**Installation of Node JS on Windows**

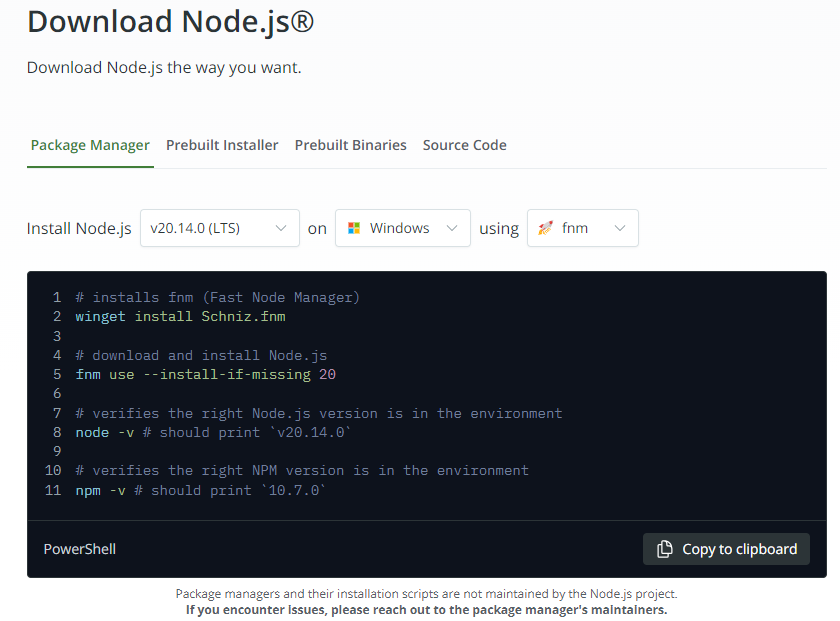
Node.js can be installed in multiple ways on a computer. The approach used by you depends on the existing Node.js development environment in the system. There are different package installers for different environments. You can install Node.js by grabbing a copy of the source code and compiling the application. Another way of installing Node.js is by cloning the Node.js GIT repository in all three environments and then installing it on the system.

**Installing Node On Windows (WINDOWS 10)**

You have to follow the following steps to install the [Node.js](https://www.geeksforgeeks.org/nodejs) on your Windows :

**Step 1: Download the NodeJS**

Downloading the Node.js ‘.msi’ installer the first step to install Node.js on Windows is to download the installer. Visit the official Node.js website i.e) [https://nodejs.org/en/download/](https://nodejs.org/en/download)



*Download Node.js*

**Step 2: Running the Node.js installer**

Now you need to install the node.js installer on your PC. You need to follow the following steps for the Node.js to be installed:

**Double-click on the .msi installer**

The Node.js Setup wizard will open.

**Welcome To Node.js Setup Wizard**

Select “Next”

A screenshot of a computer

Description automatically generated**After clicking “Next”, the End-User License Agreement (EULA) will open.**

Check “I accept the terms in the License Agreement”

Select “Next”

A screenshot of a software license agreement

Description automatically generated

**Destination Folder Set the Destination Folder where you want to install Node.js & Select “Next”.**

A screenshot of a computer

Description automatically generated

**Custom Setup**

Select “Next”

A screenshot of a computer

Description automatically generated

**Ready to Install Node.js.**

The installer may prompt you to “install tools for native modules”.

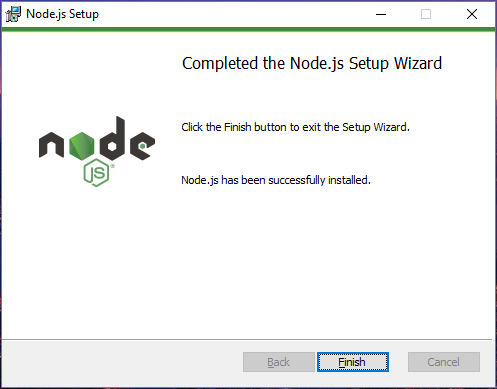
Select “Install”

A screenshot of a computer

Description automatically generated

**Do not close or cancel the installer until the installation is complete. Complete the Node.js Setup Wizard.**

Click “Finish”



**Step 3: Verify that Node.js was properly installed or not**

To check that node.js was completely installed on your system or not, you can run the following command in your command prompt or Windows Powershell and test it:-

C:\Users\Admin> node -v

A screenshot of a computer

Description automatically generated

If node.js was completely installed on your system, the command prompt will print the version of the Node JS installed.

**Step 4: Updating the Local npm version**

You can run the following command, to quickly update the npm

npm install npm --global // Updates the ‘CLI’ client

Step 5 : Running Sample application

An Example Node.js Application

The most common example Hello World of Node.js is a web server:

const { createServer } = require('node:http');

const hostname = '127.0.0.1';

const port = 3000;

const server = createServer((req, res) => {

res.statusCode = 200;

res.setHeader('Content-Type', 'text/plain');

res.end('Hello World');

});

server.listen(port, hostname, () => {

console.log(`Server running at http://${hostname}:${port}/`);

});

To run this snippet, save it as a server.js file and run node server.js in your terminal.

This code first includes the Node.js http module.

Node.js has a fantastic standard library, including first-class support for networking.

The createServer() method of http creates a new HTTP server and returns it.

The server is set to listen on the specified port and host name. When the server is ready, the callback function is called, in this case informing us that the server is running.

Whenever a new request is received, the request event is called, providing two objects: a request (an http.IncomingMessage object) and a response (an http.ServerResponse object).

Those 2 objects are essential to handle the HTTP call.

The first provides the request details. In this simple example, this is not used, but you could access the request headers and request data.

The second is used to return data to the caller.

In this case with:

res.statusCode = 200;

we set the statusCode property to 200, to indicate a successful response.

We set the Content-Type header:

res.setHeader('Content-Type', 'text/plain');

and we close the response, adding the content as an argument to end():

res.end('Hello World\n');

STEP 6: Open browser and type address 127.0.0.1:3000

Output: helloworld

Step 7 : VScode nodejs

Create a folder

Create file First.js with following content

const x=” My first web page”

Console.log(x)

Go to terminal and run

node .\First.js

**Basic Structure of an HTML Document**

An HTML document typically starts with a <!DOCTYPE html> declaration, followed by the <html>, <head>, and <body> tags.

html

Copy code

<!DOCTYPE html>

<html>

<head>

<title>My Webpage</title>

</head>

<body>

<!-- Content goes here -->

</body>

</html>

**Common HTML Tags and Their Usage**

**Headers**

Header tags range from <h1> to <h6>, with <h1> being the largest and most important heading, and <h6> the smallest.

html

Copy code

<h1>Main Heading</h1>

<h2>Subheading</h2>

<h3>Sub-subheading</h3>

**Paragraphs (<p>)**

Paragraphs are used to define blocks of text.

html

Copy code

<p>This is a paragraph of text.</p>

**Comments**

Comments are not displayed in the browser and are used to leave notes within the code.

html

Copy code

<!-- This is a comment -->

**Divisions (<div>)**

The <div> tag is used to group block-level content or elements.

html

Copy code

<div>

<h2>Section Title</h2>

<p>This is a paragraph within a div.</p>

</div>

**Lists**

There are two main types of lists: ordered (<ol>) and unordered (<ul>).

* **Ordered List**:

html

Copy code

<ol>

<li>First item</li>

<li>Second item</li>

<li>Third item</li>

</ol>

* **Unordered List**:

html

Copy code

<ul>

<li>First item</li>

<li>Second item</li>

<li>Third item</li>

</ul>

**Definition List**

Definition lists are used for terms and their definitions.

html

Copy code

<dl>

<dt>HTML</dt>

<dd>Hypertext Markup Language</dd>

<dt>CSS</dt>

<dd>Cascading Style Sheets</dd>

</dl>

**Links (<a>)**

Links are created using the <a> tag.

html

Copy code

<a href="https://www.example.com">Visit Example</a>

To create a mailto link, which opens the user's email client:

html

Copy code

<a href="mailto:someone@example.com">Send Email</a>

To create a call link, which initiates a phone call on mobile devices:

html

Copy code

<a href="tel:+1234567890">Call Us</a>

**Self-Closing Tags**

Some HTML tags are self-closing, meaning they don’t need an explicit closing tag. Examples include <img>, <br>, and <hr>.

html

Copy code

<img src="image.jpg" alt="Description">

<br> <!-- Line break -->

<hr> <!-- Horizontal rule -->

**Input Tags**

Input tags are used within forms to collect user data.

html

Copy code

<form>

<label for="name">Name:</label>

<input type="text" id="name" name="name">

<label for="email">Email:</label>

<input type="email" id="email" name="email">

<input type="submit" value="Submit">

</form>

**Additional Elements**

**Download Links**

To create a link that allows users to download a file, use the download attribute in the <a> tag.

html

Copy code

<a href="file.zip" download>Download File</a>

 </html>