**Tutorial for practice**

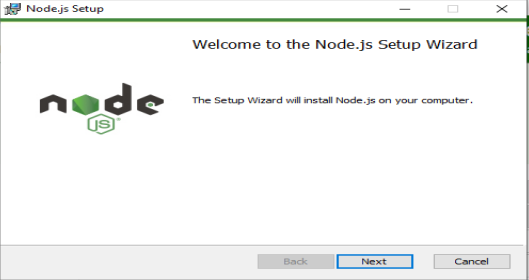
1. **Install NodeJS**

* Download the installer from the link: <https://nodejs.org/en/download/current>
* The newest version of **NodeJS** is **20.6.0**.
* Choose the installer that meets your OS requirements and install.

A screenshot of a computer

Description automatically generated

* After open the installer this window will appear

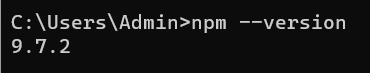


* Follow the installation step to finish the installation.
* After finished installation you can check if **NodeJS** has been installed in your device by using the command “node --version” in your OS terminal, the result showed after the command is the version of **NodeJS** installed in your device, which also means that you have successfully installed **NodeJS**.

A screen shot of a computer

Description automatically generated

* In addition, when the NodeJS has been successfully installed, **NPM** (Node Package Manager) will also be installed, you can check by typing your terminal “**npm --version**”, the result will show the version of **NPM**



1. **Start project NodeJS:**

* Step 1: Create a new folder for your project. Inside this folder, create another new folder and this folder will be used to contain our **NodeJS** project.
* Step 2: Open your OS terminal at the project directory (you can open terminal and then pointed to the project directory in your device)
* Step 3: Type “**npm init**” to initialize a new project:
  + The terminal will ask you to fill in the package name, press enter without typing anything to use the default value which is the name of the current project folder.

A screenshot of a computer program

Description automatically generated

* + After that the terminal will continue to ask to choose the version, press enter without typing anything to choose the default version.

A screenshot of a computer program

Description automatically generated

* + At the next line the terminal will ask you to write some description for the project, you can leave it empty, and press enter to move to the next step.

A screen shot of a computer

Description automatically generated

* + The next line will ask you to choose the entry point, which is the main file to run NodeJS code, by default the entry point is index.js, you can enter without typing anything to choose **index.js** as the default entry point.

A screenshot of a computer program

Description automatically generated

* + The next 5 lines the terminal will ask you to enter test command, git repository, keywords, author and license respectively, you can leave them as default by pressing enter without typing anything.

A screen shot of a computer

Description automatically generated

* + And finally, the terminal will summary your project information which you typed above, you can type “**yes**” to finish the initial step of the **NodeJS** project.

A screenshot of a computer program

Description automatically generated

* Step 4: Open Visual Code at the directory of your **NodeJS** project you will see **package.json** file where the summary information of your project is stored.

A screen shot of a computer program

Description automatically generated

* Step 5: Create **index.js** file to begin coding.
* Step 6: Create your first **NodeJS** server, the below block of code will create a **NodeJS** server which return a “Hello World!” string whenever you access your computer on port 8080.

A screen shot of a computer code

Description automatically generated

A screenshot of a computer

Description automatically generated

* + The “**http**” module is included in the **index.js** file for using by utilizing “**require()**” function.
  + The method “**createServer()**” of http module will help you to create a server which is listen on port 8080 defined at the code line “**listen(8080)**”.
  + After finishing entering the above code block into your IDE, please type in the terminal in your current **NodeJS** project folder the command “**node index.js**”, now you can access “**localhost:8080**” or “**127.0.0.1:8080**” to see the result.

1. **NodeJS With MySQL**

* In this section you will create a simple web application for filling student’s information with **NodeJS** implementing **ExpressJS** framework as backend and **MySQL** as database.
* First of all, you need to install **ExpressJS** in the terminal pointed to your project folder using:

A green and red text

Description automatically generated

* In addition, to make our development process a lot easier, we will install a tool from **npm** which is “**nodemon**”. This tool restarts our server as soon as we make a change in any of our files, otherwise we need to restart the server manually after each file modification. To install ” **nodemon**”, use the command.

A green and red text

Description automatically generated

* Thirdly, we will install “**cors”** which is a **NodeJS** for providing a Connect/Express middleware that can be used to enable **Cross-origin resource sharing** (**CORS**) with various options.

A group of colorful circles

Description automatically generated

* Last but not least, you must install “**mysql2**” module so that we can use functions from “**mysql2**” library to interact with our database.

A screen shot of a computer screen

Description automatically generated

* Before moving to coding section with **NodeJS**, you must create your local **MySQL** server, download and install **MySQL** server at the link: <https://dev.mysql.com/downloads/mysql/>
  + After installing **MySQL** server, run **MySQL** configuration and set up your **MySQL** server, you can let everything as its default to install. Set up password for your root account of **MySQL** (this account has already been created with username ‘**root’**; you just need to set up password for it).

A screenshot of a computer

Description automatically generated

* + After setting up the password, the next steps just leave them as default and press “Next” button and execute the configuration process.
  + To make it easier to manage **MySQL** database and server you can download and install **MySQL** Workbench at this link: <https://dev.mysql.com/downloads/workbench/>
  + After finishing setting up **MySQL** Workbench you can see your **MySQL** server on the workbench. The information of **MySQL** can be extracted from **MySQL** Workbench, and it is necessary for **NodeJS** to interact with **MySQL**.
* Start your **NodeJS** project with **ExpressJS** framework by using this code block, with this code block you can create a **NodeJS** server with **ExpressJS** framework and return “Hello World!” when accessing your local host on port **4000**. Run the server by typing on terminal “**nodemon index.js**”

A computer code with colorful text

Description automatically generatedA screenshot of a computer

Description automatically generated

* Now let’s build our database using **NodeJS**
  + Create a new JavaScript file for building Database, named it **mydb.js**, at first, we need to import **mysql2/promise** module into our mysql.js file and create an object to store **MySQL** server information.

A black background with blue and green text

Description automatically generated

A screen shot of a computer code

Description automatically generated

* + The following code block will help you to connect to **MySQL** server and create a new database. Pay attention that we will use synchronous to prevent **NodeJS** from closing the database connection before it finishes its task with the database.

A screen shot of a computer code

Description automatically generated

* + The next thing to do after creating the database is to create table, for this web application we only one table for storing students’ information, the following code block is to create a table names “**students“** in our database.

A computer screen shot of a code

Description automatically generated

* + Now we will create some function to interact with our database, as the website allow students to enter their information so it must have a function to insert to **“students”** table and get information form table **“students”**. The following code block will be used for this job.

A computer screen shot of colorful text

Description automatically generated

* + The next code block is for getting student’s information based on his/her student id.

A screen shot of a computer code

Description automatically generated

* + Now we have enough functions for our website, but these functions can only be used in **mydb.js** file, to use them in index.js file as well as other files we must export them as a module, the following code will help you to do this job.

A screen shot of a computer code

Description automatically generated

* + And this is the final version of **mydb.js** file.

A screenshot of a computer code

Description automatically generated

* + To use the exported functions from **mydb.js** file in others files we use this code block at the beginning of each file.

A computer code with red and blue text

Description automatically generated

In this code block the “**require()**” function pointed to file mydb.js and extract functions that we exported above and map them to functions on the left-hand side of the code block respectively.

* After finishing the functions for building our database using **NodeJS**, we will now use these functions in **index.js** file. The purpose is to initialize our database if it does not exist when accessing our localhost at first. The example code blocks below are for creating a database and table if they do not exist in the database when user access to URL **“/create-database**” and “/**create-table**”.

A screen shot of a computer code

Description automatically generated

* After adding the above code blocks into your index.js file you can now start your **ExpressJS** server by typing “**nodemon index.js**” and access the link “**127.0.0.1:4000/create-database**” and “**127.0.0.1:4000/create-table**” respectively, your database and table have now been created and you can check them at **MySQL** Workbench.
* Now we will create a URL to insert a new student into our database. After adding the code block below, you can access “**127.0.0.1:4000/insert-student**” to insert three new students into the database.

A screenshot of a computer code

Description automatically generated

* Finally, we will get information of a student based on his/her student ID, the below code allow you to search and get the information of a student based on his/her student id when access the URL “**127.0.0.1:4000/student/{student ID here}**”.

A screen shot of a computer code

Description automatically generated

* We have finished building a simple backend NodeJS local server using **ExpressJS** framework, now we will create a **ReactJS** web UI to interact with this server.
  + To create a new ReactJS project, first create terminal direct to the parent folder of your **NodeJS** project folder, then type “**npx create-react-app your-project-name**” a new **ReactJS** project will be created at the current folder, after the project has been successfully initialized, please point the terminal to this new project folder. In addition, you will need to install **axios** which is a HTTP Client library based on Promise, we will use **axios** for making http request from our **ReactJS** client to **NodeJS** server, you can install **axios** by typing in the terminal of your **ReactJS** project this command.

A screen shot of a computer

Description automatically generated

* + Now most of everything we need for the project is set up, you can now start the project by typing “**npm start**” the web will be available on “**localhost:3000**”.
  + Our project is simple so we will write code directly to the **App.js** file without creating any other components. You can try to create a component and use it after finishing this tutorial.
  + Delete all content in div tag with className “**App**”.
  + At first, we need to import “**useState**” and “**axios**” to use in our code, the below code block is the importation code block of “**usestate**” and “**axios**”

A screen shot of a computer code

Description automatically generated

* + We are going to create a page for students to enter their personal information. The information they need to enter includes their student ID, full name, email address, gender, and department. We must create **states**, which are built-in **React** objects used to contain data or information about the component, to contain data. The states usually go along with their **setState** methods, which enqueue changes to the component state and tells React that this component and its children need to be re-rendered with the updated state. The code for declare state and **setState** methods for each students’ information as well as for others needed usage are declared in function “**App()**” as below.

A screen shot of a computer program

Description automatically generated

* + Now we will create a simple form for students to enter their personal information, the below code will help you to do this job, please put this code block inside div tag with className “**App**”

A screen shot of a computer program

Description automatically generated

The result when you run the code is as below:

A screen shot of a computer screen

Description automatically generated

* + We have successfully created a simple form for students to enter their personal information, but the form at this point is just a UI not less, we must create function to handle the submission of this form as well as send the data to our created **NodeJS** server, this function will take data from the form and send to **NodeJS** server then it will get the inserted information of student from **NodeJS** server, this process is performed with the support of **axios**, the below code block is a function call **handeSubmit**, and this function will help you the we just mentioned.

A screen shot of a computer program

Description automatically generated

* + Now to make this function in use whenever the form is submitted, we will add an **onClick** attribute to the submit button of the form which utilizes the **handleSubmit** function, the submit button of the form is now changed to as below.

A screen shot of a computer

Description automatically generated

* + As we will submit the student’s information to **NodeJS** server through the route **“/student-form”**, so we will need to create a route in **NodeJS** server to receive the submitted student’s information from **ReactJS**, the below **NodeJS** code block will help you to the job.

A computer screen shot of text

Description automatically generated

* + With the above code blocks we can submit the entered information to **NodeJS** and stored it below the **MySQL**, now we will create a table to display the information after it has been successfully stored in the database, please insert the below code block inside the div tag with **className** “**App**” and below the form tag.

A computer code with many colorful text

Description automatically generated with medium confidence

* + To sum up, this is the whole file **App.js** code, now you can try to run the whole website by opening a terminal pointed to your **NodeJS** project folder to start your **NodeJS** server and opening a new terminal pointed to your ReactJS project folder and start it:

A screen shot of a computer screen

Description automatically generated