



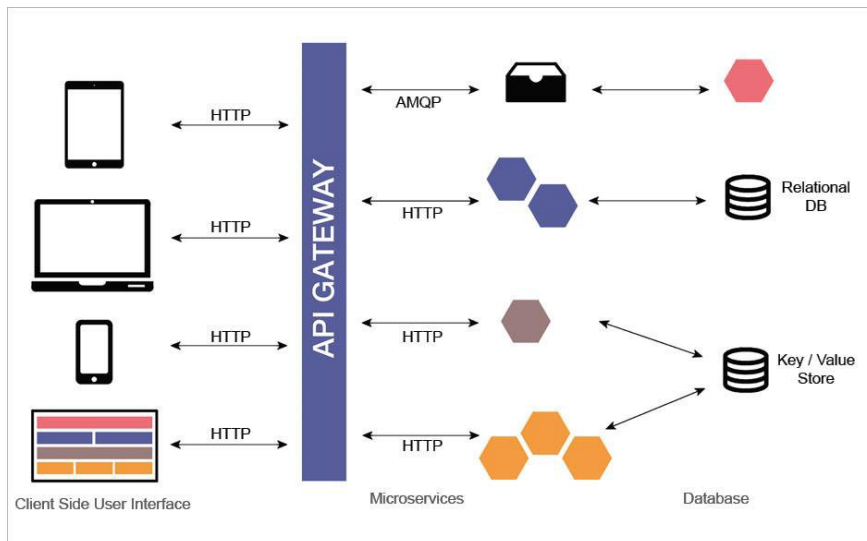
Jobayer Ahmed Mickey

Follow

Front-end Developer | UI / UX Designer | Amateur Blogger | Speed Cubist

Jul 16 · 7 min read

Let's build a timestamp microservice using nodejs, expressjs and body-parser



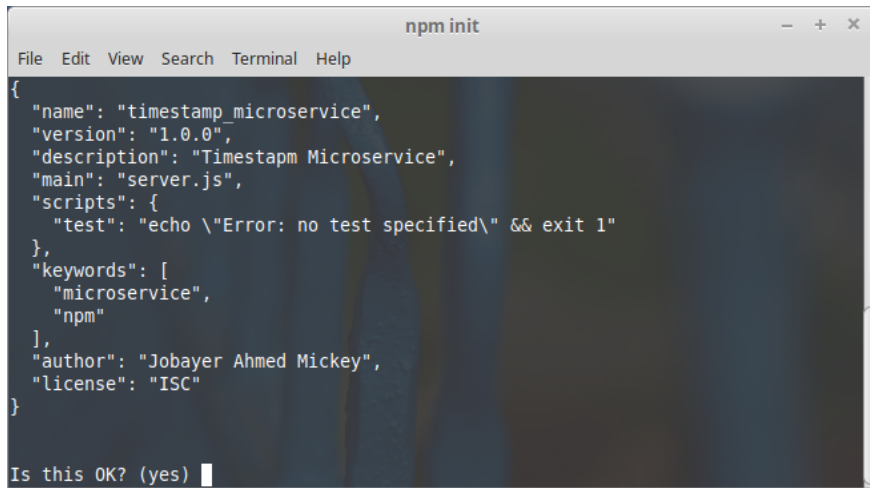
It is basically the first project of the Apeis and Microservices Projects of Freakodkamp. I've completed the Recently Apis and Microservices. So I thought that I write a series of all the projects. This is the first episode of this series. Try to finish all the episodes very quickly. So let's start

সবার প্রথমে আমাদের প্রোজেক্ট ফাইল বানাতে হবে। এখানে বলে রাখি আমি যেহেতু লিনাক্স ইউজার তাই আমি পুরো কাজটাই লিনাক্সে করছি। তবে আমার মনে হয় জানালা এবং আপেলও একই রকম ভাবে করতে হয়। (জানালা অনেকদিন ব্যবহার করি না আর আপেল এ এইসব কাজ করার কোনো অভিজ্ঞতা নেই আমার।) তাই আমরা এখন ডেক্সটপে একটা নতুন ফাইল বানাবো। ফাইলের নাম দিলাম `timestamp_microservice`. নিচের কমান্ড টি টার্মিনালে কপিপেস্ট করুন।

```
mkdir timestamp_microservice
```

আমাদের প্রোজেক্ট ডিরেক্টরী বানানো শেষ। এবার আমাদের কাজ হলো `npm` ইনিশিয়ালাইজ করা। অর্থাৎ আমাদের প্রোজেক্ট ফাইল এ একটা `package.json` ফাইল বানাতে হবে। ম্যানুয়ালি না করে আমরা `npm` এর সাহায্য নেবো। নিচের কমান্ড টি টার্মিনাল এ কপিপেস্ট করুন এবং নিচের মত করে ইনপুট দিয়ে দিন।

```
npm init
```



```

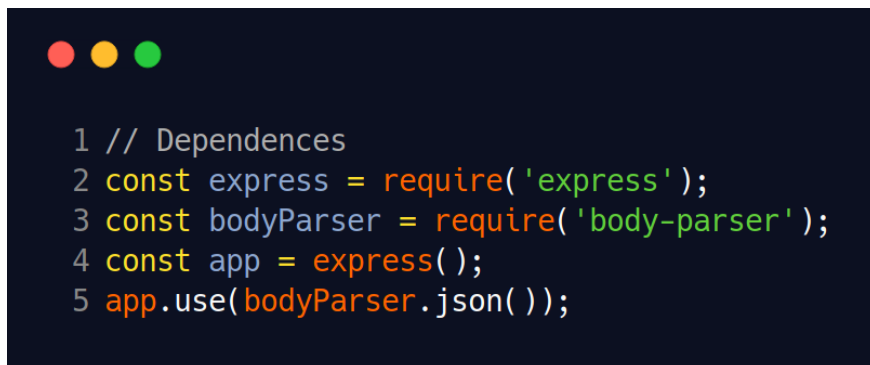
npm init
File Edit View Search Terminal Help
{
  "name": "timestamp_microservice",
  "version": "1.0.0",
  "description": "Timestamp Microservice",
  "main": "server.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "keywords": [
    "microservice",
    "npm"
  ],
  "author": "Jobayer Ahmed Mickey",
  "license": "ISC"
}
Is this OK? (yes)

```

আপনারা যেনো অথবা আমার নাম দিয়ে দিচ্ছেন না ☺ ওইটাতে আপনার নাম ই দিয়েন।

এবার যেহেতু আমরা মেইন হিসেবে server.js ডিক্লয়ার করেছি তাই আমাদের একটা server.js ফাইল বানাতে হবে। টার্মিনালে touch server.js দিলেই প্রোজেক্ট ডিরেক্টরিতে server.js নামের একটা ফাইল তৈরি হবে। আমি এই কাজের জন্য sublime text ইউজ করবো। তো এবার subl server.js টার্মিনালে লিখে এন্টার দিলেই server ফাইলটি সাবলাইম টেক্সট এ ওপেন হবে। এবার কাজের কাজ শুরু করা যাক।

সবার প্রথমে আমরা আমাদের ডিপেন্ডেন্সিগুলো ডিক্লয়ার করবো। নিচের কোডটুকু আপনার টেক্সট এডিটরে লিখে ফেলুন। (চাইলেও কপি করতে পারবেন না ☺ ☺ নিজের লিখতে হবে। ☺)



```

1 // Dependencies
2 const express = require('express');
3 const bodyParser = require('body-parser');
4 const app = express();
5 app.use(bodyParser.json());

```

এবার আমরা আমাদের সার্ভারের পোর্ট ডিক্লয়ার করবো। যেহেতু আমরা লোকালহোস্টে রান করছি, তাই আমাদের পোর্ট হবে 3000 (যদি আপনার 3000 তে কিছু রান না করা থাকে তবেই আপনি পোর্ট 3000 এ আপনার সার্ভার রান করতে পারবেন।)

```

1 // Set port
2 const port = process.env.PORT || 3000;
3
4 // Listing app
5 app.listen(port, function() {
6   console.log(`App is listing on PORT ${port}`);
7 })

```

আমাদের কাজ প্রায় ৮০ ভাগ শেষ। এখানে `PORT = process.env.PORT || 3000` দেয়ার কারন হলো আমরা যখন কোনো এপিআই বানাই তখন তা পাবলিশ করার জন্য আমরা একেকজন একেক environment চুজ করি। যেমন কেউ Heroku কেউবা আবার Glitch ইত্যাদিতে তাদের এপিআই হোস্ট করে। তো `process.env.PORT` দেয়ার বিশেষত্ব হলো আপনি যেই environment এই আপনার api হোস্ট করেন না কেন ও সেই environment এর পোর্ট টাই নিজে থেকে নিয়ে নেবে। আর `|| 3000` হলো যদি সে ওই environment থেকে কোনো পোর্ট রিটার্ন না পায় তবে সে ডিফল্ট হিসেবে 3000 কে ঠিক করে নেয়।

এখন আমাদের বাকি থাকলো শুধু রাউট ঠিক করা এবং ম্যাথম্যাটিক্যাল কিছু কাজ।

তবে তার আগে চলুন আমরা আগে দেখি আমাদের এতক্ষনের পরিশ্রম কতটুকু সফল হয়েছে। তার জন্য আমরা একটা গेट রিকোয়েস্ট করে দেখি। যদি আমরা আমাদের কাস্থিত রেজাল্ট পাই তবে সামনে এগোবো। গेट রিকোয়েস্টের নিচের কোডটুকু আমাদের `server.js` ফাইলে লিখে ফেলি।

```

1 // Sending data in /date route
2 app.get('/date', function(req, res, next){
3   res.send("Hello World");
4 })

```

এবার আমাদের কাজ হলো আমাদের প্রয়োজনীয় ডিপেন্ডেন্সিগুলো ইন্সটল করে ফেলা। আমরা সবার শুরুতে Expressjs এবং body-parser নামের দুইটা ডিপেন্ডেন্সি ডিক্লয়ার করেছিলাম। তো আমরা এবার এই তিনটা ডিপেন্ডেন্সি ইন্সটল করাবো। তার জন্য নিচের কমান্ডটুকু টার্মিনালে কপিপেস্ট করুন।

```
npm i express body-parser—save
```

ইন্সটলেশন শেষ হলে এখন আমাদের `server.js` ফাইল রান করতে হবে। এই ক্ষেত্রে আমি **nodemon** সাজেস্ট করবো। কারন নরমালি প্রতিবার `server.js` ফাইল আমাদের রান করার জন্য **node server.js** কমান্ড দিতে হবে। কিন্তু **nodemon** এ সবচেয়ে বড় সুবিধা

হলো ইহা আপনার প্রতিবার ফাইল চেঞ্জের পরে সেভ করা মাত্রই অটো রিস্টার্ট নেবে। বার বার আপনাকে ম্যানুয়ালী সার্ভার রিস্টার্ট দেয়া লাগবে না। আমি **nodemon** গ্লোবাল ইনস্টল করে নেবো। লোকাল ইনস্টল এ অসুবিধা হলো আমার অন্য কোনো প্রোজেক্টে **nodemon** লাগলে আমার আবার সেখানেও লোকাল ইনস্টল করতে হবে। তবে আপনারা চাইলে লোকাল ইনস্টল করতে পারেন।

লোকাল ইনস্টল এর জন্যঃ

```
npm i nodemon—save
```

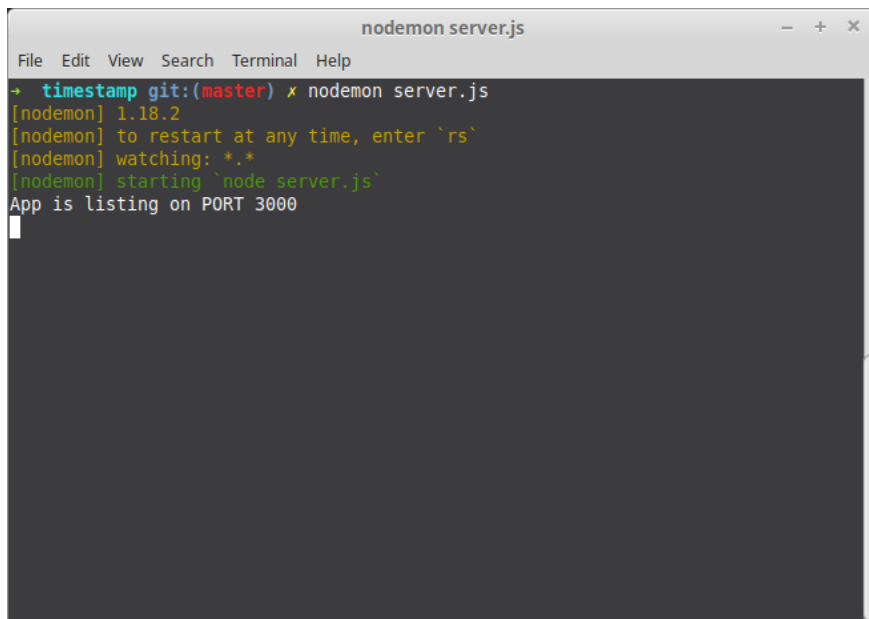
গ্লোবাল ইনস্টল এর জন্যঃ

গ্লোবাল ইনস্টল এ আপনাকে রুট এক্সেস দিতে হবে।

```
sudo npm i nodemon -g
```

nodemon ইনস্টল হয়ে গেলে আমরা আমাদের প্রোজেক্ট ডিরেক্টরীতে টার্মিনাল ওপেন করে **nodemon server.js** কমান্ড টি রান করাবো।

যদি আমাদের সবকিছু ঠিক থাকে তবে টার্মিনালে **App is listening on port 3000** লেখা আসবে।

A screenshot of a terminal window titled 'nodemon server.js'. The terminal shows the following output:

```
+ timestamp git:(master) x nodemon server.js
[nodemon] 1.18.2
[nodemon] to restart at any time, enter `rs`
[nodemon] watching: *.*
[nodemon] starting `node server.js`
App is listening on PORT 3000
```

এবার আমরা ব্রাউজার ওপেন করে <http://localhost:3000/date> এই URL এ গেলে আমরা নিচের মত কিছু দেখতে পাবো।



আমরা সফল। সুতরাং সামনে আগাইতে পারি। 😊

তো এবার আমাদের কাজ হলো, আমাদের প্রোজেক্টের মূল অংশে ফোকাস করা। এতক্ষণ আমরা যা করলাম তা শুধুই সেটাপ। অনেকটা বয়লারপ্লেট বলা যায়। এবার আমরা আমাদের মূল কাজ টা করবো।

Not before said everything. Actually it should have been said at the beginning. But in the beginning, because I want everyone to read the whole post. So in the middle of the post. Our work in this project is that we will create an API. Where we do not have a new version of RedDesign or UNIX version, which is why we do not input our API, we will output both of that date's red version and the UNIX version in Jason format. So let's start the original story.

So we will use our old routine of this work. Just after / date we will get a query called date_values. So this query here will work like a lot of us in user input. Since we will use the input later, so at the beginning I will put it on a variable called date_value. Now as the query changes, the page will be reloaded, so we can hold the date_value as a constant variable. And we need two more variables. To declare natural date and Unix date. And Eduato's value is variable. So these can not be construed as constants.

```
1 // Sending data in /date route
2 app.get('/date:date_values', function(req, res, next){
3   let natural_date, unix_date;
4   const date_value = req.params.date_values;
5 })
```

Now our code will look like this.

Now we have to check that the data that we are passing in the query is what the type of data is. Because if we input the readable date, then its data type will never be a **number**. Once again input the Unix version, its data type will never be **string**. Why not?

Because the readable date format can be of two types. 1 Jan 01, 2018 or 01-01-2018 or 01/01/2018. So now if it was in number format then here it is not possible to have **Jan** String. Again, if it were 01-01-2018 or 01/01/2018, then it would have been less than zero, one minus two thousand eight, or one part two and eight eight. And Unix will always be number format. Because it's just a number of numbers. So to check if there is an input number, we will use the JavaScript functions of the `isNaN()` function. The job of `isNaN()` is to pass a Boolean value if the argument given to him is NaN, then true will pass, or if not, then the fault. The full form of NaN is Not-A-Number.

Now, if we do not have an input number, that's the Red Date version. So we know that in a new `date()` function, when a date version is input, `new date()` will output us to that date. There can be a question here that we are the real version e input. So what do we need for the new `date()` function again? It is necessary to convert the original version to Unix version. If the date is red, then if we give 1000 percent of the date of the date, we will get the same Unix version of that date. And if the input date is unix, then with 1000 times we get the red version. And finally we will call our `toLocaleDateString` function. Because we only need the latest date version. The new `Date()` function also returns a lot more to us. And we'll pass the 'en-us' to the `toLocaleDateString` function. Because we need a stand-by version. So the code inside our gate request will be a lot like this.

```

1 app.get('/date/:date_values', function(req, res, next){
2   let natural_date, unix_date;
3   const date_value = req.params.date_values;
4
5   if (isNaN(date_value)) {
6     natural_date = new Date(date_value).toLocaleDateString('en-us');
7     unix_date = new Date(date_value).getTime()/1000;
8   } else {
9     unix_date = date_value;
10    natural_date = new Date(date_value * 1000).toLocaleDateString('en-us');
11  }
12
13  res.json({
14    unix_format: unix_date,
15    readable_format: natural_date
16  });
17 })

```

Now there's a little bit screwy here. The Red Date version or `natural_date` will return the date to us in mm / dd / yyyy format. I personally do not like this format. I like dd / mm / yyyy well. So we'll split `natural_date` to convert this mm / dd / yyyy to dd / mm / yyyy.

The viewpoint will be '/' because each section is separated by /. After split we get an array of 3 elements. Whose zero index is the month, and the first index is the day. So when we take this month to the day of the day and take the day to the month's place, our work is completed. It can be easily done with "moment.js". But here we will do this little thing ourselves. What is needed to change the entire format of a whole node module? 🤖

We will make a function to change the format. As its argument, we will pass our natural data variable. And the function will convert our mm / dd / yyyy to dd / mm / yyyy and return it. Then our code will do so

```

1 app.get('/date/:date_values', function(req, res, next){
2   let natural_date, unix_date;
3   const date_value = req.params.date_values;
4   const ddmmyy = (str) => {
5     let split = str.split('/');
6     return `${split[1]}-${split[0]}-${split[2]}`;
7   }
8
9   if (!isNaN(date_value)) {
10    natural_date = new Date(date_value).toLocaleDateString('en-us');
11    unix_date = new Date(date_value).getTime()/1000;
12  } else {
13    unix_date = date_value;
14    natural_date = new Date(date_value * 1000).toLocaleDateString('en-us');
15  }
16
17  res.json({
18    unix_format: unix_date,
19    readable_format: ddmmyy(natural_date)
20  });
21 })

```

Diameter Finish the job Now go to the / date roulette and enjoy the hot results in the hands. You can boil the amount of salt, sugar and cloves in the hot water for hot taste. Once boiled, serve hot hot with little chillies on top of your hard-pressed Timestamp Microservice 😊

```
1 // Dependencies
2 const express = require('express');
3 const bodyParser = require('body-parser');
4 const app = express();
5 app.use(bodyParser.json());
6
7 // Get function
8 app.get('/date/:date_values', function(req, res, next){
9   let natural_date, unix_date;
10   const date_value = req.params.date_values;
11   const ddmmyy = (str) => {
12     let split = str.split('/');
13     return `${split[1]}-${split[0]}-${split[2]}`;
14   }
15
16   if (isNaN(date_value)) {
17     natural_date = new Date(date_value).toLocaleDateString('en-us');
18     unix_date = new Date(date_value).getTime()/1000;
19   } else {
20     unix_date = date_value;
21     natural_date = new Date(date_value * 1000).toLocaleDateString('en-us');
22   }
23
24   res.json({
25     unix_format: unix_date,
26     readable_format: ddmmyy(natural_date)
27   });
28 });
29
30 // Set port
31 const port = process.env.PORT || 3000;
32
33 // Listing app
34 app.listen(port, function() {
35   console.log(`App is listing on PORT ${port}`);
36 });
```

Our project's full code. You can not copypast if you want. You have to type yourself. 😊

Goodbye like today Allah hafiz.

