

Teach your Parents how to build an API server in Node.js

Jon Wexler

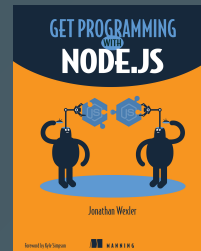
Author of *Get Programming with Node.js*

About me



Jon Wexler

- Senior Engineer at Bloomberg
- *Get Programming with Node.js*



What we'll cover

Another Node.js explanation

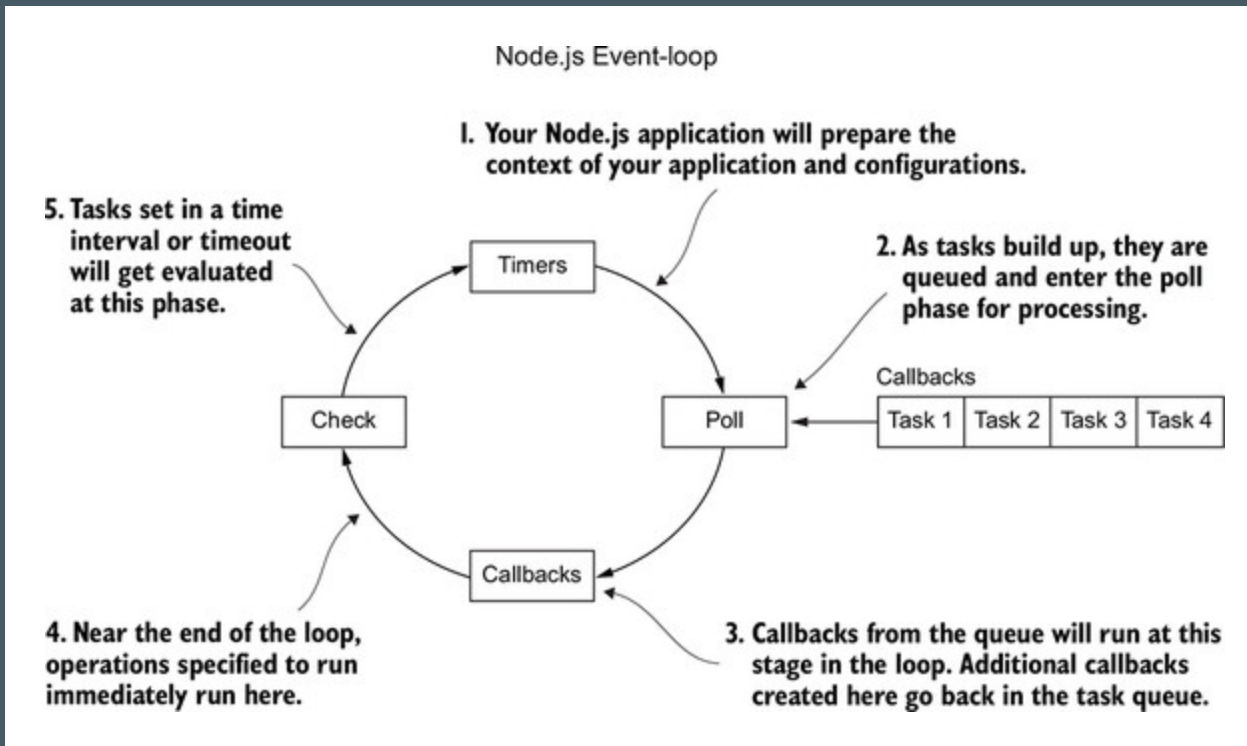
A look into APIs

Building an API with Express.js

Securing your API

What's Node.js?

- **Lesson 0** of *Get Programming with Node.js*
- A place to run JavaScript outside of your browser (*via Chrome V8*)



Install Node.js

- Go to nodejs.org
- Download `v14.15.5` (LTS version)
- Follow standard GUI steps
- Run these commands in a terminal window

```
node -v #Verify your version of node  
node #Enter the Node.js REPL environment
```

What we'll cover

Another Node.js explanation

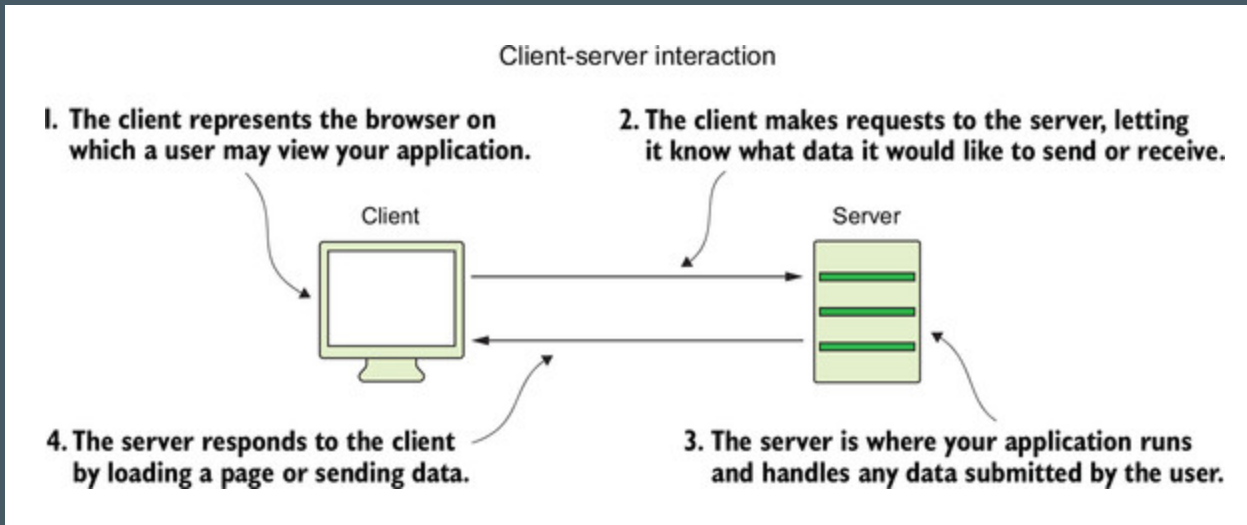
A look into APIs

Building an API with Express.js

Securing your API

What's an API?

- Application Programming Interface
- A way of communicating with an application server to get data
- Structure to handle requests in specific formats



What's in a request?

- **Request method:**

- `GET` = Read
- `POST` = Create
- `PATCH` / `PUT` = Update
- `DELETE` = Destroy

- **Headers**

- Metadata about the request
 - `Accept` = type of data you accept
 - `Authorization` = credentials

- **Payload**

- Body of the request

What's in a response?

- **Status Code**
 - 200 for success
- **Response headers**
 - metadata about the server
- **Response body**
 - data payload

Movie data API example

- Hit the api from the client [omdbapi](https://omdbapi.com/)
- Create an API key (*Your key will be emailed to you*)
- Create a new directory called `movie_api`
- add a file called `index.js`, and add the following:

Request with Node's HTTP module


```
const http = require('http');
const API_KEY = 'XXXXXXX';
const query = 'hacker';
const URI = `http://www.omdbapi.com/?apikey=${API_KEY}&s=${query}`;

http.get(URI, res => {
  let data = '';
  res.on('data', chunk => data += chunk);
  res.on('end', () => console.log(JSON.parse(data)));
}).on("error", err => console.log("Error: " + err.message));
```

The Response

```
{
  Search: [
    {
      Title: 'Hacker',
      Year: '2016',
      imdbID: 'tt3173594',
      Type: 'movie',
      Poster: 'https://m.media-amazon.com/images/M/MV5BYjk3ZWQ4ZmMtNzM1OS00NDQxLWJmZjctNzFlODAyODRkMjcyXkEyXkFqcGdeQXVyMjEwODIzODA@._V1_SX300.jpg'
    },
    {
      Title: 'The Hacker Wars',
      Year: '2014',
      imdbID: 'tt4047350',
      Type: 'movie',
      Poster: 'https://m.media-amazon.com/images/M/MV5BNzgwOTI0MjQwN15BM15BanBnXkFtZTgwMjAwNzQ3MzE@._V1_SX300.jpg'
    },
    {
      Title: 'The Hacker Wars',
      Year: '2014',
      imdbID: 'tt4056570',
      Type: 'movie',
      Poster: 'https://m.media-amazon.com/images/M/MV5BNzgwOTI0MjQwN15BM15BanBnXkFtZTgwMjAwNzQ3MzE@._V1_SX300.jpg'
    },
  ]
}
```

Make it cleaner with Axios

- We'll need to generate a `package.json` by running `npm init`
-  Then, `npm i axios`

```
const axios = require('axios');  
  
axios.get(URI)  
  .then(res => console.log(res.data))  
  .catch(error => console.log(error));
```

- There are other ways to query an API

REST

- Representational State Transfer
- Unique URI's (Uniform Resource Identifiers)

HOST

+

VERSION

+

RESOURCE

https://api.data.com

+

/v2

+

/item/:id/sub-item/:subId

- Differentiate data based on route and version

What we'll cover

Another Node.js explanation

A look into APIs

Building an API with Express.js


Securing your API

What if that API was your own server?

Control over:

- API response
- Who gets access
- Shape of the data

Why Express.js

- Great for RESTful routes
- Set up for middleware
-  Run `npm i express`
- Run `npm init` to initialize a new project called `express_api`
- create an `index.js` file to hold our main application code

Start the Express server

- Add the following code to `index.js`

```
const express = require('express');  
const app = express();  
const PORT = 8000;  
  
app.listen(PORT, () => console.log('Server is up!'));
```

- Run `node index` and notice what happens when we visit `localhost:8000`
- We have no routes set up

What we can notice

- No routes to handle the request yet
- Add a `GET` request
- Let's look at the network tab to see the response
- Let's use postman to test a request

```
app.get('/products', <handler>)
```

Data worth accessing

```
const data = {  
  products: [{  
    name: 'jeans',  
    inventory: 2  
  },  
  {  
    name: 'jackets',  
    inventory: 0  
  },  
  {  
    name: 'hats',  
    inventory: 12  
  }]  
}
```

A simple route

- Next, let's handle incoming API requests for products

```
const router = express.Router();

app.use('/v1', router);
router.route('/products')
  .get((req, res) => {
    res.status(200).send(data.products);
  })
```

We control the data

We can examine the request body and params once the request is received by our server!

- such as:

```
const { name } = req.params;
```

Dynamic params

```
function productRouter () {  
  const productRouter = express.Router();  
  productRouter.get('/', (req, res) => {  
    res.status(200).send(data.products)  
  })  
  productRouter.get('/:name', (req, res) => {  
    let result = data.products.filter(item => {  
      return item.name.includes(req.params.name)  
    });  
    res.status(200).send(result);  
  })  
  return productRouter;  
}
```

```
router.use('/products', productRouter())
```

Process JSON with Express middleware

- Express doesn't parse JSON by default so we'll need to add `app.use(express.json())` middleware that runs between the request being received and response going out.
- Then add a `POST` request to our `productRouter` like so:

```
productRouter.post('/', (req, res) => {  
  const { name, inventory } = req.body;  
  console.log(`Saving ${inventory} of ${name}.`);  
  res.status(200).send({ message: "Data Saved." });  
})
```


What we'll cover

Another Node.js explanation

A look into APIs


Building an API with Express.js

Securing your API

Securing your API

- Authentication with **Username** and **Password**
- Recieve token to send in an Authorization header
- Token is verified on the server before sending back data
- [JSON web tokens](#)

Simple security

- Similar to OMDb, we could provide an API key with specific access if an email is provided.
- As long as the user provides an email, we can send them a token and associate the two in our database to monitor or rate limit their requests.
-  For this we can use JSON web tokens by running
`npm i jsonwebtoken`
- Then, in `index.js` add:

```
const jwt = require('jsonwebtoken');
```

JWTs

- JWTs can be used to sign the token with the user's information
This is particularly useful after first verifying the user in your database.
- It takes **user info**, a **server secret key**, and an **expiration date**

```
const token = jwt.sign({ email }, SECRET_KEY, { expiresIn: '24h' });
```

- This ensures the token is associated with an email and cannot be misused past a certain date.

Use JWTs to verify a request

- First, define `const SECRET_KEY = 'get_programming';`

```
function authRouter () {  
  const authRouter = express.Router();  
  authRouter.post('/key', (req, res) => {  
    const { email } = req.body;  
    const token = jwt.sign({ email }, SECRET_KEY, { expiresIn: '24h' });  
    console.log(`Saving ${email} with token: ${token}.`);  
    res.status(200).send({ token });  
  })  
  return authRouter;  
}
```

Verify JWTs through middleware

- Within your `productRouter` we'll add `productRouter.use(authMiddleware);`
- Then we can define the middleware function as:

```
function authMiddleware (req, res, next) {  
  const auth = req.headers.authorization;  
  const decoded = jwt.verify(auth, SECRET_KEY);  
  if (decoded && decoded.email) {  
    next();  
  } else {  
    res.send({ message: 'Unauthorized request.' });  
  }  
}
```

Make an authenticated API request

- Submit the same `GET` request, but this time with an Authorization header

```
Authorization: eyJhbGciOiJIUzI1N
```

- In this way, we can secure the type of data users can access for demo purposes, versus for paid accounts.

What we covered

Another Node.js explanation

A look into APIs

Building an API with Express.js

Securing your API

Thank You