

English (auto-generated)
Click ⚙ for settings

Build a REST API



NODE | EXPRESS | MONGO

hello and welcome to this workshop on
building a REST API with node express

Subtitles/closed captions (c)



SUBSCRIBE

English (auto-generated)
Click ⚙ for settings

What is HTTP...

HyperText Transfer Protocol

- Application Layer Protocol
- Built on top of TCP/IP protocol
- Rules for transferring resources
- Every HTTP Request is executed independently without the knowledge of requests that came before it

so what does HTTP it stands

Subtitles/closed captions (c)



...What is HTTP

HyperText Transfer Protocol

- HTTP is stateless
- TCP/IP is not stateless
- Payload (body/data) can be anything as long as it is defined in header

of the requests that came before it and
that's what makes HTTP stateless

Subtitles/closed captions (c)



...What is HTTP

HyperText Transfer Protocol

- HTTP is stateless
- TCP/IP is not stateless
- Payload (body/data) can be anything as long as it is defined in header

HTML file versus a JavaScript file
embedded in

Subtitles/closed captions (c)



What is REST...

Representational State Transfer

- Architectural Pattern with design guidelines
- HTTP is usually the underlying protocol
- Use HTTP methods explicitly
- Every RESTful resource has a unique ID

interesting because you can reference different

Subtitles/closed captions (c)



...What is REST

Representational State Transfer

- Client state is not persisted between requests
- Caching Policy for responses
- Separation of concerns between clients and servers
- Layered system

subsequent requests so that it improves
the application performance it

Subtitles/closed captions (c)



What is an API...

Application Programming Interface

- Defines server-side functions that are supported
- Where requests should be made
- Format of the request and response

be made so essentially it's going to tell you what the pattern

Subtitles/closed captions (c)



...What is an API

Application Programming
Interface

- No standard way of writing APIs
- REST provides guidelines
- CRUD Operations
- Create | Read | Update | Delete

records so essentially when you're
making a call over an API

Subtitles/closed captions (c)



IDE

Integrated Development Environment

- VS Code
- Atom
- Webstorm (Paid / Trial)

which allows us to look at the data in
so as

Subtitles/closed captions (c)



Node.js

Javascript Runtime

- Node.js 8.9.3+
- NPM (Node Package Manager)
- Using natively supported ES6/7
- <https://node.green>

babel or any other transpilers if you
can

Subtitles/closed captions (c)



MongoDB

NoSQL Database

- Local Installation
- mLab Cloud
- Docker

MongoDB can be installed locally or you
can get an instance via the EM lab cloud

Subtitles/closed captions (c)



REST Client

API Testing

- Insomnia
- Postman

most of the other IDs require you to
configure

Subtitles/closed captions (c)



REST Client

API Testing

- Insomnia
- Postman

lends itself very well to as well
because you're storing JavaScript

Subtitles/closed captions (c)



Application Setup

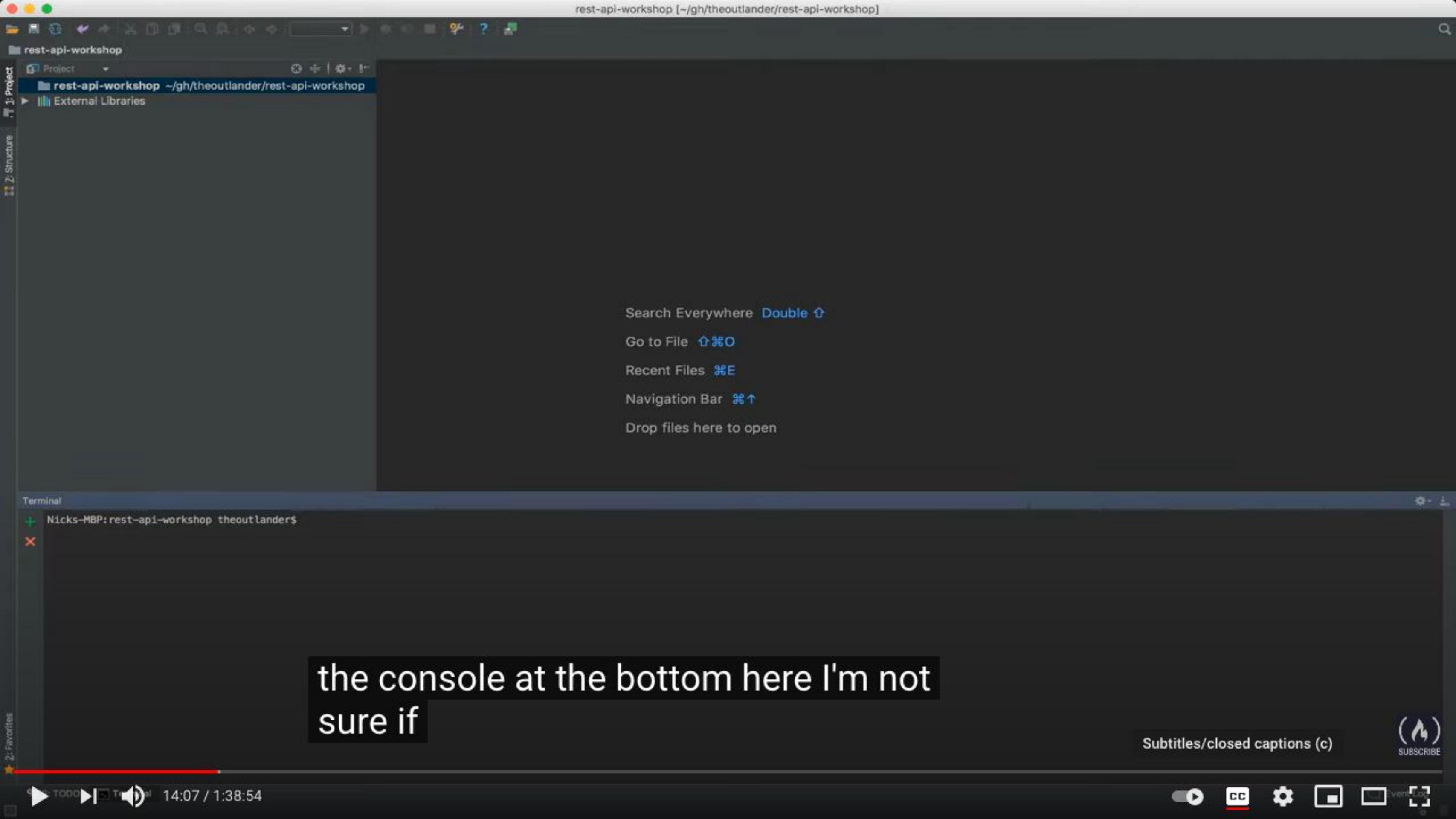
Getting Started

- Project Folder
- Initialize Node.js Project
- Hello World
- Startup Scripts

project folder and then you will open up
your IDE either

Subtitles/closed captions (c)





the console at the bottom here I'm not
sure if

Subtitles/closed captions (c)



rest-api-workshop [~/gh/theoutlander/rest-api-workshop] - .../package.json [rest-api-workshop]

rest-api-workshop package.json

Project rest-api-workshop ~/gh/theoutlander/rest-api-workshop package.json External Libraries

package.json

```
1 {  
2   "name": "rest-api-workshop",  
3   "version": "1.0.0",  
4   "description": "",  
5   "main": "index.js",  
6   "scripts": {  
7     "test": "echo \"Error: no test specified\" && exit 1"  
8   },  
9   "keywords": [],  
10  "author": "",  
11  "license": "ISC"  
12}  
13
```

Terminal

```
+ Nicks-MBP:rest-api-workshop theoutlander$ npm init -y  
Wrote to /Users/theoutlander/gh/theoutlander/rest-api-workshop/package.json:  
  
{  
  "name": "rest-api-workshop",  
  "version": "1.0.0",  
  "description": "",  
  "main": "index.js",  
  "scripts": {  
    "test": "echo \"Error: no test specified\" && exit 1"  
  },  
  "keywords": [],  
  "author": "",  
  "license": "ISC"  
}
```

that he can run as NPM commands and we'll get to that in a bit so

Nicks-MBP:rest-api-workshop theoutlander\$

Subtitles/closed captions (c) (A)
SUBSCRIBE

15:07 / 1:38:54

The screenshot shows a code editor interface with a dark theme. On the left is a sidebar titled 'Project' showing a folder structure for 'rest-api-workshop'. Inside 'src' are 'package.json' and 'index.js'. The 'index.js' file is open in the main editor area, containing the single line of code: 'console.log('Hello World')'. The status bar at the bottom indicates the file is 1 line long.

Terminal

```
+ Nicks-MBP:rest-api-workshop theoutlander$ npm init -y
Wrote to /Users/theoutlander/gh/theoutlander/rest-api-workshop/package.json:
```

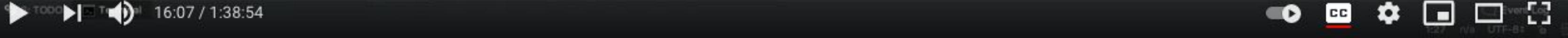
```
{
  "name": "rest-api-workshop",
  "version": "1.0.0",
  "description": "",
  "main": "index.js",
  "scripts": {
    "test": "echo \\\"Error: no test specified\\\" &> test.txt"
  },
  "keywords": [],
  "author": "",
  "license": "ISC"
}
```

pointing to the file so it's under the source folder slash index is and

Subtitles/closed captions (c)



Nicks-MBP:rest-api-workshop theoutlander\$ node src/index.js



```
rest-api-workshop [~/gh/theoutlander/rest-api-workshop] - .../package.json [rest-api-workshop]
Project rest-api-workshop ~/gh/theoutlander/rest-api-workshop package.json index.js
  src
    package.json
External Libraries

1  {
2    "name": "rest-api-workshop",
3    "version": "1.0.0",
4    "description": "",
5    "main": "index.js",
6    "scripts": {
7      "start": "node src/index.js",
8      "test": "echo \\\"Error: no test specified\\\" && exit 1"
9    },
10   "keywords": [],
11   "author": "",
12   "license": "ISC"
13 }
14
```

scripts > start

```
Nicks-MBP:rest-api-workshop theoutlander$ npm st
```

to save that I will clear my console and
I'm going to run NPM start

Subtitles/closed captions (c)



Express

Web Server

is basically a lightweight web server
what

- Install Dependency
- Reference the express library
- Create express application instance

Subtitles/closed captions (c)



rest-api-workshop [~/gh/theoutlander/rest-api-workshop] - .../package.json [rest-api-workshop]

rest-api-workshop package.json

Project rest-api-workshop ~/gh/theoutlander/rest-api-workshop node_modules library root src package.json package-lock.json External Libraries

package.json index.js

```
1  {
2      "name": "rest-api-workshop",
3      "version": "1.0.0",
4      "description": "",
5      "main": "index.js",
6      "scripts": {
7          "start": "node src/index.js",
8          "test": "echo \\\"Error: no test specified\\\" && exit 1"
9      },
10     "keywords": [],
11     "author": "",
12     "license": "ISC",
13     "dependencies": {
14         "express": "^4.16.3"
15     }
16 }
17
```

dependencies

Terminal

```
> rest-api-workshop@1.0.0 start /Users/theoutlander/gh/theoutlander/rest-api-workshop
> node src/index.js

Hello World
Nicks-MBP:rest-api-workshop theoutlander$ npm run start

> rest-api-workshop@1.0.0 start /Users/theoutlander/gh/theoutlander/rest-api-workshop
> node src/index.js

Hello World
Nicks-MBP:rest-api-workshop theoutlander$ npm start
npm notice created a lockfile as package-lock.json
npm warn rest-api-workshop@1.0.0 No description
npm warn rest-api-workshop@1.0.0 No repository
+ express@4.16.3
added 49 packages from 47 contributors in 2.784s
Nicks-MBP:rest-api-workshop theoutlander$
```

is running it's going to have Express available to it

Subtitles/closed captions (c) (A)
SUBSCRIBE

19:07 / 1:38:54

CC

A screenshot of a video player interface displaying a terminal window within a code editor. The terminal shows the command to start a Node.js application and its output.

```
> rest-api-workshop@1.0.0 start /Users/theoutlander/gh/theoutlander/rest-api-workshop
> node src/index.js

Hello World
Nicks-MBP:rest-api-workshop theoutlander$ npm run start

> rest-api-workshop@1.0.0 start /Users/theoutlander/gh/theoutlander/rest-api-workshop
> node src/index.js

Hello World
Nicks-MBP:rest-api-workshop theoutlander$ npm
npm notice created a lockfile as package-lock.json
npm warn rest-api-workshop@1.0.0 No description
npm warn rest-api-workshop@1.0.0 No repository
+ express@4.16.3
added 49 packages from 47 contributors in 2.784s
Nicks-MBP:rest-api-workshop theoutlander$
```

serving some static files so I'm going to create a folder

Subtitles/closed captions (c) (A)
SUBSCRIBE

A screenshot of a video player interface displaying a code editor and a terminal window. The video player controls are visible at the bottom.

The code editor shows the file `src/index.js` with the following content:

```
let express = require('express')
let app = express()
```

The terminal window shows the following command history and output:

```
> rest-api-workshop@1.0.0 start /Users/theoutlander/gh/theoutlander/rest-api-workshop
> node src/index.js

Hello World
Nicks-MBP:rest-api-workshop theoutlander$ npm run start

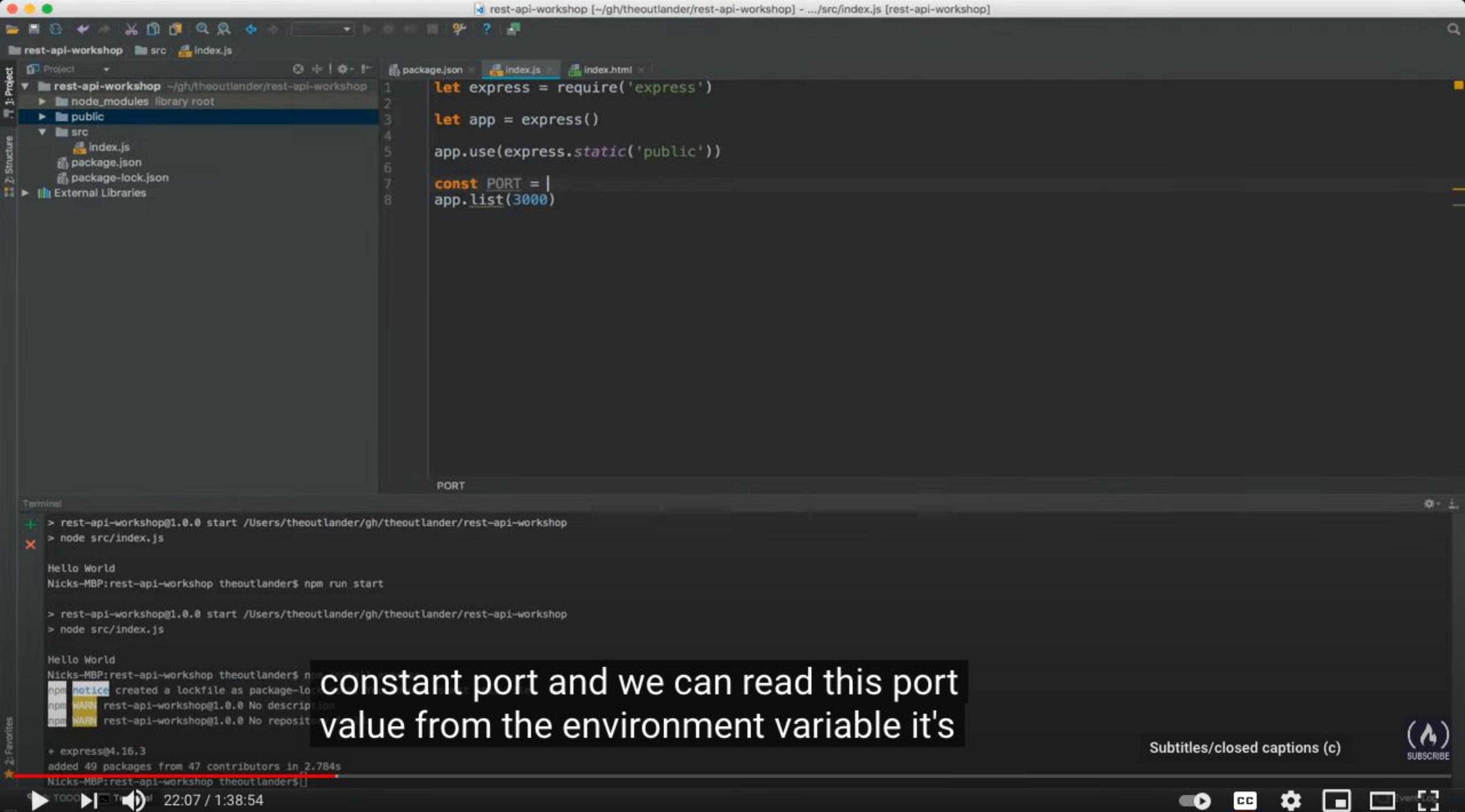
> rest-api-workshop@1.0.0 start /Users/theoutlander/gh/theoutlander/rest-api-workshop
> node src/index.js

Hello World
Nicks-MBP:rest-api-workshop theoutlander$ npm
npm notice created a lockfile as package-lock.json
npm WARN rest-api-workshop@1.0.0 No description
npm WARN rest-api-workshop@1.0.0 No repository
+ express@4.16.3
added 49 packages from 47 contributors in 2.784s
Nicks-MBP:rest-api-workshop theoutlander$
```

A large white text box with a black border is overlaid on the terminal area, containing the text:

called express dot static so I'm going
to tell

Subtitles/closed captions (c)



The screenshot shows a terminal window with the following text:

```
+ rest-api-workshop@1.0.0 start /Users/theoutlander/gh/theoutlander/rest-api-workshop
> node src/index.js

Hello World
Nicks-MBP:rest-api-workshop theoutlander$ npm run start

> rest-api-workshop@1.0.0 start /Users/theoutlander/gh/theoutlander/rest-api-workshop
> node src/index.js

Hello World
Nicks-MBP:rest-api-workshop theoutlander$ npm notice created a lockfile as package-lock.json
npm WARN rest-api-workshop@1.0.0 No description
npm WARN rest-api-workshop@1.0.0 No repository
+ express@4.16.3
added 49 packages from 47 contributors in 2.784s
Nicks-MBP:rest-api-workshop theoutlander$
```

A large black rectangular box covers the middle portion of the terminal output, containing the text:

constant port and we can read this port value from the environment variable it's

Subtitles/closed captions (c)



A screenshot of a dark-themed code editor, likely WebStorm, showing a project structure and code files. The project structure on the left shows a folder named 'rest-api-workshop' containing 'node_modules', 'public', and 'src' folders. The 'src' folder contains 'index.js', 'package.json', and 'package-lock.json'. The current file being edited is 'index.js'. The code in 'index.js' is as follows:

```
let express = require('express')
let app = express()
app.use(express.static('public'))
const PORT = process.env.PORT || 3000
app.listen(PORT, () => console.info(`Server has started on ${PORT}`))
```

The code editor has tabs for 'package.json', 'index.js', and 'index.html'. A tooltip 'callback for listen()' is visible near the 'listen()' call. Below the code editor is a terminal window showing the command 'node' entered.

and run the server here again by calling
node I'm

Subtitles/closed captions (c)





file into public folder and you can
serve it through through

Subtitles/closed captions (c)



Route

Mount to Express

- GET Request
- Request Parameters
- Query Parameters

you may have heard about something
called

Subtitles/closed captions (c)



The screenshot shows a code editor interface with a dark theme. In the top navigation bar, there are icons for file operations like new, open, save, and search. The title bar indicates the current file is `rest-api-workshop` and the specific file is `routes/person.js`. Below the title bar, the main area displays a project structure on the left and code editor panes on the right.

Project Structure:

- rest-api-workshop** (~/gh/theoutlander/rest-api-workshop)
 - node_modules
 - public
 - src**
 - routes**
 - person.js**
 - index.js
 - package.json
 - package-lock.json
- External Libraries

Code Editor:

The code editor panes show the following files:

- `package.json`
- `index.js`
- `person.js` (highlighted)
- `index.html`

Terminal

```
Nicks-MBP:rest-api-workshop theoutlander$ npm start
> rest-api-workshop@1.0.0 start /Users/theoutlander/gh/theoutlander/rest-api-workshop
> node src/index.js

Server has started on 3000
```

there are advantages to it and I will
talk about it in a bit so essentially

Subtitles/closed captions (c)



The screenshot shows a code editor interface with a dark theme. On the left, the project structure is displayed under the 'Project' tab, showing a folder named 'rest-api-workshop' containing 'node_modules', 'public', 'src' (which contains 'index.js', 'package.json', 'package-lock.json', and 'routes' which further contains 'person.js'), and 'External Libraries'. The main editor area has tabs for 'package.json', 'index.js', 'person.js', and 'index.html'. The 'person.js' tab is active, showing the following code:

```
1 let path: PathParams, ... handlers: RequestHandler
2 let path: PathParams, ... handlers: RequestHandlerParams
3
4 router.get('/person', )
5
6
7 module.exports = router
```

Terminal

```
+ Nicks-MBP:rest-api-workshop theoutlander$ npm start
> rest-api-workshop@1.0.0 start /Users/theoutlander/gh/theoutlander/rest-api-workshop
> node src/index.js

Server has started on 3000
```

usually a route has a callback method
that gets called

Subtitles/closed captions (c)



The screenshot shows a code editor interface with a dark theme. On the left, the project structure is displayed:

- rest-api-workshop (~/gh/theoutlander/rest-api-workshop)
- node_modules (library root)
- public
- src
 - routes
 - person.js
 - Index.js
 - package.json
 - package-lock.json
- External Libraries

The main editor area shows the content of `Index.js`:

```
1 let express = require('express')
2
3 let app = express()
4
5 let personRoute =
6 app.use(express.static('public'))
7
8 const PORT = process.env.PORT || 3000
9 app.listen(PORT, () => console.info(`Server has started on ${PORT}`))
```

Below the editor is a terminal window:

```
+ Nicks-MBP:rest-api-workshop theoutlander$ npm start
> rest-api-workshop@1.0.0 start /Users/theoutlander/gh/theoutlander/rest-api-workshop
> node src/index.js
Server has started on 3000
```

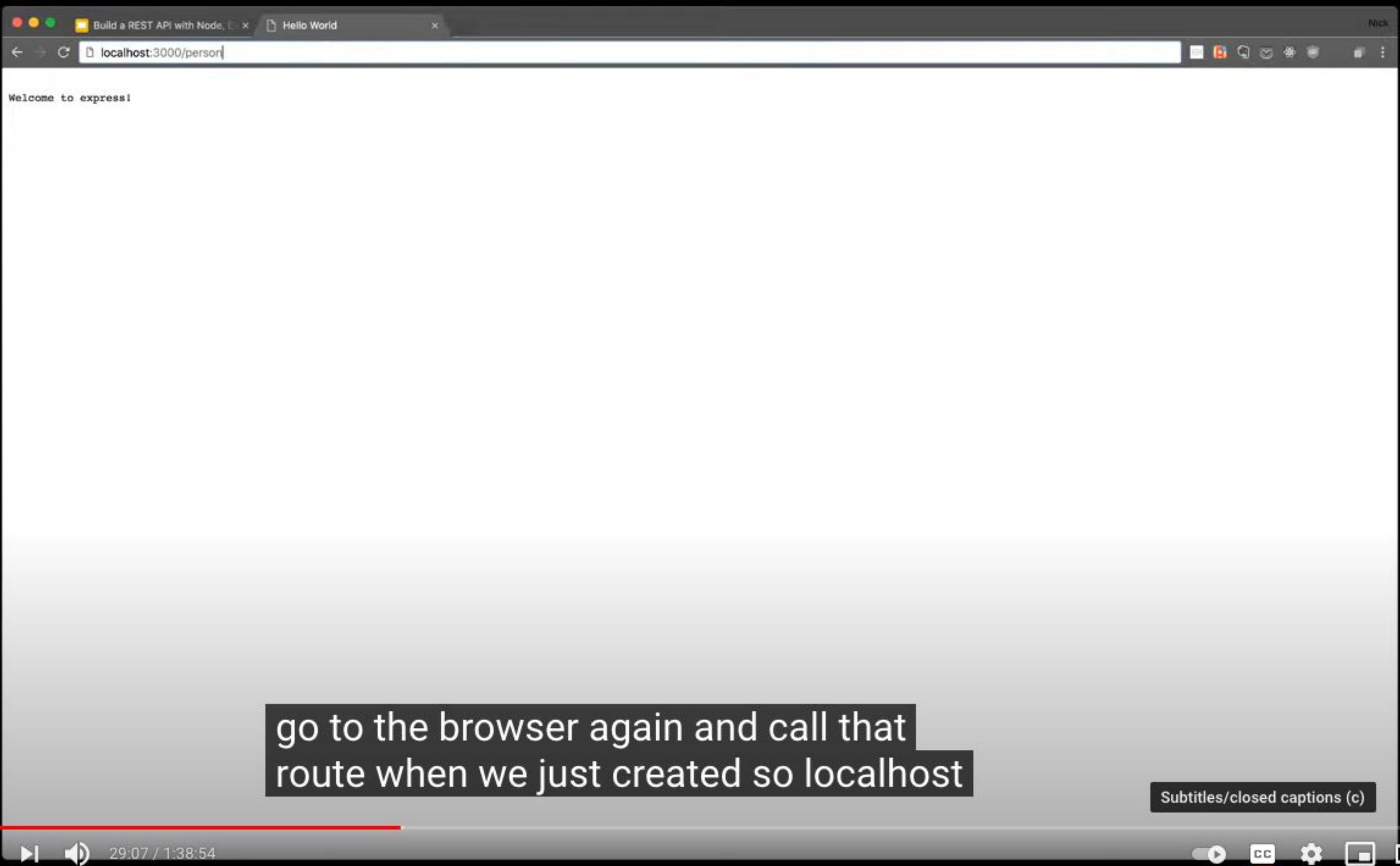
A large text overlay in the center of the screen reads:

we just created so
let's call it person route equals

At the bottom right, there are video player controls and subtitles information:

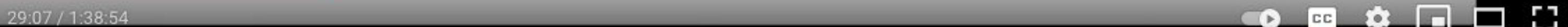
Subtitles/closed captions (c) (4)
SUBSCRIBE

At the very bottom, a progress bar indicates the video is at 28:07 / 1:38:54.



go to the browser again and call that route when we just created so localhost

Subtitles/closed captions (c)



29:07 / 1:38:54

Insomnia – GET Person

Insomnia

GET https://api.myproduct.com/v1/users Send

No Environment Cookies

Filter

REST API Workshop

GET Person

Projects

GET Person

Header Docs

GET POST PUT PATCH DELETE OPTIONS HEAD Custom Method

Send Request ⌘Enter

Focus Url Bar ⌘L

Manage Cookies ⌘K

Edit Environments ⌘E

Select a body type from above

operations map to for requests you have to create which happens on a post the

Subtitles/closed captions (c) (4)

SUBSCRIBE

▶ ▶! 🔊 30:07 / 1:38:54

▶ 🔍 CC ⚙️ 📺 🎥 []



Insomnia

GET https://api.myproduct.com/v1/users:

Send

No Environment

Cookies

GET

Query

Header

Docs

Filter:



REST API Workshop

GET GET Person

Projects

POST

PUT

PATCH

DELETE

OPTIONS

HEAD

Custom Method



Select a body type from above

Send Request

⌘Enter

Focus Url Bar

⌘L

Manage Cookies

⌘K

Edit Environments

⌘E

than postmen so I started out using
postman

Subtitles/closed captions (c)



Insomnia - GET Person

Send 200 OK TIME 5.44 ms SIZE 27 B

No Environment Cookies Body Auth Query Header Docs Preview Header Cookie Timeline

You have requested a person

REST API Workshop

GET Person Projects

Select a body type from above

routes that we've created and test them out over here one of the

Subtitles/closed captions (c) SUBSCRIBE

32:07 / 1:38:54

Insomnia – GET Person

GET http://localhost:3000/person

No Environment | Cookies | Body | Auto

REST API Workshop

GET Person

Projects

Generate Client Code

Node.js ▾ Request ▾

Copy to Clipboard

```
1 var request = require("request");
2
3 var jar = request.jar();
4 jar.setCookie(request.cookie("connect.sid=s%253A8Iq233D3GSmRRHoqqAX4t_7sV1ZNBAfe.6Ys4HkUv0h91dGSvXVoSzRSv7MzXcgFZC4zU84jbb8Y"), "http://localhost:3000/person");
5
6 var options = { method: 'GET',
7   url: 'http://localhost:3000/person',
8   jar: 'JAR' };
9
10 request(options, function (error, response, body) {
11   if (error) throw new Error(error);
12
13   console.log(body);
14 });
15
```

choose request because it's the shortest piece of code here and it's essentially

Subtitles/closed captions (c)

SubSCRIBE

Code snippets generated by httpsnippet

33:07 / 1:38:54

Done

The screenshot shows a code editor interface with a dark theme. In the top left, there's a project tree for 'rest-api-workshop' containing files like index.js, person.js, and index.html. The main editor area is focused on the 'package.json' file, which contains the following JSON:

```
1  {
2    "name": "rest-api-workshop",
3    "version": "1.0.0",
4    "description": "",
5    "main": "index.js",
6    "scripts": {
7      "start": "node src/index.js",
8      "test": "echo \\\"Error: no test specified\\\" && exit 1"
9    },
10   "keywords": [],
11   "author": "",
12   "license": "ISC",
13   "dependencies": {
14     "express": "^4.16.3"
15   }
16 }
17
```

Below the editor is a terminal window showing the command 'npm start' being run. The output indicates that the server has started on port 3000.

```
+ Nicks-MBP:rest-api-workshop theoutlander$ npm start
> rest-api-workshop@1.0.0 start /Users/theoutlander/gh/theoutlander/rest-api-workshop
> node src/index.js

Server has started on 3000
^Cnicks-MBP:rest-api-workshop theoutlander$ npm start
> rest-api-workshop@1.0.0 start /Users/theoutlander/gh/theoutlander/rest-api-workshop
> node src/index.js

Server has started on 3000
^Cnicks-MBP:rest-api-workshop theoutlander$
```

A large, semi-transparent text box is overlaid on the bottom right of the terminal area, containing the text:

however in this case because we're using
the latest node version NPM

At the very bottom of the screen, there are standard video player controls for volume, playback, and settings.

The screenshot shows a code editor interface with a dark theme. On the left, the Project Explorer displays a folder structure for a project named "rest-api-workshop". Inside "src", there are "routes" and "person.js" files. The "package.json" file is selected and open in the main editor area. The code in "package.json" is as follows:

```
1  {
2    "name": "rest-api-workshop",
3    "version": "1.0.0",
4    "description": "",
5    "main": "index.js",
6    "scripts": {
7      "start": "node src/index.js",
8      "test": "echo \\\"Error: no test specified\\\" && exit 1"
9    },
10   "keywords": [],
11   "author": "",
12   "license": "ISC",
13   "dependencies": {
14     "express": "^4.16.3"
15   }
16 }
17
```

Below the editor is a Terminal window showing the command-line output of npm install and node install. The output includes messages about fsevents and nodemon being installed.

```
+ fsevents@1.1.3 install /Users/theoutlander/gh/theoutlander/rest-api-workshop/node_modules/fsevents
> node install

[fsevents] Success: "/Users/theoutlander/gh/theoutlander/rest-api-workshop/node_modules/fsevents/lib/binding/Release/node-v57-darwin-x64/fse.node" already installed
Pass --update-binary to reinstall or --build-from-source to recompile

> nodemon@1.17.2 postinstall /Users/theoutlander/gh/theoutlander/rest-api-workshop/node_modules/nodemon
> node -e "console.log(`\u001b[32mLove nodemon? You can now support the project via the open collective:\u001b[22m\u001b[39m\n > \u001b[96m\u001b[1mhttps://opencollective.com/nodemon/donate\u001b[0m\n`)" || exit 0

Love nodemon? You can now support the project via the open collective
> https://opencollective.com/nodemon/donate

npm WARN rest-api-workshop@1.0.0 No description
npm WARN rest-api-workshop@1.0.0 No repository
+ nodemon@1.17.2
added 355 packages from 201 contributors in 11.914s
Nicks-MBP:rest-api-workshop theoutlander$ npm i nodemon -D
```

A large white text overlay "all right so we have node one created here what we're" is centered over the terminal window.

all right so we have node one created
here what we're

Subtitles/closed captions (c)





folders these are things you can look up
in the documentation as you get

Subtitles/closed captions (c)



The screenshot shows a code editor interface with a dark theme. On the left, the project structure is displayed in a tree view:

- rest-api-workshop (selected)
- node_modules
- public
- src
 - routes
 - person.js (selected)
 - index.js
- package.json
- package-lock.json

The main editor area shows the content of `person.js`:

```
1 let express = require('express')
2 let router = express.Router()
3
4 router.get('/person', (req, res) => {
5   res.send('You have requested a person')
6 })
7
8 router.get('/person/:name', (req, res) => {
9   res.send('You have requested a person')
10 })
11
12
13
14
15 module.exports = router
```

Terminal

```
+ Nicks-MBP:rest-api-workshop theoutlander$ npm run start-watch
> rest-api-workshop@1.0.0 start-watch /Users/theoutlander/gh/theoutlander/rest-api-workshop
> nodemon src/index.js

[nodemon] 1.17.2
[nodemon] to restart at any time, enter 'rs'
[nodemon] watching: ***!
[nodemon] starting 'node src/index.js'
Server has started on 3000
```

mapped to a variable so let's call it a name and we

Subtitles/closed captions (c)



The screenshot shows a video player interface with a dark theme. At the top, there's a navigation bar with icons for back, forward, search, and other controls. Below it is a file browser window titled 'rest-api-workshop' showing the project structure:

- Project
- rest-api-workshop (~/gh/theoutlander/rest-api-workshop)
 - node_modules library root
 - public
 - src
 - routes
 - person.js
 - index.js
 - package.json
 - package-lock.json
- External Libraries

The 'person.js' file is selected in the file tree and is displayed in the main code editor area. The code is as follows:

```
1 let express = require('express')
2 let router = express.Router()
3
4 router.get('/person', (req, res) => {
5   res.send('You have requested a person')
6 })
7
8 router.get('/person/:name', (req, res) => {
9   res.send(`You have requested a person ${req.params.name}`)
10})
11
12
13
14
15 module.exports = router
```

A tooltip 'callback for get()' is visible near the first 'get' method definition.

Below the code editor is a terminal window showing the command line output:

```
Nicks-MBP:rest-api-workshop theoutlander$ npm run start-watch
> rest-api-workshop@1.0.0 start-watch /Users/theoutlander/gh/theoutlander/rest-api-workshop
> nodemon src/index.js

[nodemon] 1.17.2
[nodemon] to restart at any time, enter 'rs'
[nodemon] watching: ***!
[nodemon] starting 'node src/index.js'
Server has started on 3000
[nodemon] restarting due to changes...
[nodemon] starting 'node src/index.js'
Server has started on 3000
```

A large white text box with a black border is overlaid on the terminal output, containing the text:

notice that our server automatically restarted so

At the bottom right of the video player, there are additional controls: a play button, a closed captioning icon ('CC'), a settings gear icon, a square icon, and a 'venLog' icon.

The screenshot shows a code editor interface with a dark theme. On the left, there's a sidebar titled 'Project' showing the directory structure of a 'rest-api-workshop' project. Inside 'src/routes', there are files named 'index.js', 'person.js', and 'Index.html'. The 'person.js' file is currently selected and open in the main editor area. The code in 'person.js' is as follows:

```
1 let express = require('express')
2 let router = express.Router()
3
4 router.get('/person', (req, res) => {
5   res.send('You have requested a person')
6 })
7
8 router.get('/person/:name', (req, res) => {
9   res.send(`You have requested a person ${req.params.name}`)
10})
11
12
13
14
15
16 module.exports = router
```

Terminal

```
+ Nicks-MBP:rest-api-workshop theoutlander$ npm run start-watch
> rest-api-workshop@1.0.0 start-watch /Users/theoutlander/gh/theoutlander/rest-api-workshop
> nodemon src/index.js

[nodemon] 1.17.2
[nodemon] to restart at any time, enter 'rs'
[nodemon] watching: ***!
[nodemon] starting 'node src/index.js'
Server has started on 3000
[nodemon] restarting due to changes...
[nodemon] starting 'node src/index.js'
Server has started on 3000
```

the other thing we can do is so this here is

Subtitles/closed captions (c)



The screenshot shows a code editor interface with a dark theme. On the left, the project structure is displayed in a tree view:

- rest-api-workshop (library root)
- node_modules
- public
- src
 - routes
 - person.js
 - index.js
- package.json
- package-lock.json

The right pane contains the content of the `person.js` file:

```
1 let express = require('express')
2 let router = express.Router()
3
4 // QueryString => query property on the request object
5 router.get('/person', (req, res) => {
6   if(req.query.name) {
7     |
8   }
9   else
10  {
11    res.send('You have requested a person')
12  }
13})
14
15 // Params property on the request object
16 router.get('/person/:name', (req, res) => {
17   res.send(`You have requested a person ${req.params.name}`)
18 })
19
20
21
22
23 module.exports = router
```

A tooltip "callback for get()" is visible near the bottom of the code editor.

Below the code editor is a terminal window showing the command line output:

```
Nicks-MBP:rest-api-workshop theoutlander$ npm run start-watch
> rest-api-workshop@1.0.0 start-watch /Users/theoutlander/gh/theoutlander/rest-api-workshop
> nodemon src/index.js

[nodemon] 1.17.2
[nodemon] to restart at any time, enter 'rs'
[nodemon] watching: ***!
[nodemon] starting 'node src/index.js'
Server has started on 3000
[nodemon] restarting due to changes...
[nodemon] starting 'node src/index.js'
Server has started on 3000
```

A large white text box with a black border is overlaid on the terminal area, containing the following text:

will do something there else we're going
to handle it like we were originally so

At the bottom right of the video player, there are controls for Subtitles/closed captions (c), a volume icon, and a subscribe button.

The screenshot shows a video player interface with a dark theme. At the top, there's a navigation bar with icons for back, forward, search, and other controls. Below it is a file browser window titled "rest-api-workshop". The "src" folder is expanded, showing "routes" and "person.js". The "person.js" file is selected and open in the main editor area. The code in "person.js" is as follows:

```
let express = require('express')
let router = express.Router()

// QueryString => query property on the request object
// localhost:3000/person?name=thomas&age=20
router.get('/person', (req, res) => {
  if(req.query.name) {
    res.send(`You have requested a person ${req.query.name}`)
  } else {
    res.send('You have requested a person')
  }
})

// Params property on the request object
router.get('/person/:name', (req, res) => {
  res.send(`You have requested a person ${req.params.name}`)
})

module.exports = router
```

Below the editor is a terminal window showing the command line output:

```
Nicks-MBP:rest-api-workshop theoutlander$ npm run start-watch
> rest-api-workshop@1.0.0 start-watch /Users/theoutlander/gh/theoutlander/rest-api-workshop
> nodemon src/index.js

[nodemon] 1.17.2
[nodemon] to restart at any time, enter 'rs'
[nodemon] watching: ***!
[nodemon] starting 'node src/index.js'
Server has started on 3000
[nodemon] restarting due to changes...
[nodemon] starting 'node src/index.js'
Server has started on 3000
```

A large white text box with a black border is overlaid on the terminal output, containing the text:

in but this is what that allows and in
case of params

At the bottom right of the video player, there are controls for volume, subtitles, closed captions, and a subscribe button.

Insomnia - GET Person by Name

GET http://localhost:3000/person/thomas

Send 200 OK TIME 19.3 ms SIZE 34 B

No Environment Cookies Body Auth Query Header Docs Preview Header Cookie Timeline

You have requested a person thomas

REST API Workshop

GET GET Person

GET GET Person by QueryString

GET GET Person by Name

Projects

Select a body type from above

here by specifying parameter so there's my query string and by params so

Subtitles/closed captions (c) (4)

SubSCRIBE

42:07 / 1:38:54

Middleware

Functions that execute serially

done calls into the next function it has
the ability to also modify

- Access to request/response objects and the next middleware function in the pipeline
- Execute any code
- Make changes to the request and the response objects
- End the request-response cycle

• Call the next middleware function in the stack

Subtitles/closed captions (c)



The screenshot shows a video player interface with a dark theme. At the top, there's a navigation bar with icons for back, forward, search, and other controls. Below it is a header bar with the title "rest-api-workshop [~/gh/theoutlander/rest-api-workshop] - .../src/index.js [rest-api-workshop]".

The main area contains a code editor with several tabs: package.json, index.js (which is the active tab), person.js, and index.html. The code in index.js is:

```
let express = require('express')
let app = express()
let personRoute = require('./routes/person')
app.use(personRoute)
app.use(express.static('public'))
const PORT = process.env.PORT || 3000
app.listen(PORT, () => console.info(`Server has started on ${PORT}`))
```

To the left of the code editor is a project structure sidebar titled "Project". It shows the directory tree:

- rest-api-workshop (~/gh/theoutlander/rest-api-workshop)
 - node_modules library root
 - public
 - src
 - routes
 - person.js
 - index.js
- package.json
- package-lock.json

Below the project structure is a section titled "External Libraries".

At the bottom of the screen is a terminal window titled "Terminal". It shows the command "npm run start-watch" being run, which starts a nodemon server. The output of the terminal is:

```
Nicks-MBP:rest-api-workshop theoutlander$ npm run start-watch
> rest-api-workshop@1.0.0 start-watch /Users/theoutlander/gh/theoutlander/rest-api-workshop
> nodemon src/index.js

[nodemon] 1.17.2
[nodemon] to restart at any time, enter 'rs'
[nodemon] watching: ***!
[nodemon] starting 'node src/index.js'
Server has started on 3000
[nodemon] restarting due to changes...
[nodemon] starting 'node src/index.js'
Server has started on 3000
[nodemon] restarting due to changes...
[nodemon] starting 'node src/index.js'
Server has started on 3000
```

A large callout box with a black background and white text is overlaid on the terminal window, containing the text:

functions are pushed onto the array
and executed in that same order so it's

At the bottom right of the screen, there are video player controls: play, volume, subtitles, closed captions, and a subscribe button.

The screenshot shows a code editor with a dark theme. On the left, the project structure is visible, showing a folder named 'rest-api-workshop' containing 'node_modules', 'public', and 'src' folders. Inside 'src', there are 'routes' and 'index.js' files. The 'index.js' file is open in the main editor area. The code is as follows:

```
let express = require('express')
let app = express()
let personRoute = require('./routes/person')

app.use((req, res, next) => {
  console.log()
})

app.use(personRoute)
app.use(express.static('public'))

const PORT = process.env.PORT || 3000
app.listen(PORT, () => console.info(`Server has started on ${PORT}`))
```

A tooltip 'callback for use()' appears over the first argument of the first 'use' call. Below the editor is a terminal window showing the output of 'nodemon' running 'node src/index.js'. It shows an error message:

```
[nodemon] starting `node src/index.js`
+ /Users/theoutlander/gh/theoutlander/rest-api-workshop/node_modules/express/lib/application.js:210
  throw new TypeError('app.use() requires a middleware function')
^

TypeError: app.use() requires a middleware function
  at Function.use (/Users/theoutlander/gh/theoutlander/rest-api-workshop/node_modules/express/lib/application.js:210:11)
  at Object.<anonymous> (/Users/theoutlander/gh/theoutlander/rest-api-workshop/src/index.js:7:5)
  at Module._compile (module.js:635:30)
  at Object.Module._extensions..js (module.js:648:10)
  at Module.load (module.js:554:32)
  at tryModuleLoad (module.js:497:12)
  at Function.Module._load (module.js:489:3)
  at Function.Module.runMain (module.js:682:10)
  at startup (bootstrap_node.js:187:16)
  at bootstrap_node.js:608:3
[nodemon] app crashed - waiting for file changes before starting...
```

A large white text overlay 'and in this case let's just console.log
a' is positioned in the lower-left area of the terminal window.

The screenshot shows a code editor with a dark theme. On the left, the project structure is visible, showing a folder named 'rest-api-workshop' containing 'node_modules', 'public', and 'src' directories. Inside 'src', there are 'routes' and 'index.js' files. The 'index.js' file is open in the main editor area, displaying the following code:

```
1 let express = require('express')
2
3 let app = express()
4
5 let personRoute = require('./routes/person')
6
7 app.use((req, res, next) => {
8   console.log(`new Date().toString() => ${req.originalUrl}`)
9
10  next()
11})
12
13 app.use(personRoute)
14 app.use(express.static('public'))
15
16 const PORT = process.env.PORT || 3000
17 app.listen(PORT, () => console.info(`Server has started on ${PORT}`))
```

In the terminal at the bottom, an error message is displayed:

```
TypeError: app.use() requires a middleware function
  at Function.use (/Users/theoutlander/gh/theoutlander/rest-api-workshop/node_modules/express/lib/application.js:210:11)
  at Object.<anonymous> (/Users/theoutlander/gh/theoutlander/rest-api-workshop/src/index.js:7:5)
  at Module._compile (module.js:635:30)
  at Object.Module._extensions..js (module.js:646:10)
  at Module.load (module.js:554:32)
  at tryModuleLoad (module.js:497:12)
  at Function.Module._load (module.js:489:3)
  at Function.Module.runMain (module.js:67)
  at startup (bootstrap_node.js:187:16)
  at bootstrap_node.js:608:3
```

A large callout box highlights the line `app.use((req, res, next) => {` in the code. The text "chain of functions that are called in that pipeline and" is overlaid on the callout box.

chain of functions that are called in
that pipeline and

Subtitles/closed captions (c)



The screenshot shows a code editor interface with a dark theme. On the left, the Project Explorer displays a file tree for a project named 'rest-api-workshop'. The 'src' folder contains 'routes' (with 'person.js' and 'index.js'), 'package.json', and 'package-lock.json'. The 'External Libraries' section is empty. In the center, a code editor window shows 'Index.js' with the following content:

```
1 let express = require('express')
2
3 let app = express()
4
5 let personRoute = require('./routes/person')
6
7 app.use((req, res, next) => {
8     console.log(` ${new Date().toString()} => ${req.originalUrl}` )
9     next()
10})
11
12 app.use(personRoute)
13 app.use(express.static('public'))
14
15 const PORT = process.env.PORT || 3000
16 app.listen(PORT, () => console.info(`Server has started on ${PORT}`))
```

there is a a line that's been out but
here which prints the the

Subtitles/closed captions (c)



Error Pages

Custom Handlers

- 404 Not Found Handler
- 500 Internal Server Error

you how to do respond with a specific file

Subtitles/closed captions (c)



The screenshot shows a code editor interface with a dark theme. On the left, the project structure is visible, including a `rest-api-workshop` folder containing `node_modules`, `public`, and `src` folders. Inside `src`, there are `routes` and `person.js`. The `index.js` file is currently selected and open in the main editor area. The terminal at the bottom shows a command-line session starting with `Thu Mar 22 2018 10:49:46 GMT-0700 (PDT) => /person/thomas`.

```
let express = require('express')
let app = express()
let personRoute = require('./routes/person')

app.use((req, res, next) => {
  console.log(`new Date().toString() => ${req.originalUrl}`)
  next()
})

app.use(personRoute)
app.use(express.static('public'))

// Handler for 404 - Resource Not Found
app.use((req, res, next) => {
  res.status(404)
})

const PORT = process.env.PORT || 3000
app.listen(PORT, () => console.info(`Server has started on ${PORT}`))
```

with a status code that says 404 and
we're going to send a

Subtitles/closed captions (c)





a separate app so I don't think so
it really comes down to what

Subtitles/closed captions (c)



rest-api-workshop [~/gh/theoutlander/rest-api-workshop] - .../public/500.html [rest-api-workshop]

The screenshot shows a code editor interface with a dark theme. On the left is a project tree titled 'rest-api-workshop'. It contains a 'node_modules' folder (library root), a 'public' folder which includes '500.html' and 'index.html', and a 'src' folder containing 'routes' (with 'person.js' and 'index.js') and 'package.json', 'package-lock.json'. The main workspace shows an open file '500.html' with the following content:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Error 500</title>
</head>
<body>
</body>
</html>
```

Terminal

```
+ Thu Mar 22 2018 10:49:46 GMT-0700 (PDT) => /person/thomas
✖ [nodemon] restarting due to changes...
✖ [nodemon] starting 'node src/index.js'
Server has started on 3000
Thu Mar 22 2018 10:52:24 GMT-0700 (PDT) => /sdjfskjdfbkjsdfb
✖ [nodemon] restarting due to changes...
✖ [nodemon] starting 'node src/index.js'
Server has started on 3000
```

call that file 500 HTML and so will
simply

Subtitles/closed captions (c)



rest-api-workshop [~/gh/theoutlander/rest-api-workshop] - .../src/index.js [rest-api-workshop]

Project rest-api-workshop src Index.js

Structure

rest-api-workshop ~/gh/theoutlander/rest-api-workshop

- node_modules library root
- public
 - 500.html
 - index.html
- src
 - routes
 - person.js
 - index.js
 - package.json
 - package-lock.json

External Libraries

package.json index.js person.js index.html

```
10 app.use(req, res, next) => {
11   console.log(`${new Date().toString()} => ${req.originalUrl}`)
12 }
13
14 app.use(personRoute)
15 app.use(express.static('public'))
16
17 // Handler for 404 - Resource Not Found
18 app.use((req, res, next) => {
19   res.status(404).send('We think you are lost!')
20 })
21
22 // Handler for Error 500
23 app.use((err, req, res, next) => {
24   console.error(err.stack)
25   res.sendFile(path.join(__dirname, '/'))
26 })
27
28
29 const PORT = process.env.PORT || 3000
30 app.listen(PORT, () => console.info(`Server has started on ${PORT}`))
```

callback for use()

Terminal

- + Thu Mar 22 2018 10:49:46 GMT-0700 (PDT) => /person/thomas
- [nodemon] restarting due to changes...
- [nodemon] starting `node src/index.js`
- Server has started on 3000
- + Thu Mar 22 2018 10:52:24 GMT-0700 (PDT) => /sdjfskjdfbkjsdfb
- [nodemon] restarting due to changes...
- [nodemon] starting `node src/index.js`
- Server has started on 3000
- [nodemon] restarting due to changes...
- [nodemon] starting `node src/index.js`
- Server has started on 3000

running from so the main index file is under source routes and

Subtitles/closed captions (c)

(4)

SUBSCRIBE

53:07 / 1:38:54

The screenshot shows a code editor interface with the following details:

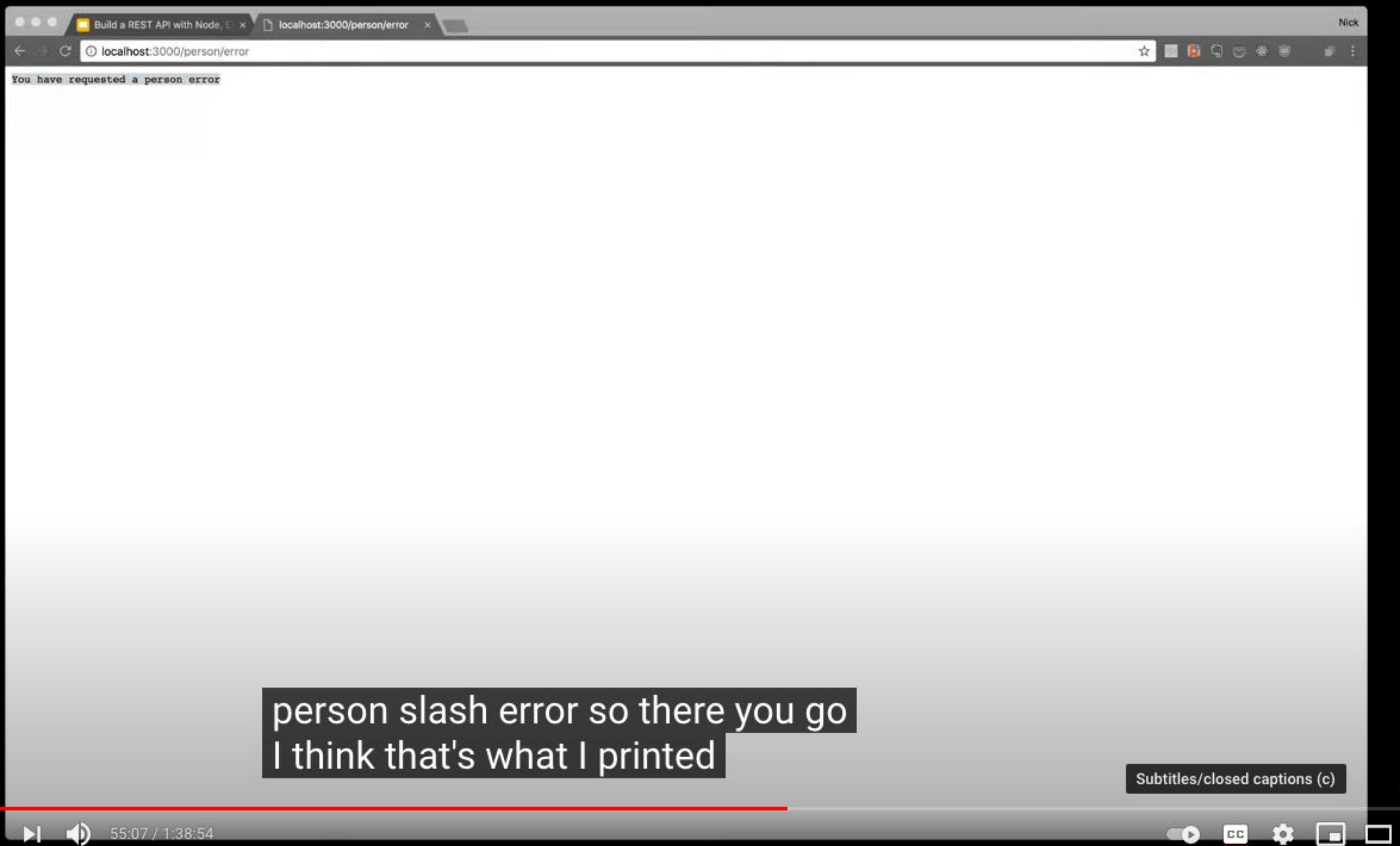
- Project Structure:** The project is named "rest-api-workshop". It contains a "src" directory which includes "routes" and "public" folders. "routes" contains "index.js" and "person.js". "public" contains "500.html" and "index.html".
- Code Editor:** The "person.js" file is open. The code defines an Express Router and handles two types of requests:
 - A query-based route: `router.get('/person', (req, res) => { if(req.query.name) { res.send(`You have requested a person \${req.query.name}`) } else { res.send('You have requested a person') } })`
 - A param-based route: `router.get('/person/:name', (req, res) => { res.send(`You have requested a person \${req.params.name}`) })`
- Terminal:** The terminal window shows the output of the nodemon command. It indicates the server has started on port 3000 and restarts automatically whenever changes are detected.
- Text Overlay:** A large white text box in the bottom right corner contains the text: "do is let's go back to the routes file and we're going to create a route here".
- Bottom Bar:** Includes standard video player controls like play, volume, and subtitles, along with the current timestamp (54:07 / 1:38:54).

do is let's go back to the routes file
and we're going to create a route here

Subtitles/closed captions (c)



SUBSCRIBE



person slash error so there you go
I think that's what I printed

Subtitles/closed captions (c)





Error: ENOENT: no such file or directory, stat '/Users/theoutlander/gh/theoutlander/rest-api-workshop/src/../public/500.html'

I need slash public so I find one at
HTML so let's retry that so going

Subtitles/closed captions (c)



Mongoose

For MongoDB

- Install dependency
- Reference Mongoose
- CRUD API

Mongoose which is an ORM or an OD M
it's basically an object data mapper

Subtitles/closed captions (c)



Mongoose

For MongoDB

- Install dependency
- Reference Mongoose
- CRUD API

we're going to focus on creating a kraut
api

Subtitles/closed captions (c)



A screenshot of a code editor (likely VS Code) showing a project structure and a script file. The project structure on the left includes a `rest-api-workshop` folder with `node_modules`, `public` (containing `500.html` and `index.html`), and `src` (containing `routes` with `person.js` and `index.js`, and files `package.json` and `package-lock.json`). The current file open is `index.js` at the top of the code editor. The code in `index.js` is a Node.js application setup:

```
4
5     let personRoute = require('./routes/person')
6
7     let path = require('path')
8
9     app.use((req, res, next) => {
10         console.log(`new Date().toString() => ${req.originalUrl}`)
11         next()
12     })
13
14     app.use(personRoute)
15     app.use(express.static('public'))
16
17     // Handler for 404 - Resource Not Found
18     app.use((req, res, next) => {
19         res.status(404).send('We think you are lost!')
20     })
21
22     // Handler for Error 500
23     app.use((err, req, res, next) => {
24         console.error(err.stack)
25         res.sendFile(path.join(__dirname, '../public/500.html'))
26     })
27
28 const PORT = process.env.PORT || 3000
path
```

Terminal

```
+ Nicks-MBP:rest-api-workshop theoutlander$ npm i mongoose
npm WARN rest-api-workshop@1.0.0 No description
npm WARN rest-api-workshop@1.0.0 No repository field.

+ mongoose@5.0.11
added 17 packages from 12 contributors in 5.424s
Nicks-MBP:rest-api-workshop theoutlander$
```

mongoose alright and what we're going to do is create a folder under source

Subtitles/closed captions (c)



The screenshot shows a code editor interface with a dark theme. On the left, there's a sidebar titled "Project" showing the directory structure of a project named "rest-api-workshop". Inside "src", there are "models" and "routes" folders. "models" contains "customer.model.js". "routes" contains "person.js" and "index.js". Below these are "package.json" and "package-lock.json". A "public" folder contains "500.html" and "index.html". At the bottom of the sidebar is an "External Libraries" section. The main workspace has tabs for "package.json", "index.js", "customer.model.js", "person.js", and "Index.html". The "customer.model.js" tab is active, displaying the following code:

```
let mongoose = require('mongoose');
```

mongoose

Terminal

```
Nicks-MBP:rest-api-workshop theoutlander$ npm i mongoose
npm WARN rest-api-workshop@1.0.0 No description
npm WARN rest-api-workshop@1.0.0 No repository field.

+ mongoose@5.0.11
added 17 packages from 12 contributors in 5.424s
Nicks-MBP:rest-api-workshop theoutlander$
```

reference Mongoose you already installed
it so that should be available

Subtitles/closed captions (c)



The screenshot shows a code editor interface with a dark theme. On the left, the project structure is displayed:

- rest-api-workshop (highlighted)
- node_modules
- public
 - 500.html
 - index.html
- src
 - models
 - customer.model.js
 - routes
 - person.js
 - index.js
- package.json
- package-lock.json

The right pane shows the content of `customer.model.js`:

```
1 let mongoose = require('mongoose')
2
3 const server = 'ds221609.mlab.com:21609'
4 const database = 'rest-api-workshop'
5 const user = 'theoutlander'
6 const password = 'fZsMGZXOMxBFCTgkBwgFtEvwD7ML'
```

Below the code editor is a terminal window showing the output of an `npm i mongoose` command:

```
Nicks-MBP:rest-api-workshop theoutlander$ npm i mongoose
npm WARN rest-api-workshop@1.0.0 No description
npm WARN rest-api-workshop@1.0.0 No repository field.

+ mongoose@5.8.11
added 17 packages from 12 contributors in 5.424s
Nicks-MBP:rest-api-workshop theoutlander$
```

A large black rectangular box highlights the text "option to see all these attributes and the connection string so the next thing". In the bottom right corner, there are video player controls and subtitles/closed captions settings.

option to see all these attributes and
the connection string so the next thing

Subtitles/closed captions (c)

The screenshot shows a code editor interface with a dark theme. On the left, the project structure is displayed:

- rest-api-workshop (library root)
- public
 - 500.html
 - index.html
- src
 - models
 - customer.model.js
 - routes
 - person.js
 - index.js
- package.json
- package-lock.json

The right pane shows the content of `customer.model.js`:

```
let mongoose = require('mongoose')

const server = 'ds221609.mlab.com:21609'
const database = 'rest-api-workshop'
const user = 'theoutlander'
const password = 'fZsMGZXOMxBFCTgkBwgFtEvwD7ML'

mongoose.connect(`mongodb://${user}:${password}@${server}/${database}`)
```

Terminal

```
Nicks-MBP:rest-api-workshop theoutlander$ npm i mongoose
npm WARN rest-api-workshop@1.0.0 No description
npm WARN rest-api-workshop@1.0.0 No repository field.

+ mongoose@5.0.11
added 17 packages from 12 contributors in 5.424s
Nicks-MBP:rest-api-workshop theoutlander$
```

all right so that should generate the
the URI for

Subtitles/closed captions (c)

1: Favorites

1:02:07 / 1:38:54

A screenshot of a video player interface displaying a code editor and a terminal window. The code editor shows a file named `customer.model.js` with the following content:

```
let mongoose = require('mongoose')

const server = 'ds221609.mlab.com:21609'
const database = 'rest-api-workshop'
const user = 'theoutlander'
const password = 'fZsMGZXQMX8FCTgkBwgFtEvwD7ML'

mongoose.connect(`mongodb://${user}:${password}@${server}/${database}`)

let CustomerSchema = new mongoose.Schema({
  name: String,
})
```

The terminal window below shows the command `npm i mongoose` being run, with the output indicating that the package was installed successfully:

```
Nicks-MBP:rest-api-workshop theoutlander$ npm i mongoose
npm WARN rest-api-workshop@1.0.0 No description
npm WARN rest-api-workshop@1.0.0 No repository field.

+ mongoose@5.0.11
added 17 packages from 12 contributors in 5.424s
Nicks-MBP:rest-api-workshop theoutlander$
```

A large text overlay in the bottom left corner reads: "tell it that the type is string and then the next thing is let's create".

Subtitles/closed captions (c)



The screenshot shows a code editor with a dark theme. On the left is a project tree for 'rest-api-workshop' containing 'src' and 'models' folders. Inside 'models' is 'customer.model.js'. The main pane displays the following code:

```
let mongoose = require('mongoose')
const server = 'ds221609.mlab.com:21609'
const database = 'rest-api-workshop'
const user = 'theoutlander'
const password = 'fZsMGZXQMX8FCTgkBwgFtEvwD7ML'

mongoose.connect(`mongodb://${user}:${password}@${server}/${database}`)

let CustomerSchema = new mongoose.Schema({
  name: String,
  email: {
    type: String,
    require: true,
    unique: true
  }
})
```

Terminal

```
Nicks-MBP:rest-api-workshop theoutlander$ npm i mongoose
npm WARN rest-api-workshop@1.0.0 No description
npm WARN rest-api-workshop@1.0.0 No repository field.

+ mongoose@5.8.11
added 17 packages from 12 contributors in 5.424s
Nicks-MBP:rest-api-workshop theoutlander$
```

customer schema we're going to create a model and export it at the same time so

Subtitles/closed captions (c)



A screenshot of a video player interface displaying a presentation slide. The slide content is overlaid on a background of a code editor and terminal window.

The code editor shows `customer.model.js` with the following content:

```
1 let mongoose = require('mongoose')
2
3 const server = 'ds221609.mlab.com:21609'
4 const database = 'rest-api-workshop'
5 const user = 'theoutlander'
6 const password = 'fZsMGZXQMX8FCTgkBwgFtEvwD7ML'
7
8 mongoose.connect(`mongodb://${user}:${password}@${server}/${database}`)
9
10 let CustomerSchema = new mongoose.Schema({
11   name: String,
12   email: {
13     type: String,
14     required: true,
15     unique: true
16   }
17 })
18
19 module.exports = mongoose.model('Customer', CustomerSchema)
20
```

The terminal window below shows the output of running `npm i mongoose`:

```
Nicks-MBP:rest-api-workshop theoutlander$ npm i mongoose
npm WARN rest-api-workshop@1.0.0 No description
npm WARN rest-api-workshop@1.0.0 No repository field.

+ mongoose@5.0.11
added 17 packages from 12 contributors in 5.424s
Nicks-MBP:rest-api-workshop theoutlander$
```

The main content of the slide is a large text box containing the following text:

needs to be any and it must be present

At the bottom right of the slide, there are controls for subtitles/closed captions (c) and a subscribe button.

The screenshot shows a code editor interface with a dark theme. On the left, the Project Explorer displays the file structure of a 'rest-api-workshop' project. The 'routes' folder contains 'customer.js', 'index.js', 'person.js', and 'package.json'. The 'models' folder contains 'customer.model.js'. The 'src' folder also contains '500.html' and 'index.html'. The 'public' folder contains '500.html' and 'index.html'. The 'node_modules' folder is listed as 'library root'. The 'External Libraries' section is empty. On the right, the main editor pane shows the content of 'customer.js': `let CustomerModel = require()`. Below the editor is a Terminal window showing the command `npm i mongoose` being run, with output indicating the package was installed successfully. The bottom right corner features a video player interface with controls for play, volume, and subtitles.

```
rest-api-workshop [-/gh/theoutlander/rest-api-workshop] - .../src/routes/customer.js [rest-api-workshop]
```

rest-api-workshop src routes customer.js

Project rest-api-workshop -/gh/theoutlander/rest-api-workshop node_modules library root public 500.html index.html src models customer.model.js routes customer.js person.js index.js package.json package-lock.json External Libraries

customer.js

```
let CustomerModel = require()
```

package.json Index.js customer.model.js customer.js person.js index.html

Terminal

```
Nicks-MBP:rest-api-workshop theoutlander$ npm i mongoose
npm WARN rest-api-workshop@1.0.0 No description
npm WARN rest-api-workshop@1.0.0 No repository field.

+ mongoose@5.8.11
added 17 packages from 12 contributors in 5.424s
Nicks-MBP:rest-api-workshop theoutlander$
```

referencing the customer model so let's do let customer model equals require and

Subtitles/closed captions (c) (4)

Subtitles/closed captions (c) SUBSCRIBE

1:06:07 / 1:38:54

The screenshot shows a video player interface with a dark theme. At the top, a browser tab displays the URL: rest-api-workshop [~/gh/theoutlander/rest-api-workshop] - .../src/routes/customer.js [rest-api-workshop]. Below the browser is a code editor window titled 'rest-api-workshop' with the file 'routes/customer.js' open. The code editor shows the following JavaScript code:

```
let CustomerModel = require('../models/customer.model')
let express = require('express')
let router = express.Router()

// Create a new customer
router.post()
```

The code editor's sidebar shows the project structure:

- Project
- rest-api-workshop (~/gh/theoutlander/rest-api-workshop)
 - node_modules library root
 - public
 - 500.html
 - index.html
 - src
 - models
 - customer.model.js
 - routes
 - customer.js
 - person.js
 - index.js
- External Libraries

Below the code editor is a terminal window titled 'Terminal' showing the command line output:

```
Nicks-MBP:rest-api-workshop theoutlander$ npm i mongoose
npm WARN rest-api-workshop@1.0.0 No description
npm WARN rest-api-workshop@1.0.0 No repository field.

+ mongoose@5.0.11
added 17 packages from 12 contributors in 5.424s
Nicks-MBP:rest-api-workshop theoutlander$
```

At the bottom of the screen, there is a large white text overlay that reads: "the verb itself is available as a method in". To the right of this text are several video player controls: a play button, a volume icon, a closed captioning icon labeled "Subtitles/closed captions (c)", a subscribe button with a bell icon labeled "SUBSCRIBE", and other standard video controls.

At the very bottom of the screen, there is a red progress bar indicating the video's duration: 1:07:07 / 1:38:54. Below the progress bar, there is a message: "Argument types do not match parameters".

The screenshot shows a code editor interface with the following details:

- Project Structure:** The project is named "rest-api-workshop". It contains a "public" folder with "500.html" and "index.html". The "src" folder contains "models" (with "customer.model.js") and "routes" (with "customer.js", "person.js", and "index.js"). There are also "package.json" and "package-lock.json" files.
- Code Editor:** The main editor window displays the "index.js" file. The code sets up an Express application, requiring "express", "path", and "mongoose" (which is currently not installed). It defines routes for "customer" and "person" and handles 404 and 500 errors.

```
let express = require('express')
let app = express()
let personRoute = require('./routes/person')
let path = require('path')

app.use((req, res, next) => {
  console.log(`new Date().toString() => ${req.originalUrl}`)
  next()
})

app.use(personRoute)
app.use(express.static('public'))

// Handler for 404 - Resource Not Found
app.use((req, res, next) => {
  res.status(404).send('We think you are lost!')
})

// Handler for Error 500
app.use((err, req, res, next) => {
  console.error(err.stack)
  res.sendFile(path.join(__dirname, '../public/500.html'))
})
```

Terminal

```
Nicks-MBP:rest-api-workshop theoutlander$ npm i mongoose
npm WARN rest-api-workshop@1.0.0 No description
npm WARN rest-api-workshop@1.0.0 No repository field.

+ mongoose@5.0.11
added 17 packages from 12 contributors in 5.424s
Nicks-MBP:rest-api-workshop theoutlander$
```

is a package we need to install for it
called body parser and so let's

Subtitles/closed captions (c)



The screenshot shows a code editor interface with a dark theme. On the left, the project structure is displayed:

- Project: rest-api-workshop (~/gh/theoutlander/rest-api-workshop)
- node_modules (library root)
- public:
 - 500.html
 - index.html
- src:
 - models:
 - customer.model.js
 - routes:
 - customer.js
 - person.js
 - index.js
 - package.json
 - package-lock.json
- External Libraries

The current file being edited is `index.js`, which contains the following code:

```
1 let express = require('express')
2
3 let app = express()
4
5 let personRoute = require('./routes/person')
6
7 let path = require('path')
8
9 let bodyParser = require('body-parser')
10
11 app.use((req, res, next) => {
12   console.log(` ${new Date().toString()} => ${req.originalUrl}` )
13   next()
14 })
15
16 app.use(personRoute)
17 app.use(express.static('public'))
18
19 // Handler for 404 - Resource Not Found
20 app.use((req, res, next) => {
21   res.status(404).send('We think you are lost!')
22 })
23
24 // Handler for Error 500
25 app.use((err, req, res, next) => {
```

Below the code editor is a terminal window showing the output of npm installations:

```
+ Nicks-MBP:rest-api-workshop theoutlander$ npm i mongoose
npm WARN rest-api-workshop@1.0.0 No description
npm WARN rest-api-workshop@1.0.0 No repository field.

+ mongoose@5.0.11
added 17 packages from 12 contributors in 5.424s
Nicks-MBP:rest-api-workshop theoutlander$ npm i body-parser
npm WARN rest-api-workshop@1.0.0 No description
npm WARN rest-api-workshop@1.0.0 No repository field.

+ body-parser@1.18.2
updated 1 package in 2.911s
Nicks-MBP:rest-api-workshop theoutlander$
```

A large white text box is overlaid on the bottom right of the terminal area, containing the following text:

here so we're going to reference body parser and we're going to

At the bottom of the screen, there are standard video player controls and a progress bar indicating the video is at 1:09:07 / 1:38:54.

Subtitles/closed captions (c)



```
let CustomerModel = require('../models/customer.model')
let express = require('express')
let router = express.Router()

// Create a new customer
// POST localhost:3000/customer
router.post('/customer', (req, res) => {
  // req.body
})

callback for post()
```

```
Nicks-MBP:rest-api-workshop theoutlander$ npm i mongoose
npm WARN rest-api-workshop@1.0.0 No description
npm WARN rest-api-workshop@1.0.0 No repository field.

+ mongoose@5.0.11
added 17 packages from 12 contributors in 5.424s
Nicks-MBP:rest-api-workshop theoutlander$ npm i body-parser
npm WARN rest-api-workshop@1.0.0 No description
npm WARN rest-api-workshop@1.0.0 No repository field.

+ body-parser@1.18.2
updated 1 package in 2.911s
Nicks-MBP:rest-api-workshop theoutlander$
```

would have to look at the raw data and then the content type and then you

A screenshot of a video player interface displaying a code editor and a terminal window.

The code editor shows the file `customer.js` from a project named `rest-api-workshop`. The code defines a `Router` for handling customer requests:

```
let CustomerModel = require('../models/customer.model')
let express = require('express')
let router = express.Router()

// Create a new customer
// POST localhost:3000/customer
router.post('/customer', (req, res) => {
  if(!req.body) {
    return res.status(400).s
  }
})
```

The terminal window shows the output of running `npm i` to install dependencies:

```
Nicks-MBP:rest-api-workshop theoutlander$ npm i mongoose
npm WARN rest-api-workshop@1.0.0 No description
npm WARN rest-api-workshop@1.0.0 No repository field.

+ mongoose@5.0.11
added 17 packages from 12 contributors in 5.424s
Nicks-MBP:rest-api-workshop theoutlander$ npm i body-parser
npm WARN rest-api-workshop@1.0.0 No description
npm WARN rest-api-workshop@1.0.0 No repository field.

+ body-parser@1.18.2
updated 1 package in 2.911s
Nicks-MBP:rest-api-workshop theoutlander$
```

A large callout box highlights the line `return res.status(400).s` with the text: "response with a status code of 400 which means it's a bad request and send a".

Subtitles/closed captions (c)

A screenshot of a video player interface displaying a code editor and a terminal window.

The code editor shows the file `customer.js` with the following content:

```
let CustomerModel = require('../models/customer.model')
let express = require('express')
let router = express.Router()

// Create a new customer
// POST localhost:3000/customer
router.post('/customer', (req, res) => {
  if(!req.body) {
    return res.status(400).send('Request body is missing')
  }

  let model = new CustomerModel(req.body)
})
```

The terminal window shows the output of the command `npm i mongoose`:

```
Nicks-MBP:rest-api-workshop theoutlander$ npm i mongoose
npm WARN rest-api-workshop@1.0.0 No description
npm WARN rest-api-workshop@1.0.0 No repository field.

+ mongoose@5.8.11
added 17 packages from 12 contributors in 5.424s
Nicks-MBP:rest-api-workshop theoutlander$ npm i body-parser
npm WARN rest-api-workshop@1.0.0 No description
npm WARN rest-api-workshop@1.0.0 No repository field.

+ body-parser@1.18.2
updated 1 package in 2.911s
Nicks-MBP:rest-api-workshop theoutlander$
```

A large white text box with a black border contains the transcription of the spoken words:

request so we're gonna pass in whatever
oops was a part

At the bottom right of the video player, there are controls for subtitles/closed captions (c), a subscribe button, and other video controls.

The screenshot shows a code editor interface with a dark theme. On the left, the project structure is displayed:

- rest-api-workshop
- src
- routes
- customer.js

The file `customer.js` is open in the editor. The code is as follows:

```
let CustomerModel = require('../models/customer.model')
let express = require('express')
let router = express.Router()

// Create a new customer
// POST localhost:3000/customer
router.post('/customer', (req, res) => {
  if(!req.body) {
    return res.status(400).send('Request body is missing')
  }

  // let user = {
  //   name: 'firstname lastname',
  //   email: 'email@gmail.com'
  // }

  let model = new CustomerModel(req.body)
  model.save()
})
```

A yellow dot marker is placed on the line `model.save()`. Below the code editor, a tooltip says "callback for post() > model".

In the bottom-left corner, the terminal window shows the following npm commands and their output:

```
Nicks-MBP:rest-api-workshop theoutlander$ npm i mongoose
npm WARN rest-api-workshop@1.0.0 No description
npm WARN rest-api-workshop@1.0.0 No repository field.

+ mongoose@5.8.11
added 17 packages from 12 contributors in 5.424s
Nicks-MBP:rest-api-workshop theoutlander$ npm i body-parser
npm WARN rest-api-workshop@1.0.0 No description
npm WARN rest-api-workshop@1.0.0 No repository field.

+ body-parser@1.18.2
updated 1 package in 2.911s
Nicks-MBP:rest-api-workshop theoutlander$
```

A large black callout box is overlaid on the bottom right, containing the text:

the request of body object and validated
via the customer model and

At the very bottom of the screen, there is a navigation bar with icons for play, search, and other media controls.

The screenshot shows a code editor interface with a dark theme. On the left, the project structure is displayed:

- rest-api-workshop
- src
 - models (containing customer.model.js)
 - routes (containing customer.js, person.js, index.js, package.json, package-lock.json)
- public (containing 500.html, index.html)
- node_modules (library root)
- External Libraries

The right pane shows the content of `customer.js`:

```
let CustomerModel = require('../models/customer.model')
let express = require('express')
let router = express.Router()

// Create a new customer
// POST localhost:3000/customer
router.post('/customer', (req, res) => {
  if(!req.body) {
    return res.status(400).send('Request body is missing')
  }

  // let user = {
  //   name: 'firstname lastname',
  //   email: 'email@gmail.com'
  // }

  let model = new CustomerModel(req.body)
  model.save()
    .then(doc => {
      if(!doc || doc.length === 0) {
        return res.status(500).send(doc)
      }

      res.status(201)
    })
})
```

Below the code editor, the terminal window shows the following npm commands and their output:

```
Nicks-MBP:rest-api-workshop theoutlander$ npm i mongoose
npm WARN rest-api-workshop@1.0.0 No description
npm WARN rest-api-workshop@1.0.0 No repository field.

+ mongoose@5.8.11
added 17 packages from 12 contributors in 5.424s
Nicks-MBP:rest-api-workshop theoutlander$ npm i body-parser
npm WARN rest-api-workshop@1.0.0 No description
npm WARN rest-api-workshop@1.0.0 No repository field.

+ body-parser@1.18.2
updated 1 package in 2.911s
Nicks-MBP:rest-api-workshop theoutlander$
```

A large, semi-transparent text box is overlaid on the bottom right of the screen, containing the following text:

respond with the status code of 201
which

At the very bottom of the screen, there is a red progress bar indicating the video's duration.

Subtitles/closed captions (c)



A screenshot of a video player interface displaying a code editor and a terminal window.

The code editor shows the file `customer.js` from a project named `rest-api-workshop`. The code handles a POST request to '/customer'. It checks if the request body is missing, returns a 400 status if so. It then creates a new `CustomerModel` instance from the body, saves it, and returns a 201 status with the saved document. If there's an error during saving, it catches it and returns a 500 status with the error object.

```
7 router.post('/customer', (req, res) => {
8   if(!req.body) {
9     return res.status(400).send('Request body is missing')
10  }
11
12  // let user = {
13  //   name: 'firstname lastname',
14  //   email: 'email@gmail.com'
15  // }
16
17  let model = new CustomerModel(req.body)
18  model.save()
19    .then(doc => {
20      if(!doc || doc.length === 0) {
21        return res.status(500).send(doc)
22      }
23
24      res.status(201).send(doc)
25    })
26    .catch(err => {
27      res.status(500).json(err)
28    })
29})
```

The terminal window shows the command `npm i mongoose` being run, followed by the output of the installation:

```
Nicks-MBP:rest-api-workshop theoutlander$ npm i mongoose
+ mongoose@5.0.11
added 17 packages from 12 contributors in 5.424s
Nicks-MBP:rest-api-workshop theoutlander$ npm i body-parser
+ body-parser@1.18.2
updated 1 package in 2.911s
Nicks-MBP:rest-api-workshop theoutlander$
```

A large callout box in the bottom right corner contains the text: "to take in the error object and and transform it and send it".

to take in the error object and and
transform it and send it

Subtitles/closed captions (c)



The screenshot shows a video player interface with a dark theme. The main area is a code editor displaying a file named `customer.js` from a project titled "rest-api-workshop". The code implements a POST endpoint for creating a customer. The terminal below shows the command to install the `body-parser` package and the command to start the application with `npm run start-watch`. A large white text overlay in the bottom right corner reads: "an error there and guess what we forgot to add this to the".

```
Project: rest-api-workshop [~/gh/theoutlander/rest-api-workshop] - .../src/routes/customer.js [rest-api-workshop]
  1: Project
  2: Structure
  3: rest-api-workshop ~/gh/theoutlander/rest-api-workshop
    node_modules library root
    public
      500.html
      index.html
    src
      models
        customer.model.js
      routes
        customer.js
        person.js
        index.js
        package.json
        package-lock.json
  4: package.json
  5: index.js
  6: customer.model.js
  7: customer.js
  8: person.js
  9: index.html

  7: router.post('/customer', (req, res) => {
  8:   if(!req.body) {
  9:     return res.status(400).send('Request body is missing')
 10:   }
 11:   // let user = {
 12:   //   name: 'firstname lastname',
 13:   //   email: 'email@gmail.com'
 14:   // }
 15:   let model = new CustomerModel(req.body)
 16:   model.save()
 17:   .then(doc => {
 18:     if(!doc || doc.length === 0) {
 19:       return res.status(500).send(doc)
 20:     }
 21:     res.status(201).send(doc)
 22:   })
 23:   .catch(err => {
 24:     res.status(500).json(err)
 25:   })
 26: })
 27: })
 28: })
 29: }

callback for post()
```

```
Nicks-MBP:rest-api-workshop theoutlander$ npm i body-parser
+ body-parser@1.18.2
updated 1 package in 2.911s
Nicks-MBP:rest-api-workshop theoutlander$ npm run start-watch

> rest-api-workshop@1.0.0 start-watch /Users/theoutlander/gh/theoutlander/rest-api-workshop
> nodemon src/index.js

[nodemon] 1.17.2
[nodemon] to restart at any time, enter `rs`
[nodemon] watching: ***!
[nodemon] starting 'node src/index.js'
Server has started on 3000
Thu Mar 22 2018 11:18:48 GMT-0700 (PDT) => /customer {}
```

an error there and guess what we forgot to add this to the

Subtitles/closed captions (c) (A)
SUBSCRIBE

The screenshot shows a video player interface with a dark theme. The main area displays a code editor for a Node.js project named 'rest-api-workshop'. The code in `index.js` is as follows:

```
let express = require('express')
let app = express()
let personRoute = require('./routes/person')
let customerRoute = require('./routes/customer')
let path = require('path')
let bodyParser = require('body-parser')

app.use(bodyParser.json())

app.use((req, res, next) => {
  console.log(` ${new Date().toString()} => ${req.originalUrl}`, req.body)
  next()
})

app.use(personRoute)
app.use(customerRoute)
app.use(express.static('public'))

// Handler for 404 – Resource Not Found
app.use((req, res, next) => {
  res.status(404).send('We think you are lost!')
  next()
})

// Handler for Error 500
app.use((err, req, res, next) => {
  callbackForUse()
})
```

The terminal window below shows the output of running `node src/index.js`. It starts with '[nodemon] starting 'node src/index.js'', followed by an error message: 'throw new TypeError('app.use() requires a middleware function')'. This is followed by a detailed stack trace of the error, ending with '[nodemon] app crashed - waiting for file changes before starting...'. A large, semi-transparent text overlay in the bottom right corner reads 'console you can type in RS which tells it to restart and'.

console you can type in RS which tells
it to restart and

Subtitles/closed captions (c)



Insomnia - POST Customer

POST localhost:3000/customer

Send 404 Not Found TIME 21 ms SIZE 22 B

No Environment Cookies Body Auth Query Header Docs Preview Header Cookie Timeline

We think you are lost!

REST API Workshop

POST POST Customer

GET GET Person

GET GET Person by QueryString

GET GET Person by Params

Projects

Select a body type from above

and it and the 404 handler the the
resource not found handler kicked in and

Subtitles/closed captions (c)

INSOMNIA SUBSCRIBE

▶ ▶ 🔍 1:18:07 / 1:38:54

▶ 🔍 CC ⚙️ ⚡ [] []

Insomnia

POST → localhost:3000/customer

Send

01 Created

ME 116 ms

ZE 94 B

6

No Environment - Cookies

JSON

Auth -

Q1973

Header 3

Docs

Preview ▾

Header 9

Cookie

Timeline

```
1 = {  
2   "name": "Thomas Anderson",  
3   "email": "thomas@gmail.com"  
4 }
```

```
1 - {  
2   "_id": "5ab3f442f3d76b4ae4cb433b",  
3   "name": "Thomas Anderson",  
4   "email": "thomas@gmail.com",  
5   "__v": 0  
6 }
```

REST API Workshop

POST POST Customer

GET GET Person

Projects

gmail.com and essentially I can when I click send here it

Subtitles/closed captions (c)





mLab (1)
rest-api-workshop
Collections (2)
System
customers
Functions
Users

db.getCollection('customers').find({}) db.getCollection('customers').find({})

mLab ds221609.mlab.com:21609 rest-api-workshop

1 db.getCollection('customers').find({})

customers 0.089 sec.

Key	Value	Type
1 ObjectId("5ab3f442f3d76b4ae4cb433b")	{ 4 fields }	Object
_id	ObjectId("5ab3f442f3d76b4ae4cb433b")	ObjectId
name	Thomas Anderson	String
email	thomas@gmail.com	String
__v	0	Int32

remember a few things at this point but
that is okay you know a little

Subtitles/closed captions (c)



rest-api-workshop [~/gh/theoutlander/rest-api-workshop] - .../src/routes/customer.js [rest-api-workshop]

rest-api-workshop src routes customer.js

Project rest-api-workshop ~/gh/theoutlander/rest-api-workshop node_modules library root public 500.html index.html src models customer.model.js routes customer.js person.js index.js package.json package-lock.json External Libraries

package.json index.js customer.model.js customer.js person.js index.html

```
11 if(!req.body.email) {
12     // ...
13 }
14 // let user = {
15 //     name: 'firstname lastname',
16 //     email: 'email@gmail.com'
17 // }
18
19 let model = new CustomerModel(req.body)
model.save()
    .then(doc => {
        if(!doc || doc.length === 0) {
            return res.status(500).send(doc)
        }
        res.status(201).send(doc)
    })
    .catch(err => {
        res.status(500).json(err)
    })
}
module.exports = router
```

Terminal

```
+ TypeError: app.use() requires a middleware function
  at Function.use (/Users/theoutlander/gh/theoutlander/rest-api-workshop/node_modules/express/lib/application.js:210:11)
x at Object.<anonymous> (/Users/theoutlander/gh/theoutlander/rest-api-workshop/src/index.js:16:5)
  at Module._compile (module.js:635:30)
  at Object.Module._extensions..js (module.js:646:18)
  at Module.load (module.js:554:32)
  at tryModuleLoad (module.js:497:12)
  at Function.Module._load (module.js:489:3)
  at Function.Module.runMain (module.js:676:10)
  at startup (bootstrap_node.js:187:16)
  at bootstrap_node.js:608:3
[nodemon] app crashed - waiting for file changes...
[nodemon] restarting due to changes...
[nodemon] starting `node src/index.js`
Server has started on 3000
Thu Mar 22 2018 11:21:00 GMT-0700 (PDT) => /customer {}
Thu Mar 22 2018 11:21:54 GMT-0700 (PDT) => /customer { name: 'Thomas Anderson', email: 'thomas@gmail.com' }
```

going to next add the ability to retrieve

Subtitles/closed captions (c) (A) SUBSCRIBE

1:21:07 / 1:38:54

rest-api-workshop [~/gh/theoutlander/rest-api-workshop] - .../src/routes/customer.js [rest-api-workshop]

The screenshot shows a code editor with a dark theme. On the left is a project tree for 'rest-api-workshop'. It includes a 'node_modules' folder, a 'public' folder with '500.html' and 'index.html', a 'src' folder containing 'models' (with 'customer.model.js') and 'routes' (with 'customer.js, person.js, index.js'). There are also 'package.json' and 'package-lock.json' files. The right side shows the content of 'customer.js'. The code uses Express.js to handle POST requests to '/customer'. It checks if the request body is missing or if it lacks an 'email' field. It then creates a new 'CustomerModel' instance with the provided data, saves it, and returns a 201 status with the saved document. If there's an error or no document is found, it returns a 500 status with an error JSON. The code then handles GET requests to '/customer', finding one customer by ID. Finally, it exports the router.

```
let CustomerModel = require('../models/customer.model')
let express = require('express')
let router = express.Router()

// Create a new customer
// POST localhost:3000/customer
router.post('/customer', (req, res) => {
  if(!req.body) {
    return res.status(400).send('Request body is missing')
  }

  if(!req.body.email) {
    // ...
  }

  // let user = {
  //   name: 'firstname lastname',
  //   email: 'email@gmail.com'
  // }

  let model = new CustomerModel(req.body)
  model.save()
    .then(doc => {
      if(!doc || doc.length === 0) {
        return res.status(500).send(doc)
      }

      res.status(201).send(doc)
    })
    .catch(err => {
      res.status(500).json(err)
    })
  }

  router.get('/customer', (req, res) => {
    CustomerModel.findOne({
      // ...
    })
  })

  module.exports = router
})
```

but anyway in this case we're gonna use
find one and find one takes an object

Subtitles/closed captions (c) (4)

callback for get() 1:22:07 / 1:38:54

```
 7   router.post('/customer', (req, res) => {
 8     if(!req.body) {
 9       return res.status(400).send('Request body is missing')
10    }
11
12    if(!req.body.email) {
13      // ...
14    }
15
16    // let user = {
17    //   name: 'firstname lastname',
18    //   email: 'email@gmail.com'
19    //}
20
21    let model = new CustomerModel(req.body)
22    model.save()
23    .then(doc => {
24      if(!doc || doc.length === 0) {
25        return res.status(500).send(doc)
26      }
27
28      res.status(201).send(doc)
29    })
29    .catch(err => {
30      res.status(500).json(err)
31    })
32  })
33
34
35  router.get('/customer', (req, res) => {
36    if(!req.query.email) {
37      return res.status(400).send('Missing URL parameter: email')
38    }
39
40    CustomerModel.findOne({
41      email: req.query.email
42    })
43    .then(doc => {
44      |
45    })
46  })
47
```

should also have our document available
on on the den method that gets called

Subtitles/closed captions (c)



rest-api-workshop [~/gh/theoutlander/rest-api-workshop] - .../src/routes/customer.js [rest-api-workshop]

The screenshot shows a code editor with a dark theme. On the left is a project tree for 'rest-api-workshop'. It includes a 'node_modules' folder, a 'public' folder with '500.html' and 'index.html', a 'src' folder containing 'models' (with 'customer.model.js') and 'routes' (with 'customer.js', 'person.js', 'index.js', 'package.json', and 'package-lock.json'). The 'customer.js' file in the 'routes' folder is currently selected and shown in the main editor area. The code in 'customer.js' handles POST and GET requests for customers, utilizing a 'CustomerModel' to interact with a database.

```
14      }
15
16      // let user = {
17      //   name: 'firstname lastname',
18      //   email: 'email@gmail.com'
19      // }
20
21      let model = new CustomerModel(req.body)
22      model.save()
23        .then(doc => {
24          if(!doc || doc.length === 0) {
25            return res.status(500).send(doc)
26          }
27
28          res.status(201).send(doc)
29        })
30        .catch(err => {
31          res.status(500).json(err)
32        })
33      })
34
35      router.get('/customer', (req, res) => {
36        if(!req.query.email) {
37          return res.status(400).send('Missing URL parameter: email')
38        }
39
40        CustomerModel.findOne({
41          email: req.query.email
42        })
43        .then(doc => {
44          res.json(doc)
45        })
46        .catch(err => {
47          res.status(500).json(err)
48        })
49      })
50
51      router.put('/cusomter', (req))

```

I could I could use some of this reuse
some of that so we have request response

Subtitles/closed captions (c)

(4)

SUBSCRIBE

1: Favorites

1: Project

1: Structure

1: TODO 1: 1:24:07 / 1:38:54

1: CC 1: 1:30 LF: UTF-8

rest-api-workshop [~/gh/theoutlander/rest-api-workshop] - .../src/routes/customer.js [rest-api-workshop]

Project rest-api-workshop ~/gh/theoutlander/rest-api-workshop

src models customer.model.js routes customer.js person.js index.html

customer.js

```
38     .catch(err => {
39         res.status(500).json(err)
40     })
41 }
42
43 router.get('/customer', (req, res) => {
44     if(!req.query.email) {
45         return res.status(400).send('Missing URL parameter: email')
46     }
47     CustomerModel.findOne({
48         email: req.query.email
49     })
50         .then(doc => {
51             res.json(doc)
52         })
53         .catch(err => {
54             res.stats(500).json(err)
55         })
56 }
57
58 router.put('/cusomter', (req, res) => {
59     if(!req.query.email) {
60         return res.status(400).send('Missing URL parameter: email')
61     }
62     CustomerModel.findOneAndUpdate({
63         email: req.query.email
64     }, req.body, {
65         new: true
66     })
67         .then(doc => {
68             res.json(doc)
69         })
70         .catch(err => {
71             res.stats(500).json(err)
72         })
73 })
```

that the change actually went through we want to do that

Subtitles/closed captions (c)

(A)

Callback for put() - callback for catch()

1:25:07 / 1:38:54

The screenshot shows a code editor interface with a dark theme. On the left, there's a sidebar titled "Project" showing the directory structure of a Node.js application named "rest-api-workshop". The "routes" folder contains "customer.js", "index.js", "person.js", and "customer.model.js". The "src" folder contains "models" and "routes" subfolders. The "models" folder has "customer.model.js". The "routes" folder has "customer.js", "index.js", "package.json", and "package-lock.json". The "public" folder contains "500.html" and "index.html". The "node_modules" folder is listed as "library root".

The main editor area displays the content of "customer.js". The code handles HTTP requests for customers:

```
44     .then(doc => {
45       res.json(doc)
46     })
47     .catch(err => {
48       res.status(500).json(err)
49     })
50   })
51
52   // UPDATE
53   router.put('/customer', (req, res) => {
54     if(!req.query.email) {
55       return res.status(400).send('Missing URL parameter: email')
56     }
57     CustomerModel.findOneAndUpdate({
58       email: req.query.email
59     }, req.body, {
60       new: true
61     })
62     .then(doc => {
63       res.json(doc)
64     })
65     .catch(err => {
66       res.status(500).json(err)
67     })
68   })
69
70   // DELETE
71   router.delete('/customer', (req, res) => {
72     if(!req.query.email) {
73       return res.status(400).send('Missing URL parameter: email')
74     }
75     CustomerModel.findOneAndDelete({
76       email: req.query.email
77     }, req.body, {
78       new: true
79     })
80     .then(doc => {
81       res.json(doc)
82     })
83     .catch(err => {
84       res.status(500).json(err)
85     })
86   })
87
88 }

Callback for delete()
```

A large callout box with a black background and white text is overlaid on the bottom-left portion of the code, containing the text:

and then there is a method called find one and remove

At the bottom of the screen, there are standard video player controls: play, pause, volume, and a progress bar indicating the video is at 1:26:07 / 1:38:54.

Insomnia - POST Customer

PUT localhost:3000/customer?email=thomas@gmail.com Send 201 Created TIME 116 ms SIZE 94 B

No Environment Cookies JSON Auth Query Header Docs Preview Header Cookie Timeline

Filter REST API Workshop PUT POST Customer POST PUT Customer GET GET Person GET GET Person by QueryString GET GET Person by Params Projects

1 = {
2 "name": "Thomas Updated Anderson",
3 "email": "thomas@gmail.com"
4 }

1 = {
2 "_id": "5ab3f442f3d76b4ae4cb433b",
3 "name": "Thomas Anderson",
4 "email": "thomas@gmail.com",
5 "__v": 0
6 }

in the body here let's call it Thomas updated Andersen and we're

Subtitles/closed captions (c) (4)
SUBSCRIBE

▶ ▶ 🔍 1:27:07 / 1:38:54 ⏪ ⏴ CC ⏵ ⏷ ⏸ ⏹

Insomnia - GET Customer

GET localhost:3000/customer?email=thomas@gmail.com

Send 200 OK TIME 97.3 ms SIZE 102 B

No Environment Cookies JSON Auth Query Header 1 Docs Preview Header 8 Cookie Timeline

Filter 1 ...

REST API Workshop

PUT POST Customer

POST PUT Customer

GET GET Customer

GET GET Person

GET GET Person by QueryString

GET GET Person by Params

Projects

1 = {
2 " _id": "5ab3f442f3d76b4ae4cb433b",
3 " name": "Thomas Updated Anderson",
4 " email": "thomas@gmail.com",
5 " __v": 0
6 }

and now if we click send here it's going to retrieve whatever was sent in and

Subtitles/closed captions (c) (4)

SUBSCRIBE

▶ ▶ 🔊 1:28:07 / 1:38:54 ⏪ ⏴ CC ⏵ ⏷ ⏸ ⏹

The screenshot shows a code editor interface with a dark theme. On the left, the project structure is displayed:

- rest-api-workshop
- src
- routes
- customer.js

The right pane shows the content of `customer.js`:

```
let CustomerModel = require('../models/customer.model')
let express = require('express')
let router = express.Router()

// Create a new customer
// POST localhost:3000/customer
router.post('/customer', (req, res) => {
  if(!req.body) {
    return res.status(400).send('Request body is missing')
  }

  if(!req.body.email) {
    // ...
  }

  // let user = {
  //   name: 'firstname lastname',
  //   email: 'email@gmail.com'
  // }

  let model = new CustomerModel(req.body)
  model.save()
    .then(doc => {
      if(!doc || doc.length === 0) {
        return res.status(500).send(doc)
      }

      res.status(201).send(doc)
    })
    .catch(err => {
      res.status(500).json(err)
    })
})

// GET
// callback for put()
```

stuff like that so let's do a quick recap

```
+ Server has started on 3000
x [nodemon] restarting due to changes...
x [nodemon] starting 'node src/index.js'
Server has started on 3000
Thu Mar 22 2018 11:29:58 GMT-0700 (PDT) => /customer?email=thomas%40gmail.com { name: 'Thomas Updated Anderson', email: 'thomas@gmail.com' }
Thu Mar 22 2018 11:30:51 GMT-0700 (PDT) => /customer?email=thomas%40gmail.com {}
Thu Mar 22 2018 11:31:13 GMT-0700 (PDT) => /customer?email=thomas%40gmail.com {}
Thu Mar 22 2018 11:31:20 GMT-0700 (PDT) => /customer?email=thomas%40gmail.com {}
```

Subtitles/closed captions (c)



```
10      }
11      if(!req.body.email) {
12          // ...
13      }
14      // let user = {
15      //     name: 'firstname lastname',
16      //     email: 'email@gmail.com'
17      // }
18
19      let model = new CustomerModel(req.body)
20      model.save()
21          .then(doc => {
22              if(!doc || doc.length === 0) {
23                  return res.status(500).send(doc)
24              }
25
26              res.status(201).send(doc)
27          })
28          .catch(err => {
29              res.status(500).json(err)
30          })
31      })
32
33
34
35      // GET
36      router.get('/customer', (req, res) => {
37          if(!req.query.email) {
38              return res.status(400).send('Missing URL parameter: email')
39          }
40
41          CustomerModel.findOne({
42              email: req.query.email
43          })
44          .then(doc => {
45              res.json(doc)
46          })
47      })
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
```

model so we create a new instance of the customer model with that incoming JSON

```
+ Server has started on 3000
x [nodemon] restarting due to changes...
x [nodemon] starting `node src/index.js`
Server has started on 3000
Thu Mar 22 2018 11:29:58 GMT-0700 (PDT) => /customer?email=thomas%40gmail.com {}
Thu Mar 22 2018 11:30:51 GMT-0700 (PDT) => /customer?email=thomas%40gmail.com {}
Thu Mar 22 2018 11:31:13 GMT-0700 (PDT) => /customer?email=thomas%40gmail.com {}
Thu Mar 22 2018 11:31:20 GMT-0700 (PDT) => /customer?email=thomas%40gmail.com {}
```

Subtitles/closed captions (c)



The screenshot shows a code editor interface with a dark theme. On the left is a project tree titled "rest-api-workshop". It includes a "node_modules" folder, a "public" folder with "500.html" and "index.html", a "src" folder containing "models" (with "customer.model.js") and "routes" (with "customer.js", "person.js", "index.js", "package.json", and "package-lock.json"). The "customer.js" file in the "routes" folder is the active tab, showing the following code:

```
33      })
34
35      // GET
36      router.get('/customer', (req, res) => {
37        if(!req.query.email) {
38          return res.status(400).send('Missing URL parameter: email')
39        }
40        CustomerModel.findOne({
41          email: req.query.email
42        })
43        .then(doc => {
44          res.json(doc)
45        })
46        .catch(err => {
47          res.status(500).json(err)
48        })
49      })
50
51      // UPDATE
52      router.put('/customer', (req, res) => {
53        if(!req.query.email) {
54          return res.status(400).send('Missing URL parameter: email')
55        }
56
57        CustomerModel.findOneAndUpdate({
58          email: req.query.email
59        }, req.body, {
60          new: true
61        })
62        .then(doc => {
63          res.json(doc)
64        })
65        .catch(err => {
66          res.status(500).json(err)
67        })
68      })
69
```

A tooltip "callback for get()" is visible at the bottom of the code editor.

other methods on the route so you have
get and an update you have fine

The terminal window at the bottom shows the output of a Node.js application running on port 3000 using nodemon. The logs include:

- "Server has started on 3000"
- "[nodemon] restarting due to changes..."
- "[nodemon] starting 'node src/index.js'"
- "Server has started on 3000"
- Timestamped log entries for requests to "/customer?email=thomas%40gmail.com":
 - Thu Mar 22 2018 11:29:58 GMT-0700 (PDT) => /customer?email=thomas%40gmail.com {}
 - Thu Mar 22 2018 11:30:51 GMT-0700 (PDT) => /customer?email=thomas%40gmail.com {}
 - Thu Mar 22 2018 11:31:13 GMT-0700 (PDT) => /customer?email=thomas%40mail.com {}
 - Thu Mar 22 2018 11:31:20 GMT-0700 (PDT) => /customer?email=thomas%40mail.com {}

On the right side of the terminal, there are video player controls (play, pause, volume), a "Subtitles/closed captions (c)" button, and a "SUBSCRIBE" button.

A screenshot of a video player interface displaying a code editor and a terminal window. The code editor shows the `package.json` file for a Node.js project named "rest-api-workshop". The terminal window shows the command `ls` being run in the project directory.

```
rest-api-workshop [~/gh/theoutlander/rest-api-workshop] - .../package.json [rest-api-workshop]
Project rest-api-workshop package.json Index.js customer.model.js customer.js person.js index.html
  node_modules library root
  public
    500.html
    index.html
  src
    models
      customer.model.js
    routes
      customer.js
      person.js
      index.js
    package.json
    package-lock.json
External Libraries
```

```
1 {  
2   "name": "rest-api-workshop",  
3   "version": "1.0.0",  
4   "description": "",  
5   "main": "index.js",  
6   "scripts": {  
7     "start": "node src/index.js",  
8     "start-watch": "nodemon src/index.js",  
9     "test": "echo \"Error: no test specified\" && exit 1"  
10  },  
11  "keywords": [],  
12  "author": "",  
13  "license": "ISC",  
14  "dependencies": {  
15    "body-parser": "^1.18.2",  
16    "express": "^4.16.3",  
17    "mongoose": "^5.0.11"  
18  },  
19  "devDependencies": {  
20    "nodemon": "^1.17.2"  
21  }  
22}
```

```
Nicks-MBP:rest-api-workshop theoutlander$ ls  
node_modules          package-lock.json      package.json      public           src  
Nicks-MBP:rest-api-workshop theoutlander$
```

excrement things like that
there's a service called now

Subtitles/closed captions (c)

1:32:07 / 1:38:54

▶ CC ⚙️ 🎧

The screenshot shows a video player interface with a terminal session integrated into the code editor. The terminal window at the bottom displays the output of the 'now' command, indicating the deployment of the 'rest-api-workshop' application under the user 'theoutlander'. The code editor above shows the 'package.json' file for the project, which defines the application's metadata, scripts, dependencies, and devDependencies.

```
rest-api-workshop [~/gh/theoutlander/rest-api-workshop] - .../package.json [rest-api-workshop]
Project rest-api-workshop package.json
1  {
2    "name": "rest-api-workshop",
3    "version": "1.0.0",
4    "description": "",
5    "main": "index.js",
6    "scripts": {
7      "start": "node src/index.js",
8      "start-watch": "nodemon src/index.js",
9      "test": "echo \\\"Error: no test specified\\\" && exit 1"
10    },
11    "keywords": [],
12    "author": "",
13    "license": "ISC",
14    "dependencies": {
15      "body-parser": "^1.18.2",
16      "express": "^4.16.3",
17      "mongoose": "^5.0.11"
18    },
19    "devDependencies": {
20      "nodemon": "^1.17.2"
21    }
22  }
```

Terminal

```
+ Nicks-MBP:rest-api-workshop theoutlander$ now
> Deploying ~/gh/theoutlander/rest-api-workshop under theoutlander
> Your deployment's code and logs will be publicly accessible because you are subscribed to the OSS plan.
> > NOTE: You can use now --public or upgrade your plan (https://zeit.co/account/plan) to skip this prompt
> Using Node.js 8.10.0 (default)
> Ready! https://rest-api-workshop-vwlwvsvsk.now.sh (copied to clipboard) [4s]
> Synced 10 files (25.46KB) [4s]
> Initializing...
> Building
> ▲ npm install
> ✓ Using "package-lock.json"
> ✓ Installed 421 modules [3s]
```

server side and then it's unzipping it
and it is

Subtitles/closed captions (c)

1:33:07 / 1:38:54

rest-api-workshop [~/gh/theoutlander/rest-api-workshop] - .../package.json [rest-api-workshop]

Project 1: rest-api-workshop package.json

1 {
2 "name": "rest-api-workshop",
3 "version": "1.0.0",
4 "description": "",
5 "main": "index.js",
6 "scripts": {
7 "start": "node src/index.js",
8 "start-watch": "nodemon src/index.js",
9 "test": "echo \"Error: no test specified\" && exit 1"
10 },
11 "keywords": [],
12 "author": "",
13 "license": "ISC",
14 "dependencies": {
15 "body-parser": "^1.18.2",
16 "express": "^4.16.3",
17 "mongoose": "^5.0.11"
18 },
19 "devDependencies": {
20 "nodemon": "^1.17.2"
21 }
22 }
dependencies : express

Terminal

```
+ Nicks-MBP:rest-api-workshop theoutlander$ now  
+ Deploying ~/gh/theoutlander/rest-api-workshop under theoutlander  
+ Your deployment's code and logs will be publicly accessible because you are subscribed to the OSS plan.  
+ > NOTE: You can use `now --public` or upgrade your plan (https://zeit.co/account/plan) to skip this prompt  
+ Using Node.js 8.10.0 (default)  
+ Ready! https://rest-api-workshop-vwlwvsxvsk.now.sh (copied to clipboard) [4s]  
+ Synced 10 files (25.46KB) [4s]  
+ Initializing...  
+ Building  
+ ▲ npm install  
+ ✓ Using "package-lock.json"  
+ ✓ Installed 421 modules [3s]  
+ ▲ npm start  
+ Deployment complete!  
+ Running in sf01  
Nicks-MBP:rest-api-workshop theoutlander$
```

start command I think one of the things
that I missed here in

Subtitles/closed captions (c)

(4)

SUBSCRIBE

1: Favorites

1: Project

1: Structure

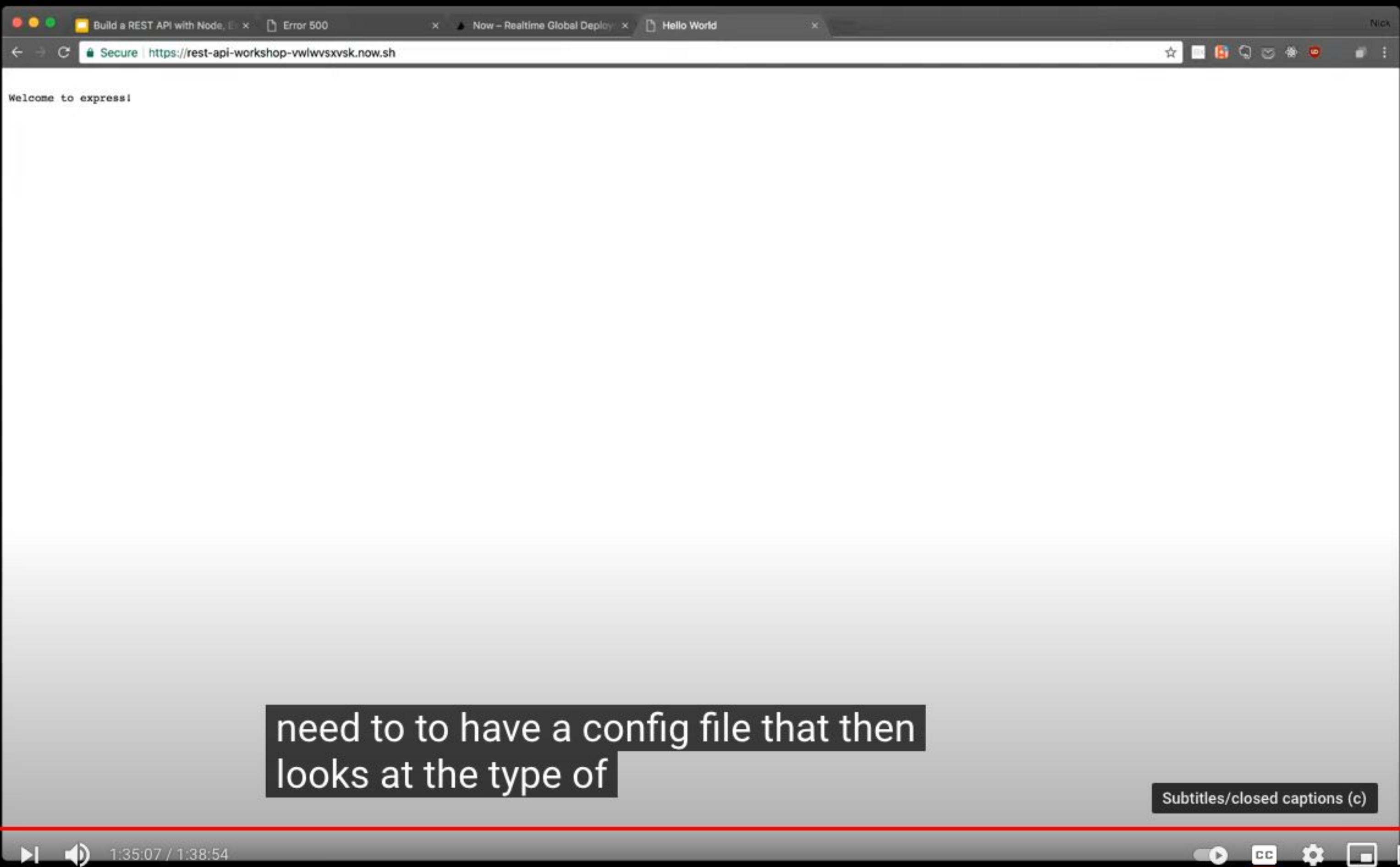
1: TODO

1: 1:34:07 / 1:38:54

1: CC

1: Settings

1: Log



need to have a config file that then
looks at the type of

Subtitles/closed captions (c)



Welcome to express!

models out and add specific methods on
it to do more complex operations

Subtitles/closed captions (c)





and over again.

2.1K



Let's assume your development stack consists of several libraries, such as React, Babel, Express, Jest, Webpack, etc. When you start a new project, you initialize all these libraries and configure them to work with each other.

With every new project that you start, you will be repeating yourself. You could also introduce inconsistencies in how these libraries are set up in each project. This can cause confusion when you switch between projects.

This is where boilerplates come in. A boilerplate is a template that you can clone and reuse for every project.

The modular Javascript ecosystem simplifies application development through various libraries, frameworks and tools. Boilerplates can be daunting if you don't understand the fundamentals of their underlying components.

Let's learn about these basic building blocks while creating our own.

[Click here for source on GitHub](#)

I am using Webstorm, Git, NodeJS 8.9, NPM 5.6, and React 16. Fire up your favorite IDE, create a blank project, and let's get started!

people start out with using some
existing boilerplate like create
Git Repository Setup

Subtitles/closed captions (c)

