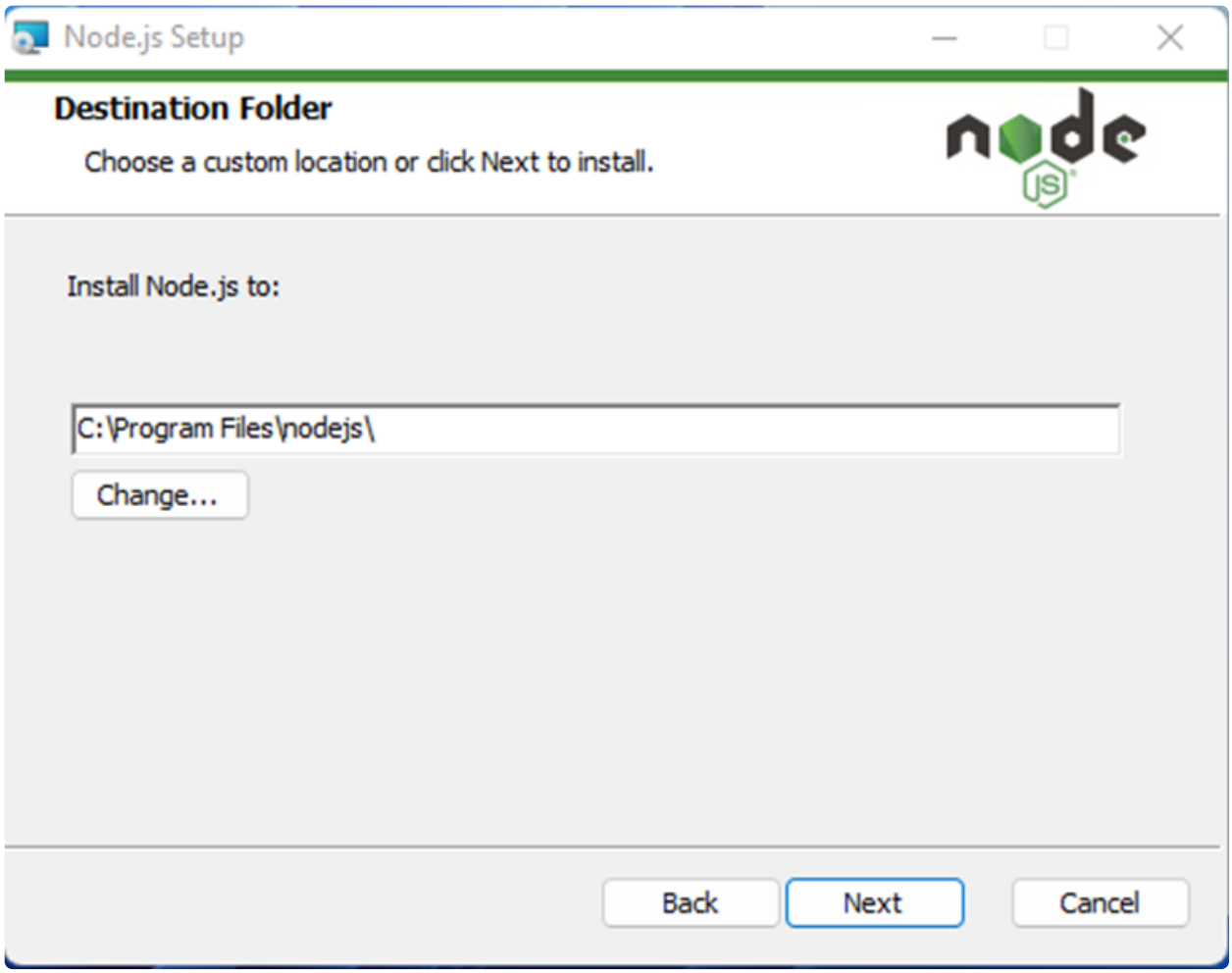
2. Double click on the installation package to start the installation.



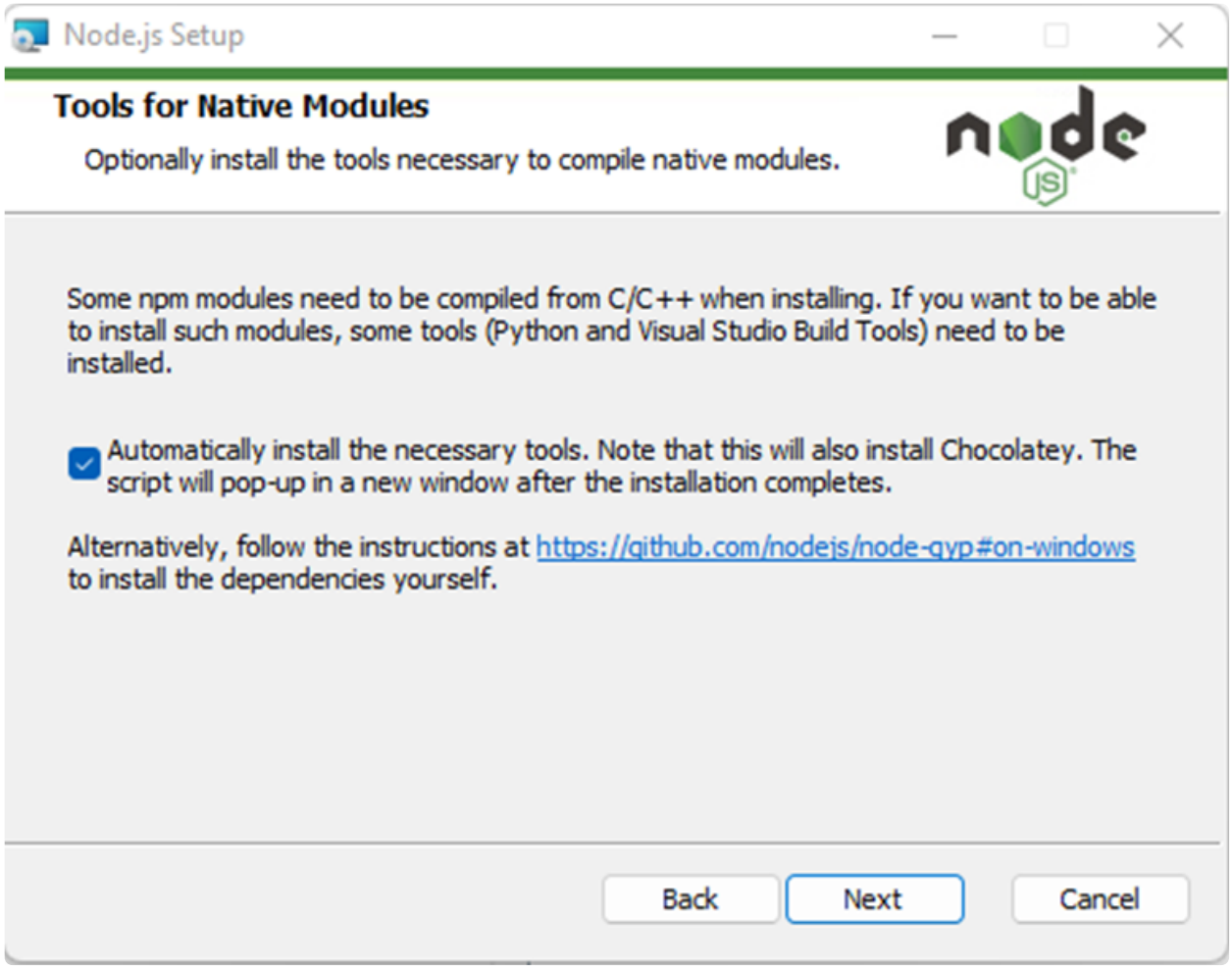
3.Click to accept the license and then click the “Next” button.



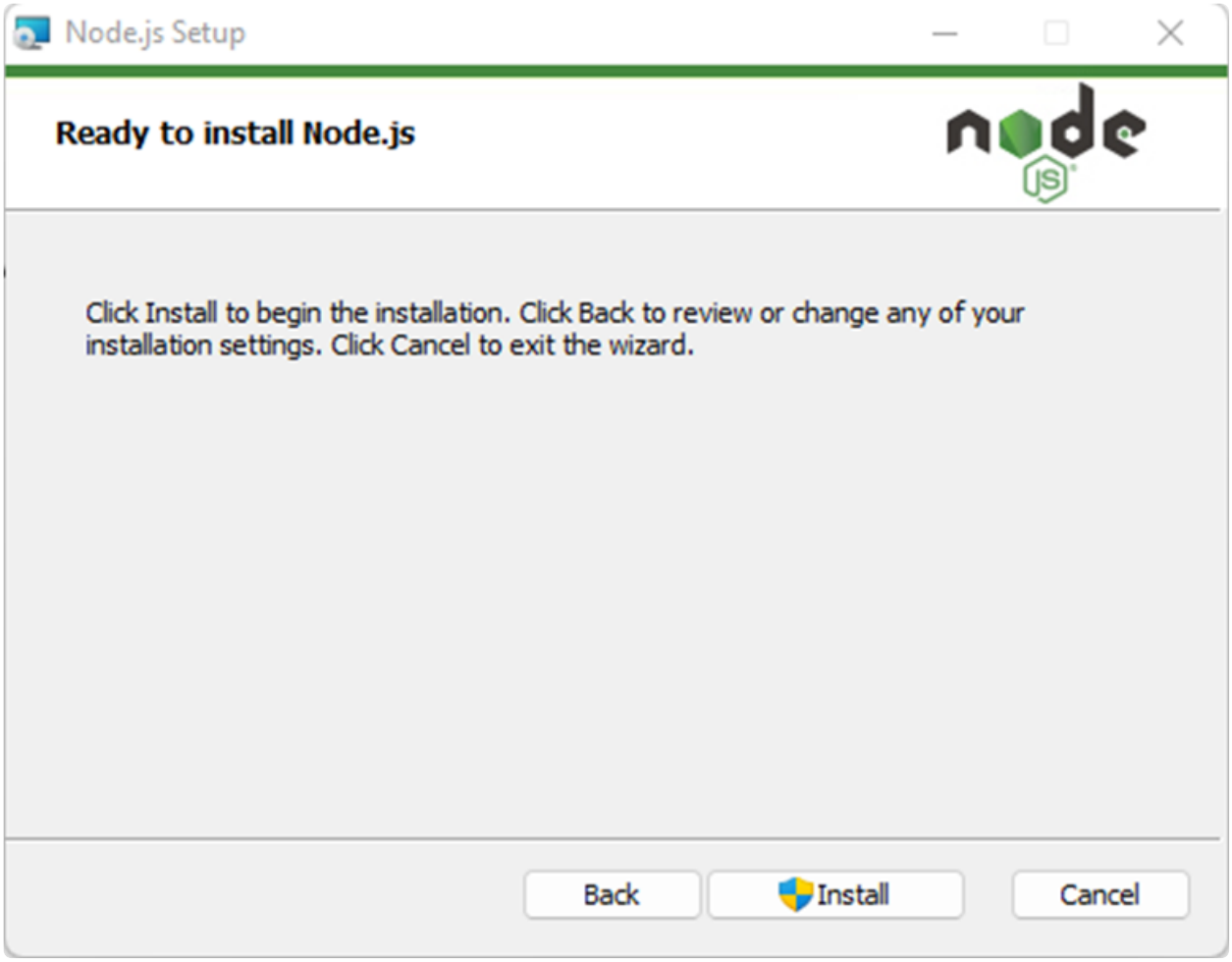
4. Choose the destination folder for the installation (it is recommended to choose the suggested folder)



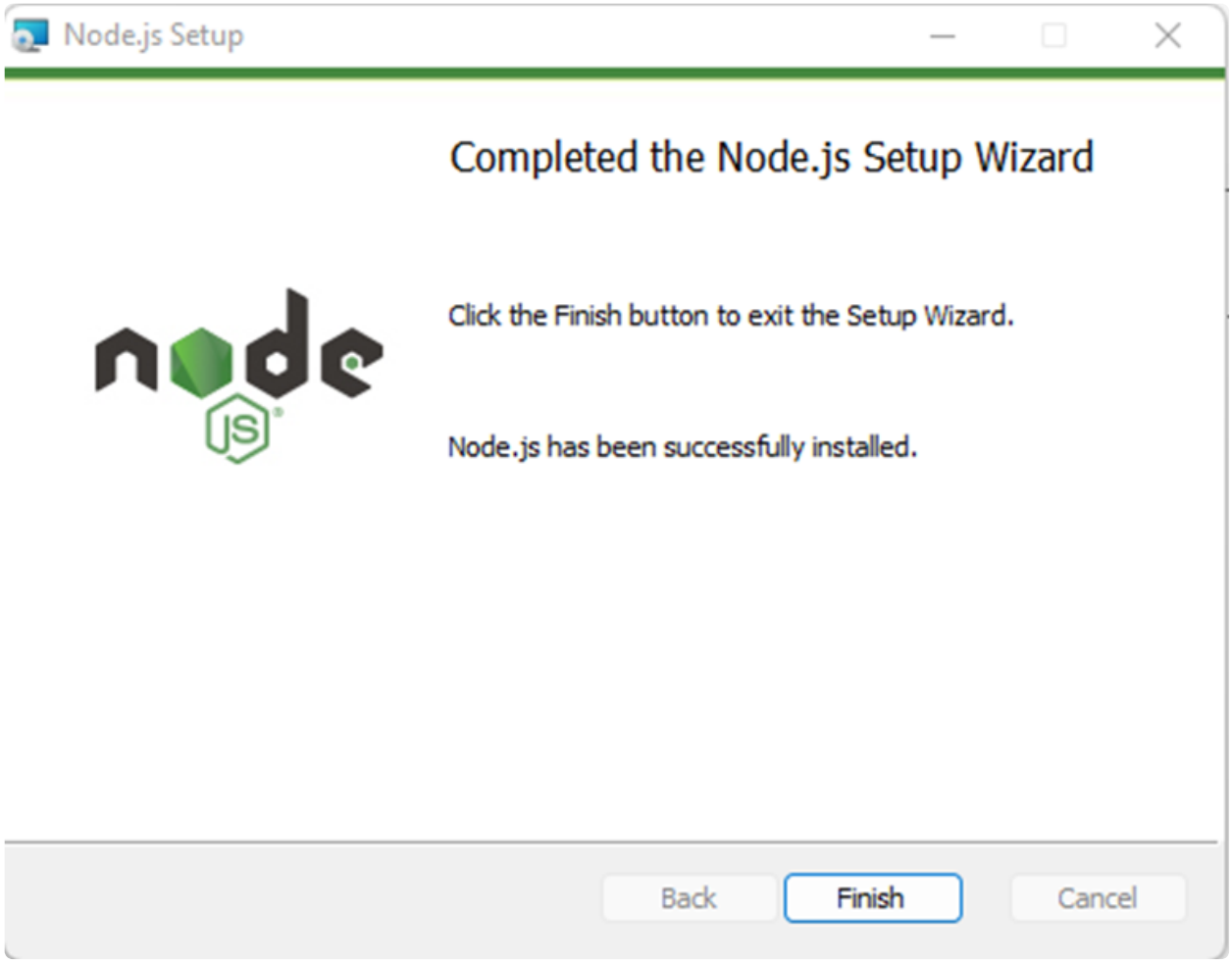
5. At this step, select the option to automatically install the tools.



6.Click “Install” to start the installation.



8. Installation complete. Check that you have received the successful installation message.



CHECK NODE.JS INSTALLATION?

To verify that the Node.js installation was successful, open a command line on your system and run: node –v

Assuming the installation was successful you should see a message similar to the one below.

FIRST NODE.JS PROGRAM – HELLO WORLD!

After installing Node.js on your computer, let's try to display "Hello World" in a browser.

Create a Node.js file named helloworld.js with the following content:

```
var http = require('http');

http.createServer(function (req, res) {
    res.writeHead(200, {'Content-Type': 'text/html'});
    res.end('Hello World!');
}).listen(8080);
```

Code explanation:

1. The "require" function imports a JavaScript file/module, executes that file and returns an object. With this object, you can then use the various functionalities available in the module called by the require function.

2. In the 2nd line of code, we create a simple server application, through a function. This function is called whenever a request is made to our server application.

3. When a request is received, our function returns a response to the client with the text "Hello World". The writeHead function is used to send header data to the client and the end function closes the connection with the client.

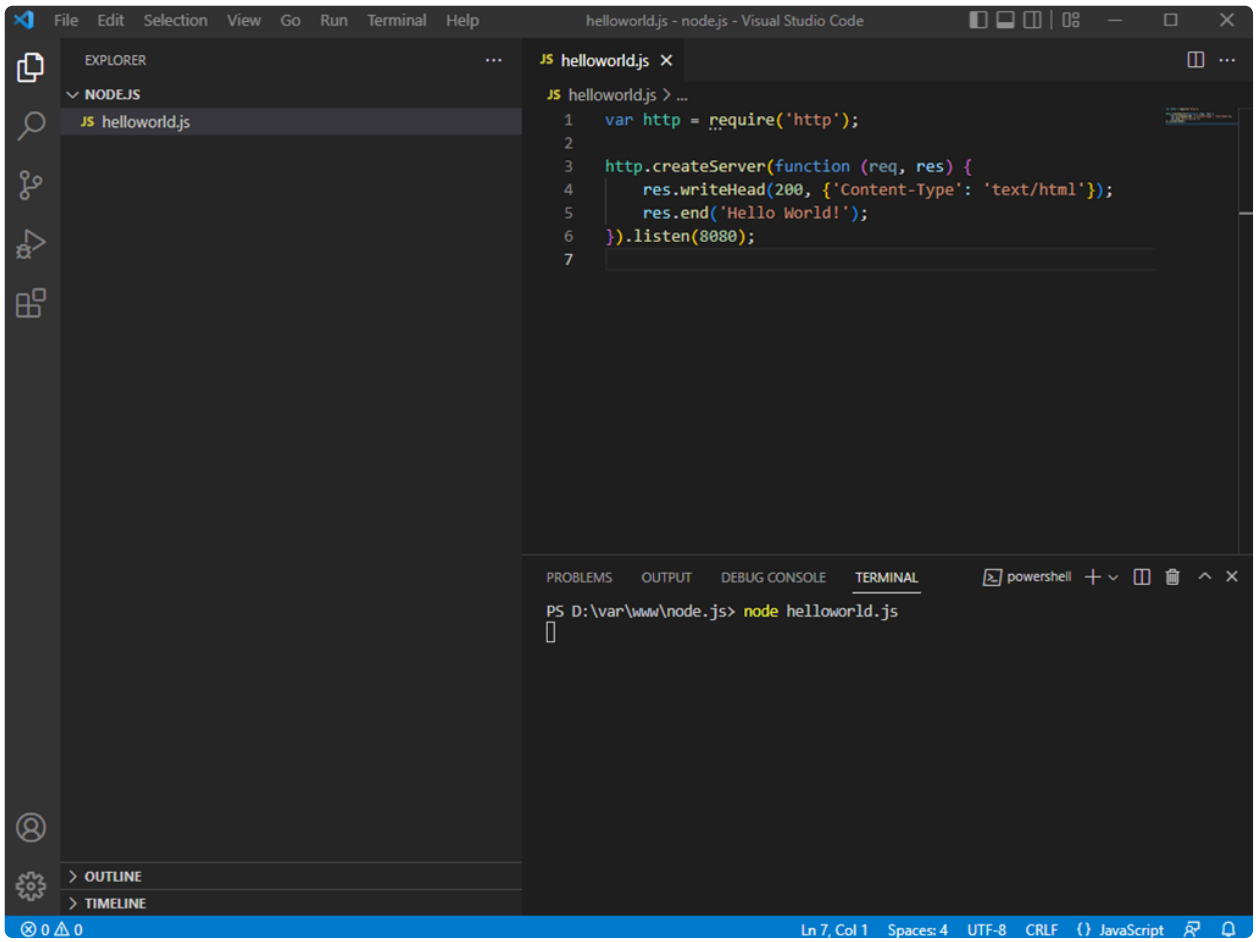
4. We use the .listen function to make our server application listen to client requests on port 8080 (we can specify any free port).

RUN THE CODE

Save the code to your computer, go to the respective folder and run:

node helloworld.js

Alternatively, you can run the program directly in a VSCode terminal.



In your browser, enter the following address:

http://127.0.0.1:8080

SUMMARY

- We explained what Node.js is, its applicability and functionalities. We installed
- Node.js through the msi installation package that is available on the Node.js website.
- This installation installed the modules required to run a Node.js application.

- We created a simple application in Node.js that consisted of creating a server that listens on a specific port.

- When a request arrives at the server, the client automatically sends a 'Hello World' response to the client.