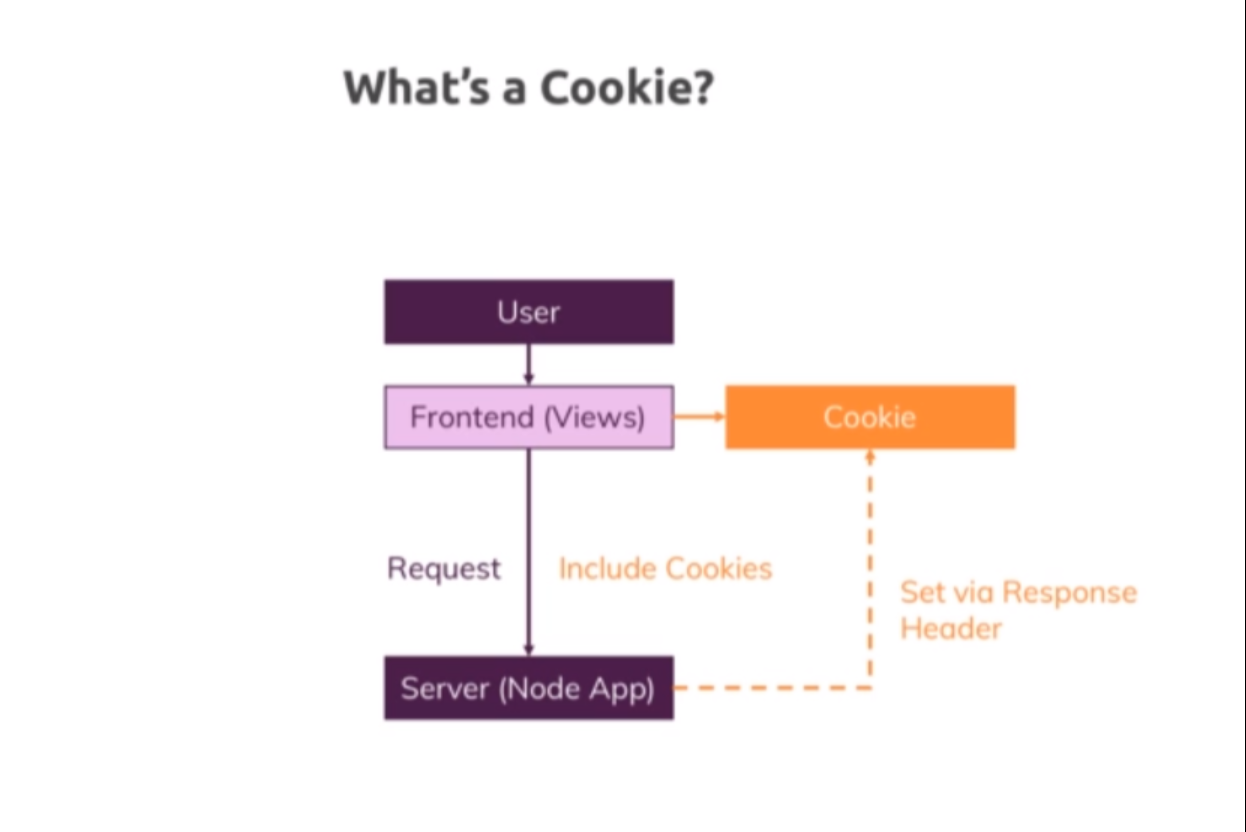
**Lecture 230**

**What is a Cookie?**



Refer notes

**Lecture 232**

**Optional: Creating the Login Form**

* Code 01-creating-the-login-form
* Controllers/auth.js
* Routes/auth.js
* Views/auth/login.ejs

**Lecture 233**

**Adding the request driven login solution**

* Code 02-adding-the-request-driven-solution
* Controllers/auth.js 🡪 postLogin
* Views/includes/navigation.ejs

**Lecture 234**

**Setting a Cookie**

* Code 02-adding-the-request-driven-solution
* Controllers/auth.js 🡪 getLogin , postLogin methods
* Controllers/shop.us 🡪 getIndex method
* So we found out that using a request for storing this is not ideal because the request is dead after sending a response, which alternatives do we have? Well one alternative would be some kind of global variable. You could use a global variable which you store in an extra file and which you export from that file and which you then change and that variable would actually survive your request cycles but since that variable would be shared across all requests, it would also be shared across all users and that is exactly where cookies can help us. With cookies we can store data in the browser of a single user and store data in that browser which is customized to that user which does not affect all the other users but can be sent with requests to tell us hey I already am authenticated and that is exactly what we will do here.

**Lecture 235**

**Manipulating Cookies**

* Code 02-adding-the-request-driven-solution
* Now if you want to extract cookies by the way, there also are third party packages which can help you with that but our approach has another flaw. Well obviously since I can access my cookies that easily in the developer tools, I can easily change them, I can go here and manipulate the value, for example if I set it to false and I reload, I'm actually still logged in because false is sent as text and text is always treated as true but we can simply add a comparison here and see if that value is equal to true, so to the text true here and now if I reload here, I'm not logged in anymore.
* If I change it back to true though and I do reload, I am. So the issue here is we can manipulate that from inside the browser and obviously you don't want to allow the users of your website to login by simply manipulating some cookie value. So whilst it is certainly interesting to store some data in the client side, especially things that are related to tracking users, advertisements tracking and so on, whilst this is interesting, sensitive data should not be stored in the browser because users can edit them as you see, we can edit our logged in cookie. So whilst cookies are generally a good thing for storing data across requests, it might not be the best approach in all scenarios and that is exactly something where sessions can help us with

**Lecture 236**

**Configuring Cookies**

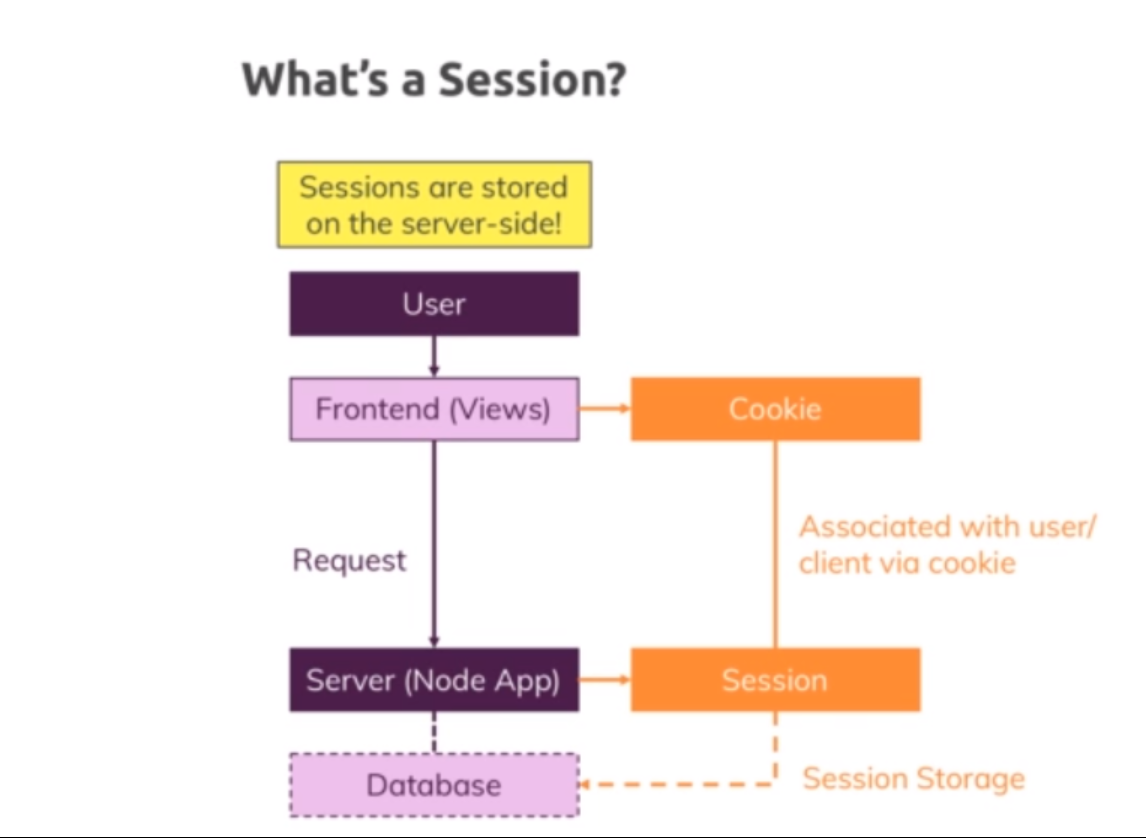
* Code 03-configuring-cookies
* Refer notes
* We can add multiple key value pairs

Res.setHeader(‘Set-Cookie’,’loggedIn=true’,’userId=3232’)

* we can also add a semi-colon after the key value pair and for example set expires to some expiration date, this date would have to follow a certain format, the http date format, I'll link it attached to this video here. So you could set a certain date when this cookie will expire because remember if you don't set this, it will expire once you close your browser. Alternatively to expires, you can set max age written like this and this is a number in seconds, how long that cookie should stay around, so we could set this to 10 for example and now if I click that login button here, I got logged in and now you see the expiry date also changed here, the expiry date if I decrease that, the expiry date here basically is today and now it already is expired and if I reload that page, is logged in is therefore gone.
* So this is something we can do and this is of course interesting if you want to control for example how long you want to track a user or we will actually use that together with authentication later, you could use this to also control how long an authenticated session stays active for a user, you might know that from your online bank where you timeout after a certain duration. You can also add a domain to which the cookies should be sent and here we again are on that tracking thing again. You can add secure just like this without an equal sign, just secure, this means this cookie will only be set if the page is served via https.
* Now I can't demonstrate this here because our local development server is not using https but we will eventually use https later in the course where I will show you how to set this up, so now you can already see however that I get an error if I try to reload login because I try to extract the value which is not there. So for now let's simply comment this out and set this always to false so that I can just show you how this cookie is now not set, if I reload and I click here, you don't see the cookie here because I added secure and it would only be set if we are serving the page via https and you can also set this to http only.
* Now if I do that and I go back to login and I click here, it is set but now it has this checkmark here in the http column and that means that now we can't access the cookie value through client side javascript, so in the scripts running in the browser. This can be an important security mechanism because it protects us against cross-site scripting attacks now because now your client side javascript where someone could have injected malicious code can't read your cookie values and that will be important later with authentication where a cookie will not store the sensitive information but an important part of authenticating the user. So this can be an extra security layer because now the cookie will still be attached to every request that is sent to the server but you can't read the cookie value from inside the browser javascript code. Obviously as you can tell, as a user in the developer tools, you can still read it but then again it's your own cookie and you will not store information like hey I'm logged in there because that would be easy to manipulate and you can't protect against that.

**Lecture 237**

**What is a Session?**



Refer Notes

**Lecture 238**

**Initializing the session middlware**

* Code 04-using-the-session-middleware
* Npm install –save express-session
* App.js
* app.use(

session({ secret: 'my secret', resave: false, saveUninitialized: false })

);

* secret – used for signing the hash which secretly stores our ID in the cookie ; can be any text
* resave: false – session will not be saved on evey request ; will be saved only something changed in the session ; this will improve performance
* saveUninitialized : true – This also basically ensures that no session gets saved for a request where it doesn’t need to be saved because nothing was changed about it.
* https://github.com/expressjs/session

**Lecture 239**

**Using the session middlware**

* Code 04-using-the-session-middleware
* Controllers/auth.js 🡪 postLogin method
* The session id wil be stored in the browser in the cookie named ‘connect.sid’
* Refer my implementation of logout feature.

**Lecture 240**

**Using MongoDB to store sessions**

* Code 05-using-mongodb-session-store
* Storing sessions in memory can overload the production server
* Npm install –save connect-mongodb-session
* App.js
* Whenever we modify the request.session property , it will be updated in the sessions collections in the database

**Lecture 241**

**Sessions and Cookies - A Short Summary**

* Refer notes

**Assignment 5**

**Sessions and Cookies**

* Code 05-using-mongodb-session-store\05-using-mongodb-session-store - with my assignment solution
* Code assignment-solution 🡪 max’s solution

**Lecture 242**

**Deleting a cookie**

* Code 06-deleting-a-cookie
* Navigation.ejs 🡪 adding logout button
* Controllers/auth.js 🡪 postLogout method
* Now let's head back to our application and first of all, let's login. Now let me open the developer tools again, I've got no session cookie here, I can click login and redirected session cookie is set and we can now use that session. Now if we go to compass real quick and we refresh, we see we have four objects now which makes sense, we had three before, now we have four and now let me click log out. We are redirected, the session cookie here actually still exists but you see the session was deleted over there and the session cookie still exists but that is no problem because no matching session will be found so that is fine, it's basically not doing anything and it will be renewed once we login again, then this will be overwritten and when we close the browser, it would also be deleted because it's not a permanent cookie, it's a session cookie which means it's a cookie that does not have an expiry date in the future, it does not have a max age, it will simply get deleted when we close the browser and it's worthless in this state here.

**Lecture 243**

**Fixing some minor bugs**

* Code 07-fixing-some-minor-bugs

**Lecture 244**

**Making add to cart work again**

* Code 08-two-tiny-improvements
* Before using sessions , when we set the user object in request, it was a mongoose object which had all the methods. When we used session, adding the user to it , it will add only the data. That user object doesnot have the methods we defined the User mongoose model.
* Solution to this is : when we login , in the postLogin method , we will fetch the user from db and store it in the session object as we did before. Along with this , we register a middleware which executes for all requests. This middle ware basically takes the user id from the req.session.user which we set in the postLogin method, and gets the user object from db, which will be a Mongoose object which have all the methods. This user object will be set in the req.
* App.js, controllers/auth.js 🡪 postLogin method
* Refer notes

**Lecture 245**

**Two tiny improvements**

* Code 08-two-tiny-improvements
* Controllers/auth.js 🡪 postLogin method -> using save method to save session
* Now one thing I just noticed is in the views, if I go to my product detail, there this add to cart of course also should only be rendered if I am authenticated, so I'll grab that link from product list and replace this here to make sure this works correctly because previously, if I clicked on details even though I'm not logged in, I did see my button there, now if I login and I go to the, whoops, details I see it here, I don't see it if I'm logged out. Now one other thing you might notice is if you do login like this, you sometimes might end up in a scenario where after logging in, the view didn't update accordingly. Now I fail to reproduce this at the moment but you might see this, that you login and still some items are missing and you need to reload the page to get there.
* The reason for this is that in auth.js when I have post login here, I do set my session and when I then redirect, when I send a response, the session middleware goes ahead and creates that session and that means it writes it to mongodb because we use the mongodb sessions store and it sets the cookie. Now the problem we can face here is writing that data to a database like mongodb can take a couple of milliseconds or depending on your speed even a bit more milliseconds. The redirect is fired independent from that though, so you might redirect too early. Now to be sure that your session has been set, you can use request session here and call the save method, you normally don't need to do that but you need to do it in scenarios where you need to be sure that your session was created before you continue because here, you can pass in a function that will be called once you're done saving the session.

**Lecture 246**

**Wrapup**

* Refer notes

