Creating a REST API with Node.js

Using Node.js with Express, creating your own API is simple. You don't even need to render anything because we only need an API, not a pretty website. Take a look at the following code:

```
const express = require("express");
const path = require("path");

const app = express();
const port = process.env.PORT || "8888";

app.use(express.urlencoded({ extended: true }));
app.use(express.json()); //need for parsing JSON data from requests

app.get("/api/test", (req, res) => {
  let product = {
    name: "Test name",
    value: 5.00
  };
  res.json(product);
});

//server listening
app.listen(port, () => {
  console.log(`Listening on http://localhost:${port}`);
});
```

The important code is in yellow. The rest is standard stuff to set up a server and set up server listening.

In the `app.get()` this defines a GET request with the endpoint "***/api/test***". Inside the callback function, the only thing this does is print/send the JSON data (as defined in the `product` variable). Using `res.json()` is good because it sends appropriate JSON headers as well.

The above is simplified code to demonstrate the basics. If you want to define a POST request, just use `app.post()`.

Just as we did with form data, you can receive data (from the computer making the rest) via `req.query` for GET data or `req.body` for POST body data.

You can retrieve your JSON data in any way (e.g. by pulling data from a MongoDB database). How you retrieve or generate the data is up to you, but to create your own REST API, the important stuff is outlined above. For more endpoints, just add more page routes.

Enabling CORS

The last thing to do is to allow for requests from other computers. This is especially important if deploying your API online. You will need to install the cors module.

- **`npm i cors`**

Now in your code (you can put it in index.js), add the following:

```
const cors = require("cors");
```

Then somewhere after creating the **`app`** object/variable:

```
//allow requests from all domains (need it to deploy API)
//this should go after creating the app object
app.use(cors({
  origin: '*'
}));
```

The above lines will allow incoming requests from all domains/computers. If you want to restrict the domain (e.g. after deploying your application which is making requests to this API), you can change the * in the quotes to a domain name (e.g. `'https://yourdomain.com'`) or if you want to whitelist more than one domain, you can set origin to an array (e.g. **`origin: [ 'https://domain1.com', 'https://domain2.com']`**).

How does this all fit?

In a MERN/MEAN (MongoDB Express React Node or MongoDB Express Angular Node) stack, your Node backend would be an exposed API so that your frontend framework (e.g. React, Angular, etc) can make HTTP requests to this Node API to retrieve data. For example, in your React API, to retrieve the data from the "/api/test" endpoint, you would make an HTTP request to ***<domain>/api/test*** and use the JSON response data received.