Tutorial 0.1

Introduction to Node.js

>>> Node@1 :: what you will learn?

1. What is Node.js and introduction to how to use it
2. What is and how to use Node.js REPL (READ-EVAL-PRINT-LOOP)
3. What is NPM (Node package manager)
4. What is package.json file
5. How we can use NPM to create package.json file
6. Create your first Node app and run it using CLI (command-line-interface)

>>> Node@1 :: what is Node.js?

Node.js (like Deno)is an open-source, cross-platform JavaScript run-time environment that executes JavaScript code outside of a browser. Node.js represents a "JavaScript everywhere" paradigm, unifying web application development around a single programming language, rather than different languages for server side and client side scripts.

>>> Node@2 :: what is Node.js?

- Node.js is an open source server environment
- Node.js is free
- Node.js runs on various platforms (Windows, Linux, Unix, Mac OS X, etc.)
- Node.js uses JavaScript on the server

>>> Node@3 :: where to use Node.js?

- I/O bound Applications
- Data Streaming Applications
- Data Intensive Real-time Applications (DIRT)
- JSON APIs based Applications
- Single Page Applications

>>> Node@4 :: Browser vs Node

In the browser, most of the time what you are doing is interacting with the DOM, or other Web Platform APIs like Cookies. Those do not exist in Node, of course. You don’t have the `document`, `window` and all the other objects that are provided by the browser.

And in the browser, we don’t have all the nice APIs that Node.js provides through its modules, like the filesystem access functionality.

Another big difference is that in Node.js you control the environment. Unless you are building an open source application that anyone can deploy anywhere, you know which version of Node you will run the application on. Compared to the browser environment, where you don’t get the luxury to choose what browser your visitors will use, this is very convenient.

>>> Node@5 :: Node.js environment setup

We need to install text editor and Node runtime to start using/learning it.

(a) Text Editor (Sublime, VSCode, etc.)

(b) The Node.js binary installables.

*We must have both in PClabs (if no use this)

>>> Node@6 :: Node.js usage

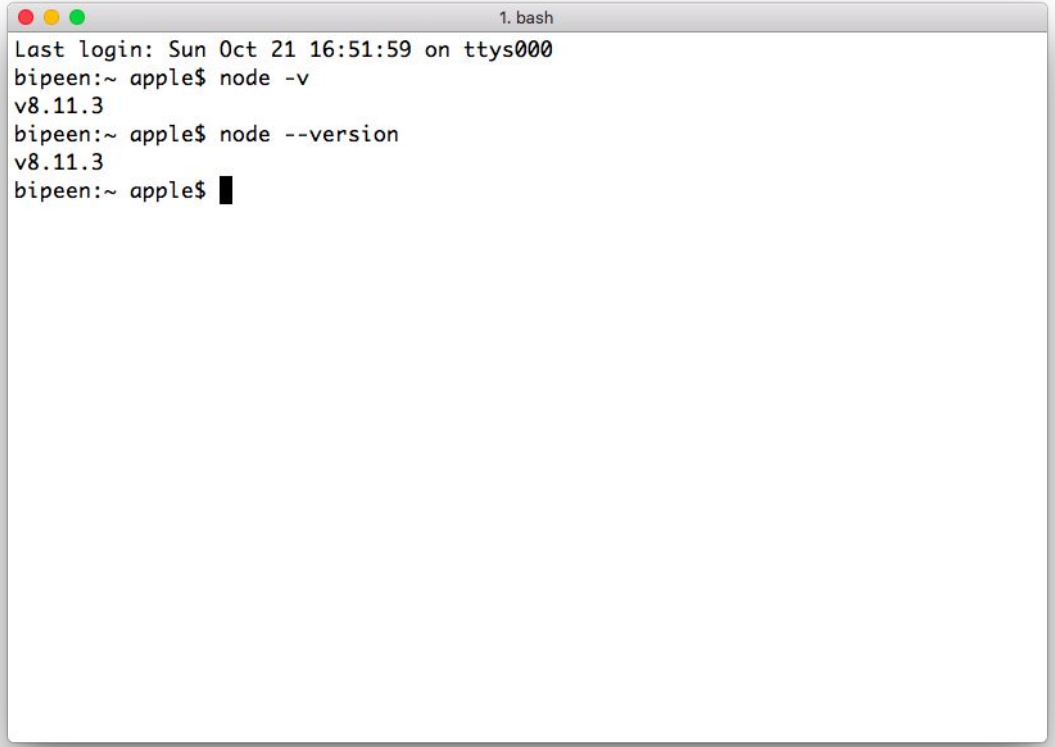
Node does not have GUI (graphical user interface) and you have to interact with it using CLI (command line interface)

In windows - cmd.exe

In mac - terminal

>>> Node@7 :: Node.js usage - 2

If you have installed Node.js correctly, you can start cmd.exe(windows) or terminal (linux, mac) and type [node -v] or [node --version] and you will get corresponding response.

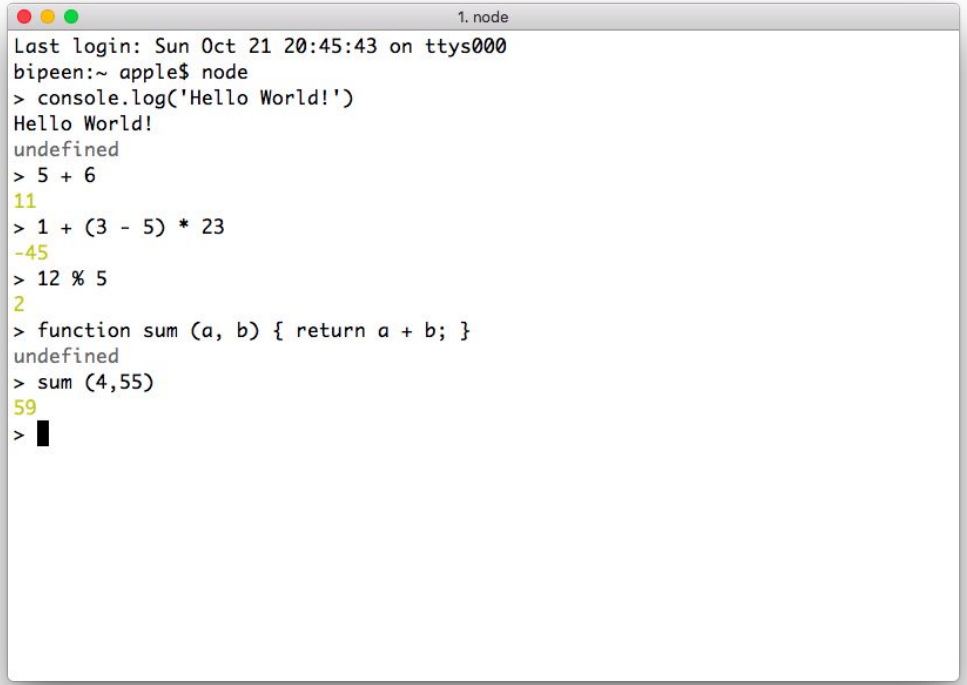


>>> NODE REPL

>>> Node@8 :: Node.js REPL

If you have Node.js installed you can use Node.js REPL (READ-EVAL-PRINT-LOOP). REPL is a interactive tool where you can run JavaScript code and get immediate result.

To start Node REPL open cmd.exe (windows) or terminal (mac, linux) and type [node] . Then enter your JS expressions and press ENTER. Try console.log(‘Hello World!’) and simple math operations.



>>> Node@9 :: Node.js REPL

REPL is a great way to explore Node features in a quick way.

Try entering the name of a JavaScript object, like `Number`, add a dot and press `TAB 2 times`.

The REPL will print all the properties and methods you can access on that object

1. node

```
Last login: Sun Oct 21 21:10:41 on ttys001
bipeen:~ apple$ node
> Number.
Number.__defineGetter__       Number.__defineSetter__
Number.__lookupGetter__       Number.__lookupSetter__
Number.__proto__              Number.constructor
Number.hasOwnProperty         Number.isPrototypeOf
Number.propertyIsEnumerable   Number.toLocaleString
Number.toString               Number.valueOf

Number.apply                  Number.arguments
Number.bind                   Number.call
Number.caller                 Number.length
Number.name

Number.EPSILON                Number.MAX_SAFE_INTEGER
Number.MAX_VALUE              Number.MIN_SAFE_INTEGER
Number.MIN_VALUE              Number.NEGATIVE_INFINITY
Number.NaN                    Number.POSITIVE_INFINITY
Number.isFinite               Number.isInteger
Number.isNaN                  Number.isSafeInteger
Number.parseFloat             Number.parseInt
Number.prototype

> Number.
```

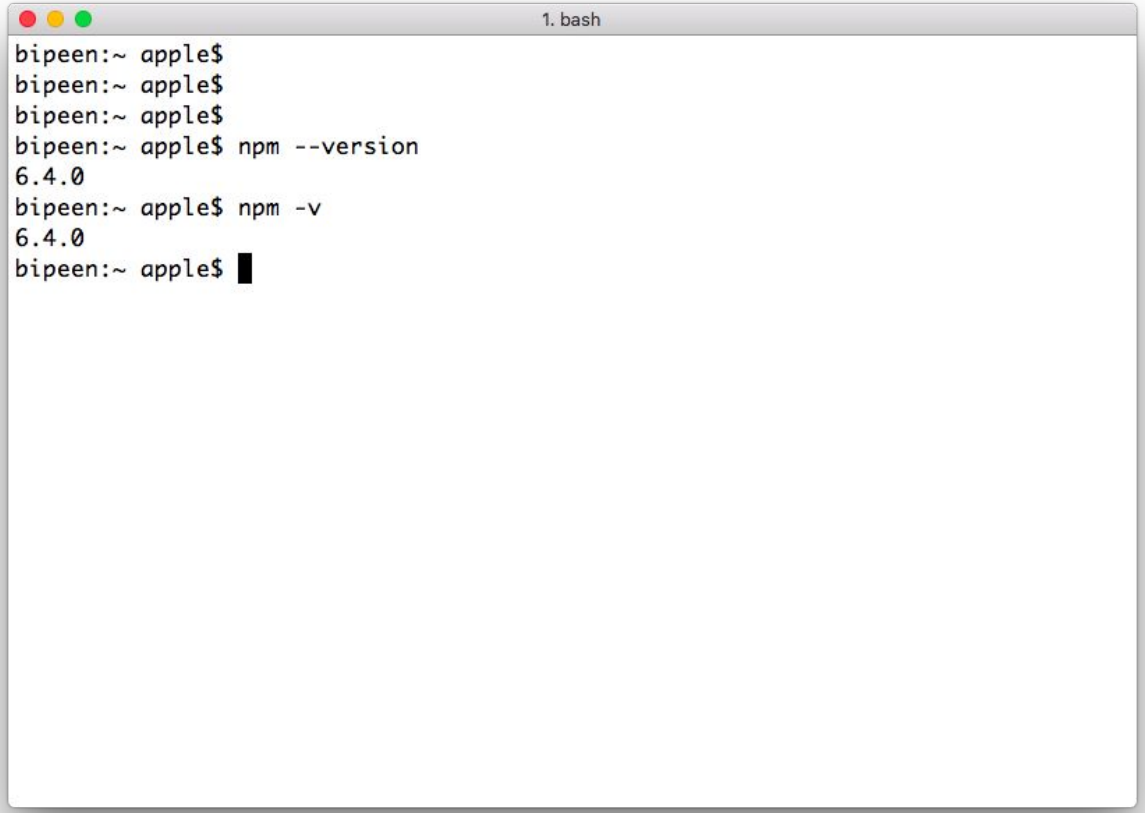
>>> NODE NPM

>>> Node@10 :: NPM - node package manager

NPM is the standard package manager for Node.js. It comes with Node.js.

After you installed Node you can check **NPM** version in cmd.exe or terminal.

Just type [npm --version] or [npm -v]



>>> Node@12 :: NPM - node package manager

This can be a bit confusing for newcomers to Node.js. You may think - “why I need package manager?” - when you start building your application using node.

Is it possible to build app without NPM? Yes, it is possible. If you don’t want to include additional packages of other developers don’t use NPM.

One of the main task of NPM is to manage external packages added to your project and their dependencies. About dependencies we will talk in the coming tutorials.

You will use [npm install <package_name> --save] command to install external packages to your packages (libraries).

```
>>> NODE package.json
```

>>> Node@13 :: package.json

The package.json file is kind of a manifest for your project. It can do a lot of things, completely unrelated. It’s a central repository of configuration for tools, for example. It’s also where NPM and store the names and versions of the package it installed.

In simple terms, this file explains your Node.js application. It tells what tools are used to build your app and holds the information about that tools.

You can create this file on your own or use NPM to instantiate.

>>> Node@14 :: NPM to create package.json

We create package.json file to describe our NODE.js application.

To create package.json we use NPM command init.

First, we create application folder anywhere we want go to that folder using terminal or cmd.exe and type [npm init]

Then, you have to answer several questions about your future project.

1. Shell

```
Last login: Sun Oct 21 21:43:19 on ttys000
bipeen:~ apple$ cd desktop
bipeen:desktop apple$ mkdir nodeapp
bipeen:desktop apple$ cd nodeapp
bipeen:nodeapp apple$ npm init
This utility will walk you through creating a package.json file.
It only covers the most common items, and tries to guess sensible defaults.

See `npm help json` for definitive documentation on these fields
and exactly what they do.

Use `npm install <pkg>` afterwards to install a package and
save it as a dependency in the package.json file.

Press ^C at any time to quit.
package name: (nodeapp)
```

>>> Node@15 :: NPM to create package.json

Look at the questions and type the answer and press enter and you will get next question until the NPM will ask you to confirm all entered data.

After you confirm NPM will create .json file. Again, it is simple .json file with a certain structure. You can create it without NPM, but with NPM it is easier.

If you don’t type anything NPM will leave default values for each question. Check the pic. and answers. In the round brackets NPM offered default values.

If some points are confusing, don’t worry we will go through each of them during the tutorials related to Node.js

1. node

```
Press ^C at any time to quit.
package name: (nodeapp)
version: (1.0.0)
description: My first node application
entry point: (index.js) app.js
test command:
git repository:
keywords: nodeapp
author: Bunyod Kh.
license: (ISC) MIT
About to write to /Users/apple/Desktop/nodeapp/package.json:

{
  "name": "nodeapp",
  "version": "1.0.0",
  "description": "My first node application",
  "main": "app.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "keywords": [
    "nodeapp"
  ],
  "author": "Bunyod Kh.",
  "license": "MIT"
}


Is this OK? (yes)
```

>>> Node@16 :: NPM to create package.json

You can leave these (they don't have default answers) empty:

-test command

-git repository

-keywords

-author

Just press enter and skip these questions.

1. node

```
Press ^C at any time to quit.
package name: (nodeapp)
version: (1.0.0)
description: My first node application
entry point: (index.js) app.js
test command:
git repository:
keywords: nodeapp
author: Bunyod Kh.
license: (ISC) MIT
About to write to /Users/apple/Desktop/nodeapp/package.json:

{
  "name": "nodeapp",
  "version": "1.0.0",
  "description": "My first node application",
  "main": "app.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "keywords": [
    "nodeapp"
  ],
  "author": "Bunyod Kh.",
  "license": "MIT"
}


Is this OK? (yes)
```

>>> Node@16 :: NPM to create package.json

Entry point is the application's starting point. you will run this .js file to start your app.

If you don't give a name to your main .js file NPM will set a default value "index.js"

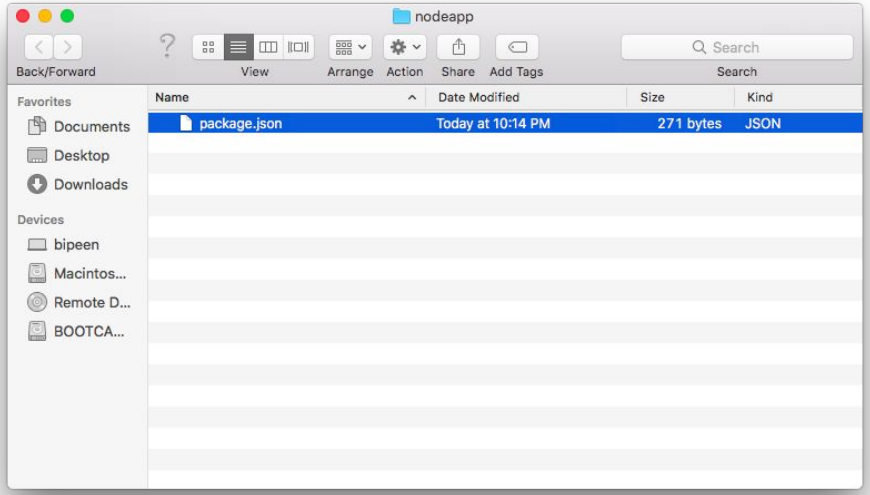
Usually, you call your main .js file "app.js". But it is up to you at the end of the day.

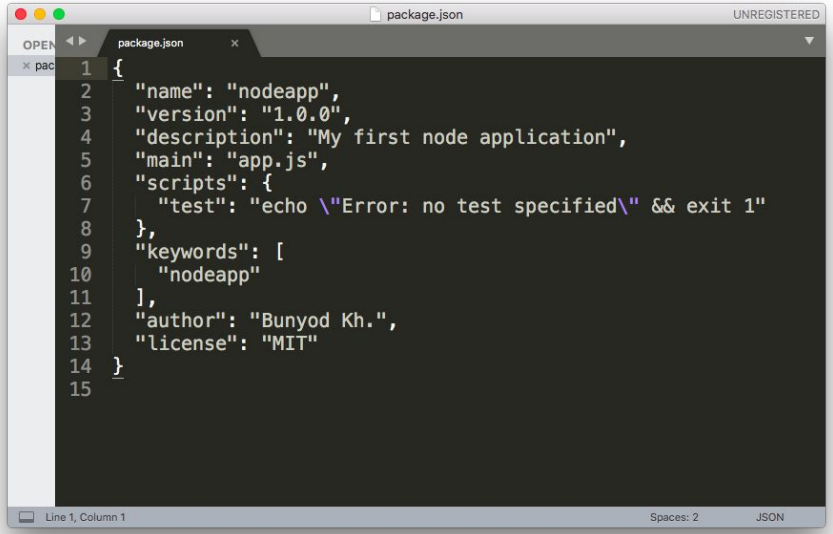
1. node

```
Press ^C at any time to quit.
package name: (nodeapp)
version: (1.0.0)
description: My first node application
entry point: (index.js) app.js
test command:
git repository:
keywords: nodeapp
author: Bunyod Kh.
license: (ISC) MIT
About to write to /Users/apple/Desktop/nodeapp/package.json:

{
  "name": "nodeapp",
  "version": "1.0.0",
  "description": "My first node application",
  "main": "app.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "keywords": [
    "nodeapp"
  ],
  "author": "Bunyod Kh.",
  "license": "MIT"
}


Is this OK? (yes)
```

>>> Node@17 :: NPM to create package.json

If you confirm at the end, package.json file be created in your application folder. You can open it and check the structure of the created package.json file.





>>> Node@18 :: package.json

It is very important to name the .json file package.json

>>> NODE practice

>>> Node@19 :: Practice - 1

1. Open CLI -> cmd.exe (windows) or terminal (mac)
2. Type [node --version] to check whether node is installed
3. Create a folder with any name you like [i.e. mynodeapp] on the desktop
4. Create a file [app.js] inside this folder
5. Then navigate to that folder using CLI
    a. You can open CLI and type ([cd] + single space) and then drag and drop your folder to the CLI window. Then press enter and it will take your folder.
6. Write [console.log(‘Hello World!’)] inside your app.js file and run it using [node app.js] command in your CLI.

>>> Node@20 :: Practice - 2 : create package.json file

1. Go to your folder using CLI (cmd or terminal)
2. type npm init and fill all required fields
3. Check your folder for the package.json file
4. Change that package.json. i.e.:
    a. change the author name. put your own name
    b. change license from ‘ISC’ to ‘MIT’
    c. change the value of ‘main’ from [index.js] to [app.js]

5. Add external package using [npm install <package_name> --save] command and then [require()] installed package and use the functionality of that external package in your app. i.e. ‘one-liner-joke’ -> npm install one-liner-joke --save

```
>>> see you
```