**NodeJS and Rest API -** [Ravindu Wijerathne](https://medium.com/@ravinduwijerathne629?source=post_page-----5221e542c9bf--------------------------------) May 14, 2022

**NodeJS**

NodeJS allows us to take JavaScript out of the browser and allow it to interact directly with the hardware of a computer. We can use JavaScript inside our websites in order to give it functionality and behavior. For example, implementing animations, dropdown menus like vice. But NodeJS allows us to use JavaScript to interact directly with the hardware of the computer. So we can use it for example to create desktop applications. For example, Atom is a desktop application built with HTML, JavaScript, CSS, and Node.js integration. It runs on Electron, a framework for building cross platform apps using web technologies. NodeJS you can write JavaScript code not just in the browser to affect the behavior of your website but also to write full applications that work on your computer.

We can run NodeJS on our own computer. But we could also use NodeJS to run JavaScript code on somebody else’s computer or rather a server. For example, your browser makes a request to Google servers, and on those servers execute JavaScript code to process that request. And after that is done Google server sends back the results to the client and simply displays the results in our web browser. All the heavy lifting code executions happen behind the scenes on the server side rather than on the client side.

**Native Node Modules**

Interact with the local file system using node we can use File Systems built-in module. In order to use the module first, we have to require it.

Text

Description automatically generated

**NPM (Node Package Manager) / External Node Modules**

NPM is a package manager for external modules. So modules that anybody could build that you could incorporate into your projects.

First, we have to initialize npm in our project using **npm init** command and you can give relevant information about your project. After that automatically create a file called package.json in your project folder. Now you can install npm modules.

Text

Description automatically generated

Install npm module → **npm install <module name>**

All the npm modules that you installed you can see inside package.json file under dependencies.

Text

Description automatically generated

**Node Frameworks**

Node has powerful frameworks like Express, Koa and these frameworks are bunch of codes that somebody else wrote that add extra features and help to organize and structure your code. Specially for web apps built with Node. Basically, build these frameworks for write less repetitive code when you are building web applications and it is made especially for web developers.

Install frameworks using **npm install express** and **npm install koa** commands.

**Nodemon Npm Package**

NPM package that monitor changes in our source code and it will automatically restart your server when any changes happen in the code.

**npm install nodemon**

**Node server using express frameworks**

Text

Description automatically generated

**REST API**

REST is essentially just an architectural style for designing APIs. It proposes a set of rules that web developers could follow when they are building their APIs. All websites across the web would use the same structure for building their APIs. If every web API was built using the same common guiding principles then it would be so easy for everybody to work together and be able to use different APIs quickly, easily, and efficiently.

You will also have a whole bunch of rules for making API RESTful. But the two most important once are

(1) Use HTTP request verbs

(2) Use specific pattern of routes/endpoint URLs

These two things are the most important parts you have to consider when you are making your APIs.

**(1)** **HTTP verbs**

Five HTTP verbs are commonly that you have to use in order to make your API RESTful.

Looking HTTP request verbs alongside with our database crud operations

GET → Read

POST → Create

PUT/ PATCH → Update

DELETE → Delete

PUT → Updating database by sending an entire entry to replace the previous one

PATCH → When you are sending a patch request to the server you are only sending the piece of data that needs to be updated. Instead of replacing the entire entry only update the things that need to be updated.

**(2) Use specific pattern of routes/endpoint URLs**

In our server, we can specify specific routes or URLs in order to access certain resources.

For example, if our API was Wikipedia API and in our database, we have a whole bunch of articles. Now the route for **/articles** applies to all of the articles. Now we created a route for articles. When a client makes a **get** request to **/articles** it should fetch all of the articles. And when a client makes a **post** request to the **/articles** route then it should create a single new article and add it to our database of articles.

When we make a **delete** request to **/articles** then it would delete all the articles in our database. But RESTful routing also has rules for individual resources. If the client is targeting **/articles/article\_name** then if the client made a get request that would fetch the specific article from the database.

**Requests targeting all articles**

Text

Description automatically generated

**Requests targeting a specific article**

Text

Description automatically generated