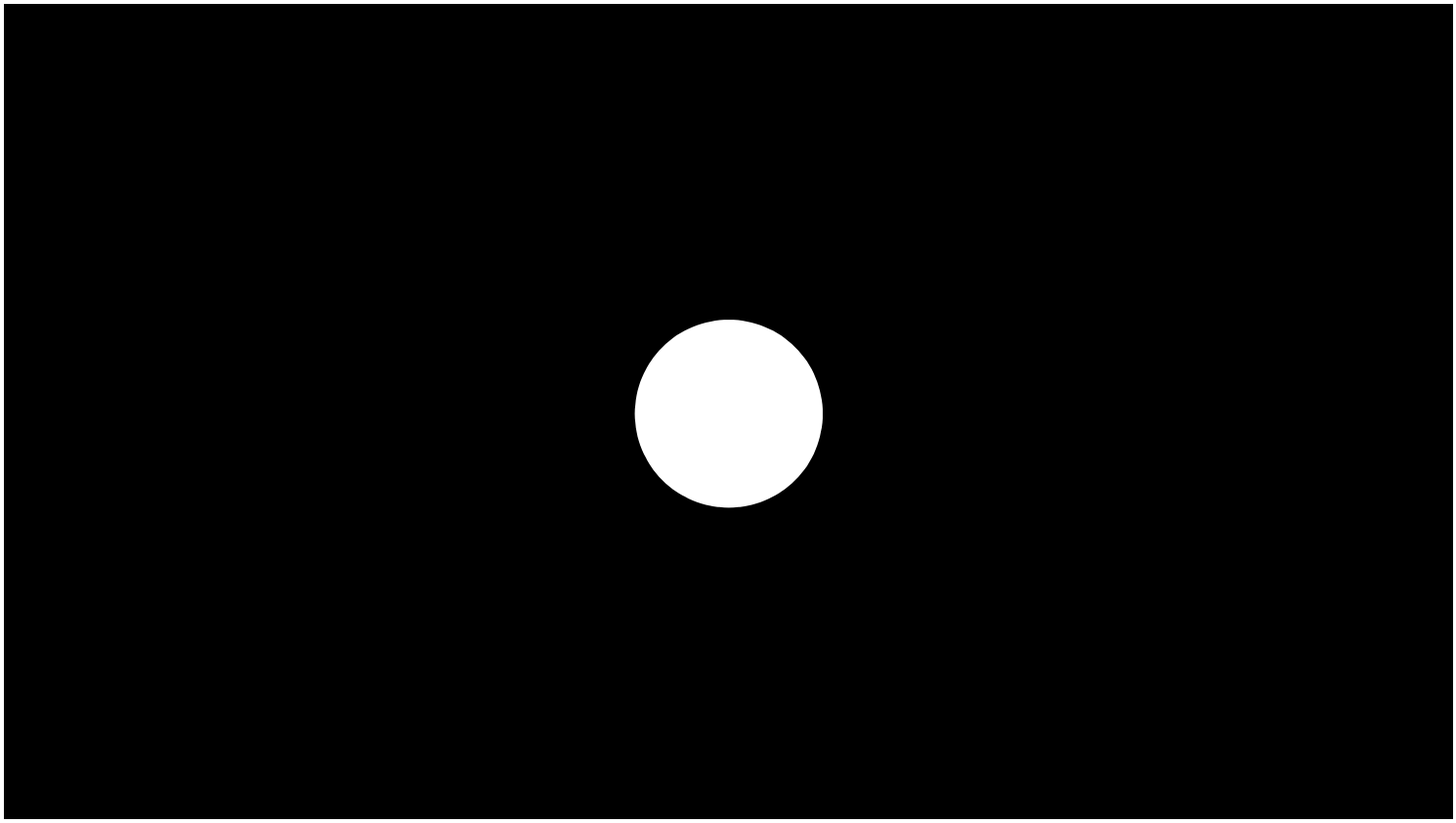
The Complete Node.js Developer Course (3rd Edition)



Course content | Overview | Q&A | Notes | Announcements | Reviews | Learr

Create a new note at 14:45

All lectures | Sort by most recent

13:59 **5. Debugging Node.js (Notes Apps)** 26. Debugging Node.js

use `debugger` in between code to debug

use `node inspect app.js --title="happy" --body="course"`

Go to `chrome://inpect` from chrome browser to further inspect via stackcalls , locals etc

0:08 **6. Asynchronous Node.js (Weather App)** 28. Section Intro: Asynchronous Node.js

node is asynchronous , non blocking , single threaded and event driven.

13:11 **6. Asynchronous Node.js (Weather App)** 31. Making HTTP Requests

npm request is depreceated , still can be used.

8:51 **6. Asynchronous Node.js (Weather App)** 31. Making HTTP Requests

```
api.weatherstack.com/current?access_key =
ea3a7922a46cce11b334fc277ec4a5ca
```

2:29 **6. Asynchronous Node.js (Weather App)** 31. Making HTTP Requests

darksky.net weather API is depreceated and no longer exists.

0:46 **6. Asynchronous Node.js (Weather App)** 33. An HTTP Request Challenge

geocoding is the process of taking an address like philadelphia united states and converting that into latitute and longitude pair ,

0:09 **6. Asynchronous Node.js (Weather App)** 34. Handling Errors

```
http://api.mapbox.com/geocoding/v5/mapbox.places/dhubri.json?
proximity=-74.70850,40.78375&access_token=pk.eyJ1IjoidG52cmFobWVkOTgiLCJ
hIjoiY2w1ZHR6MG1hMGp5YjNsbXB4b2R4NGR2eSJ9.mmGyPjUfcL-
YA1_Usu1_vA&limit=1
```

mapbox is used to get gecoding response like giving a place name as input and returning an array that includes all the arrays that is related to the place.

14:36 **6. Asynchronous Node.js (Weather App)** 35. The Callback Function

Asynchronous events using callbacks

```
const add1 = (a, b, callback1) => {
    setTimeout(() => {
        let sum = a+b;
        callback1(sum);
    }, 1000);
}

add1(1,2,(x)=>{
    console.log(x);
})
```

1:45 **6. Asynchronous Node.js (Weather App)** 35. The Callback Function

A call back function is a function that is passed to another function as argument.

0:01 **6. Asynchronous Node.js (Weather App)** 36. Callback Abstraction

`encodeURIComponent(address)`

this converts any address stiring containing special characters like '?' into some encoding format which so that the address url doesn't break .

// before

```
const url = "http"+address+" &query = lat,long;
```

//after

```
const url = "http"+encodeURIComponent(address)+" &query = lat,long;
```

10:43 **6. Asynchronous Node.js (Weather App)** 38. Callback Chaining

use `process.argv` to view command line arguments

`process.argv[1]` gives the first command line argument

11:33 **6. Asynchronous Node.js (Weather App)**

39. ES6 Aside: Object Property Shorthand and Destructuring

setting default value :

```
const product = {
    label:"kingfisher",
    price : 3,
    stock : 201,
    salesPrice : undefined
}

const {label:productLabel,stock , rating} = product;
```

10:14 **6. Asynchronous Node.js (Weather App)**

39. ES6 Aside: Object Property Shorthand and Destructuring

To set custom label for object elements

```
const product = {
    label:"kingfisher",
    price : 3,
    stock : 201,
    salesPrice : undefined
}

const {label:productLabel,stock , rating} = product;
```

13:17 **6. Asynchronous Node.js (Weather App)**

41. Bonus: HTTP Requests Without a Library

`response.end()` is used to actually send the request

8:48 **6. Asynchronous Node.js (Weather App)**

41. Bonus: HTTP Requests Without a Library

`response.on('end' , ()=>{});`

tells the request when to end the request

7:54 **6. Asynchronous Node.js (Weather App)**

41. Bonus: HTTP Requests Without a Library

`Reponse.on()` is a function that allows us to register a handler

8:58 **7. Web Servers (Weather App)** 47. Dynamic Pages with Templating

using `res.render()` we can render handlebar templates i.e views

5:16 **7. Web Servers (Weather App)** 47. Dynamic Pages with Templating

dynamic pages with templating

hbs is a express. js wrapper for the **handlebars. js javascript template engine.** Handlebars. js is a template engine to make writing html code easier.

to use hbs

`npm i hbs@4.0.1`

in src/app.js file add

`app.set('view engine', 'hbs');`

2:34 **7. Web Servers (Weather App)** 47. Dynamic Pages with Templating

`npm handlebar` : template engnine ,low level library , it can be user in a wide variey of settings like browser the server the desktop applicaiton with electron

1:11 **7. Web Servers (Weather App)** 47. Dynamic Pages with Templating

```
const app = express();
```

```
app.use(express.static(publicDir));//way to customer server
```

1:10 **7. Web Servers (Weather App)** 47. Dynamic Pages with Templating

`console.log(__dirname)` gives current directory path

`console.log(__filename)` gives current file path

1:05 **7. Web Servers (Weather App)** 47. Dynamic Pages with Templating

```
app.get('/' (req,res)=>{
return res.send('messaage');
});
```

3:02 **7. Web Servers (Weather App)** 49. Advanced Templating

Handlebars is a simple **templating language**.

It uses a template and an input object to generate HTML or other text formats. Handlebars templates look like regular text with embedded Handlebars expressions.

template

```
<p>{{firstname}} {{lastname}}</p>
```

3:22 **7. Web Servers (Weather App)** 50. 404 Pages

* is called wildcard

Put the below code below all other routes so that if no path was found you see error 404

```
app.get('*',(req,res)=>{
    res.send('My 404 Page Not Found');
})
```

10:33 **8. Accessing API from Browser (Weather App)** 54. The Query String

Question : what is the difference between req.params and req.query.

6:36 **8. Accessing API from Browser (Weather App)**

56. ES6 Aside: Default Function Parameters

Default function parameters

```
const transaction = (type, {label,stock}={}) =>{};
transaction ('order');
```

9:34 **8. Accessing API from Browser (Weather App)**

57. Browser HTTP Requests with Fetch

```
fetch('http://localhost:3000/weather?address=dhubri')
.then(response => response.json())
.then(data => console.log( data))
```

fetch returns a promise and response.json() also returns a promise .You cannot directly do

```
fetch('http://localhost:3000/weather?address=dhubri')
.then(response => response.json().data)
```

3:35 **8. Accessing API from Browser (Weather App)** 58. Creating a Search Form

all the templates in the .hbs replaces the partials templates in the actual files with the html content , it's similar to #include concept in c++

3:34 **8. Accessing API from Browser (Weather App)** 58. Creating a Search Form

you have `data.error` to check if data returns any error

6:16 **9. Application Deployment (Weather App)** 67. Deploying Node.js to Heroku

```
"scripts": {
   "start":"node src/app.js",
   "test": "echo \"Error: no test specified\" && exit 1"
 },
```

in package.json file in scripts section

now we can do locally `npm run start` to run the script

4:11 **9. Application Deployment (Weather App)** 67. Deploying Node.js to Heroku

HEROKU

before creating any project , add ssh keys to heroku using

`heroku keys:add`

To create a heroku App

`heroku add website_name` (must be unique website name)

1:11 **9. Application Deployment (Weather App)**

68. New Feature Deployment Workflow

`nodemon src/app.js -e js, hbs` is used to run the nodemon and also specifically monitor changes within files with .hbs and .js extensions

6:55 **10. MongoDB and Promises (Task App)** 77. The ObjectID

docs.mongodb.com

ObjectId(<hexadecimal>) : returns a 12 byte objectid value that consist of the following :

```
a. 4 byte value representing the seconds since the unix epoch
b. 5 byte random value
c. 3 byte counter starting with a random value
```

6:35 **10. MongoDB and Promises (Task App)** 78. Querying Documents

An objectid is not a string value , so we cannot directly do

```
findOne({ _id: "62e75bdcb0e76a19cca50c7b"})
```

this will return null , instead do

```
findOne({ _id: new ObjectID("62e75bdcb0e76a19cca50c7b")})
```

10:40 **11. REST APIs and Mongoose (Task App)** 83. Setting up Mongoose

in mongoose 6.0 above useNewUrlParse ,useCreateIndex is by default true

8:37 **11. REST APIs and Mongoose (Task App)**
85. Data Validation and Sanitization: Part I

NPM validator : used to validate fields .

4:54 **11. REST APIs and Mongoose (Task App)**
85. Data Validation and Sanitization: Part I

custom validator

9:25 **11. REST APIs and Mongoose (Task App)**

86. Data Validation and Sanitization: Part II

Operations like trimming , minlength is called sanitization

12:02 **11. REST APIs and Mongoose (Task App)** 87. Structuring a REST API

HTTP REQUEST EXPLAINED

13:30 **11. REST APIs and Mongoose (Task App)** 91. Resource Reading Endpoints: Part I

mongoose automatically converts string id to object id, so no need to bother about doing new Object()

9:56 **11. REST APIs and Mongoose (Task App)** 93. Promise Chaining

promise chaining

6:25 **15. Sending Emails (Task App)** 134. Creating a Production MongoDB Database

Steps to create cloud free server with mongo atlas

Help and Support

Terms

Privacy policy

Sitemap

Accessibility statement

business