Index

Some basic points

- There is big difference between authorization and authentication.
- JWTs, local login, session based authentication is all about who the user is.
- Authorization is like sigin with google, facebook, etc. Its like we don't care about who we are talking to but who has access to what resources.
- In this tut we will be talking about session and JWTs.

Chapter 1: Topics and Prerequisites

What is Passport JS

1. Welcome to Express Middlewares!

- On each HTTP request, Passport will use a "Strategy" to determine whether the requestor has permission to view that resource.

- If the user does not have a permission, a 401 Unauthorized Error is thrown.

2. Passport Strategies?

- Each strategy uess the Passport JS framework as a template.
- The Passport Local Strategy utilized Cookies, Express Sessions, and some authentication logic.

3. Passport JS in laymen language

- Passport JS is just a framework, that is a middleware that also allows individual developers to develop other middlewares called strategies that connect in to bigger middleware we called passport js framework. And then all of that is wrapped up into a bundle and can be used easily into our express app.

4. Summary of what is Passport JS

- In short, passport js is just a middleware and on every http request that a user calls to our express server the passport framework is going to first pickup what strategy we are using here and then it will use that strategy to validate if the user is authenticated or not. If the user is authenticated then only passport will let that user access the requested resource.

Chapter 2: Intro to HTTP Headers and Cookies

What is a HTTP Header?

- Headers is a whole big topic and it is not in the scope of this lecture.
- But we have to learn specifically about a HTTP header called `set cookie` in the cookie header in order to learn about how server side sessions work and interact with browser.
- To learn more about headers
- A HTTP client could be anything from an IOT device, a laptop, a phone, a *

General Headers

- These can be either request or response related.
- This is like general metadata about the request such as what is the url we are requesting, what type of method we are using, status code, etc.

Request Headers

- When we searched google, browser created a request header, which is basically instructions for the server that what data the request wants.
- We can also put other things like cookies, user agent

Response Headers

- HTTP headers are basically meta-data about our HTTP request.
- We can do similar things in response also, which would be set by server

- Server that we have requested data from, for example google here will set the response headers.
- The response headers will give additional instructions to the client that requested the data.

`set cookie` Header

- It generally gives key value pairs
- HTTP protocol is a stateless protocol, in laymen language, it constantly forgets what user has done on the site unless we have a way to remember that.
- Cookies are used on server-side as well as on client-side
- In simple terms, the server will see the credentials and if they are correct the server will send some data to client/browser to let it remember that the user has logged in
- If we don't have these type of persistance storage then everytime we refresh the page the previous stage, i.e, login process have to be redone.
- This is why we use `set cookie` and `cookie` header
- Once the browser will get cookie, it will attach it with the request headers to all the domains to which cookie belong to and it let the server know that the user has already logged in.
- Client->get req to google->google server sets cookie in the client's browser via response header
- So when we will refresh the page the browser will see what cookies are currently set in our browser and I will attach those cookies to every single request for the domain/context that is applicable to, for example google.com
- The this method of sending and receiving cookies is a powerfull way of authenticating users.

Summary of `set cookie` and `cookie` header

- once the user is authenticated correctly, server will set a cokkie corresponding to it and will send it back as response
- next time user will refresh the page or do something related to it, browser will first check if it have any cookie available.
- So the server will only have to valid cookie in order to make client access the data.
- Therefore, in this way we don't have to re-loggin the user again and again.
- We can also set expiry date to the cookies.

Chapter 3: Intro to Express Middleware

Why to learn about middleware?

- because both passport.js and express sessions are middlewares.
- below is a very basic way to run a node application

```
const express = require('express');
const app = express();
app.listen(3000)
```

- usual signature of middleware: `function (req, res, next)`
- here req means request, res means response and next means next middleware, express will automatically call the next middleware when execution of current one will finish

- In our get ('/') route we can also pass a function from outside if we don't want to define it in the parameter.
- for example:

```
function standardExpressCallback(reqObject, resObject, nextMiddleware){
    resObject.send("<h1>This is home page</h1>")
}
app.get('/', standardExpressCallback)
```

- just like we defined function standardExpressCallback, we can define other functions called **middlewares**

Defining middleware

- There can be two types of middlewares
    - Route specific middleware
    - Global middleware
- **NOTE**
    - The **order** of defining middleware whether it is global or route specific really **matters**

Route specific middleware

- below is an example of route spefic middleware

```
// this is an exmple of middleware
function middleware1(req, res, next){
    console.log("I am a middleware");
    next();
}
// this is how we call middleware
app.get('/', middleware1 ,standardExpressCallback)
```

Global Middleware

- We can declare a middleware globaly, the signature of the global middleware is same as we define route specific middleware
- To decalre a global middleware just write app.use(**middleware_name**) after we initialize our express app
- below is an example of global middleware

```
const app = express();

// this is the way to declare middleware globaly
// this means that this is the first middleware that we want to use in all of our
routes
app.use(middleware2)
```

```
// our global middleware
function middleware2(req, res, next){
    console.log("I am middleware2")
    next()
}
```

- Global middleware will be called everytime our app will receive a request!

Error handler middleware

- This is different type of middleware because it takes an extra parameter to know about error

Example of error handler

```
function errorHandler(err, req, res, next){
    // handle error
}
```

- we should try to declare the usage of error handler at the end because then if our app encounters error then it will just call the last error handler

Moulding properties in middlewares

- we can also customize the properties in a middleware and that properties will also be available to the next middleware in queue
- for example

```
app.use(middleware2)
app.use(middleware3)

function middleware2(req, res, next){
  req.customProperty = 100;
  // now the customProperty will also be passed to next middlewares
  next()
}

// we can access the customProperty in next middlewares as:
req.customProperty
```

How passport.js will work?

- The working of passport.js can be easily explained by how custom properties work in middlewares
- The passport.js will take the req object and will append some properties to it

Conclusion about middleware

- Middleware is an amazing concept and it makes our code much more readable and understandable
- This also helps us to add new features very easily like authentication and authorization
- It can also be used in checking if the users have administrator rights.
    - If the user will have administrator rights then it will take user forward
    - and if user doesn't have admin rights then we can show him some error like 401.