18-Short-lnk-app video

Node HTTP and connect

- google node module http to learn more about it
- and for this courses we will be taking a lot of time on the Class: http.ServerResponse
- and this is going to allow us to respond to server requests
- and we are also going to be using Class: http.IncomingMessage
- and this is going to figure out exaclty what URL is trying to be loaded-it's going to let us know if we should redirect or show the application
- we also going to be using node connect
- this is going to allow us to intercept a request and make changes to it
- and this gets done via **middleware**
- **now** lets go to server/main.js
- our goal is to how to attach the middleware, how do we teach the server to do stuff
- first import a named export WebApp
- then we are going to use the connectHandlers library, and then we are going to use the use method
- and then we are going to pass it a function
- and for now we are just to log something to the screen

```
import { Meteor } from 'meteor/meteor';
import { WebApp }from 'meteor/webapp';


import '../imports/api/users';
import '../imports/api/links';
import '../imports/startup/simple-schema-configuration.js';


Meteor.startup(() => {
    WebApp.connectHandlers.use(()=>{
        console.log('this is from my custom middleware');
    });
});
```

- you will see this message on the terminal but over in the browser if you refresh the page is not going to stop loading
- the use() method will get 3 arguments,\
  - req-store things about the incoming requests, like what headers were used, and what URLs were they trying to use
  - res - let us respond to that http request , so maybe we want to redirect them to a different page
  - next - is just a function, it allows the application to keep on moving
- so to fix the bug that we had before all we have to do is use next

```
import { Meteor } from 'meteor/meteor';
import { WebApp }from 'meteor/webapp';

import '../imports/api/users';
import '../imports/api/links';
import '../imports/startup/simple-schema-configuration.js';

Meteor.startup(() => {
    WebApp.connectHandlers.use((req,res, next)=>{
        console.log('this is from my custom middleware');
        next();
    });
});
```

- now we are going to find out what its inside the req and res objects
- lets console.log req
- and you will see the output in the terminal
- we are going to be using a few properties available on req
- the req.url, and it returns what ever page they were trying to load

```
import { Meteor } from 'meteor/meteor';
import { WebApp }from 'meteor/webapp';

import '../imports/api/users';
import '../imports/api/links';
import '../imports/startup/simple-schema-configuration.js';

Meteor.startup(() => {
    WebApp.connectHandlers.use((req,res, next)=>{
        console.log('this is from my custom middleware');
        console.log(req.url);
        next();
    });
});
```

- and this is

```
I20171020-06:17:16.005(-7)? /links
```

- **if i go to the root of our app in the browser, this is what will show up**

```
I20171020-06:19:11.108(-7)? /
```

- **another property is req.method, it returns what http method was used**
- **we are also going to check req.headers and req.query**

```
import { Meteor } from 'meteor/meteor';
import { WebApp }from 'meteor/webapp';

import '../imports/api/users';
import '../imports/api/links';
import '../imports/startup/simple-schema-configuration.js';

Meteor.startup(() => {
    WebApp.connectHandlers.use((req,res, next)=>{
        console.log('this is from my custom middleware');
        console.log(req.url, req.method, req.headers, req.query);
        next();
    });
});
```

- and this is the result we get in the terminal

```
I20171020-06:22:12.997(-7)? /links GET { 'x-forwarded-proto': 'http',
I20171020-06:22:12.997(-7)?   'x-forwarded-port': '3000',
I20171020-06:22:12.997(-7)?   'x-forwarded-for': '127.0.0.1',
I20171020-06:22:12.998(-7)?   'accept-language': 'en-US,en;q=0.8,la;q=0.6',
I20171020-06:22:12.998(-7)?   'accept-encoding': 'gzip, deflate, br',
I20171020-06:22:12.998(-7)?   accept: 'text/html,application/xhtml+xml,application/xml;q=0.9,ima
ge/webp,image/apng,*/*;q=0.8',
I20171020-06:22:12.998(-7)?   'user-agent': 'Mozilla/5.0 (Macintosh; Intel Mac OS X 10_12_6) App
leWebKit/537.36 (KHTML, like Gecko) Chrome/61.0.3163.100 Safari/537.36',
I20171020-06:22:12.999(-7)?   'upgrade-insecure-requests': '1',
I20171020-06:22:12.999(-7)?   'cache-control': 'max-age=0',
```

```
I20171020-06:22:12.999(-7)?   connection: 'keep-alive',
I20171020-06:22:12.999(-7)?   host: 'localhost:3000' } {}
```

- if we notice that the end here there is an empty {} object
- if we go over to the browser , and the url we add a ?test=123 to the end of the url, this object will be filled with a key a pair
- it is called a query search string

```
I20171020-06:26:52.516(-7)?   host: 'localhost:3000' } { test: '123' }
```

- **we next are going** to set HTTP status code - then set http headers - then set HTTP body - then end HTTP request
- over in chrome tools there is a network tab, and under general

```
Request URL:http://localhost:3000/links?test=123
Request Method:GET
Status Code:200 OK
Remote Address:127.0.0.1:3000
Referrer Policy:no-referrer-when-downgrade
```

- now we are customize the status code
- and we can give it any status code we want but we are going to stick to 404
- to learn no more about status code just go to httpstatuses.com
- and now if you go back to the network tab in chrome tools you should see your 404

```
Request URL:http://localhost:3000/links?test=123
Request Method:GET
Status Code:404 Not Found
Remote Address:127.0.0.1:3000
Referrer Policy:no-referrer-when-downgrade
```

- Now we going to setup the http header on the response, setHeader() takes two arguments, the 1st is the name of the header you're trying to set and the second is the value for that header

```
import { Meteor } from 'meteor/meteor';
import { WebApp }from 'meteor/webapp';

import '../imports/api/users';
import '../imports/api/links';
import '../imports/startup/simple-schema-configuration.js';
```

```
Meteor.startup(() => {
    WebApp.connectHandlers.use((req,res, next)=>{
        console.log('this is from my custom middleware');
        console.log(req.url, req.method, req.headers, req.query);
        //Set HTTP status code
        res.statusCode = 404
        res.setHeader('my-custom-header', 'Jose was here')
        next();
    });
});
```

- and now we should see this in the network tab and response Headers

```
connection:keep-alive
content-encoding:gzip
content-type:text/html; charset=utf-8
date:Fri, 20 Oct 2017 13:42:13 GMT
my-custom-header:Jose was here
transfer-encoding:chunked
```

- and we can also write to the body - and we can completly overwrite what gets shown in the browser

```
import { Meteor } from 'meteor/meteor';
import { WebApp }from 'meteor/webapp';

import '../imports/api/users';
import '../imports/api/links';
import '../imports/startup/simple-schema-configuration.js';

Meteor.startup(() => {
    WebApp.connectHandlers.use((req,res, next)=>{
        console.log('this is from my custom middleware');
        console.log(req.url, req.method, req.headers, req.query);
        //Set HTTP status code
        res.statusCode = 404
        //Set HTTP Header
        res.setHeader('my-custom-header', 'Jose was here')
```

```
        //Set HTTP Body
        res.write('<h1>This is my middleware at work!</h1>');
        next();
    });
});
```

- and this H1 tag does overwrite what is on the page
- and now we are going to learn how to stop the request and that gets done via end() method, and this is going to completely take over the server
- lets comment out the write() method to show what happens to the page

```
import { Meteor } from 'meteor/meteor';
import { WebApp }from 'meteor/webapp';

import '../imports/api/users';
import '../imports/api/links';
import '../imports/startup/simple-schema-configuration.js';

Meteor.startup(() => {
    WebApp.connectHandlers.use((req,res, next)=>{
        console.log('this is from my custom middleware');
        console.log(req.url, req.method, req.headers, req.query);
        //Set HTTP status code
        res.statusCode = 404
        //Set HTTP Header
        res.setHeader('my-custom-header', 'Jose was here')
        //Set HTTP Body
        // res.write('<h1>This is my middleware at work!</h1>');
        //End HTTP request
        res.end();
        next();
    });
});
```

- and we get a blank page, because there is no data to load, because it immediately ends changes to the request and it sends that response back