SESSIONS AND AUTH DATA FLOW

Not web security!

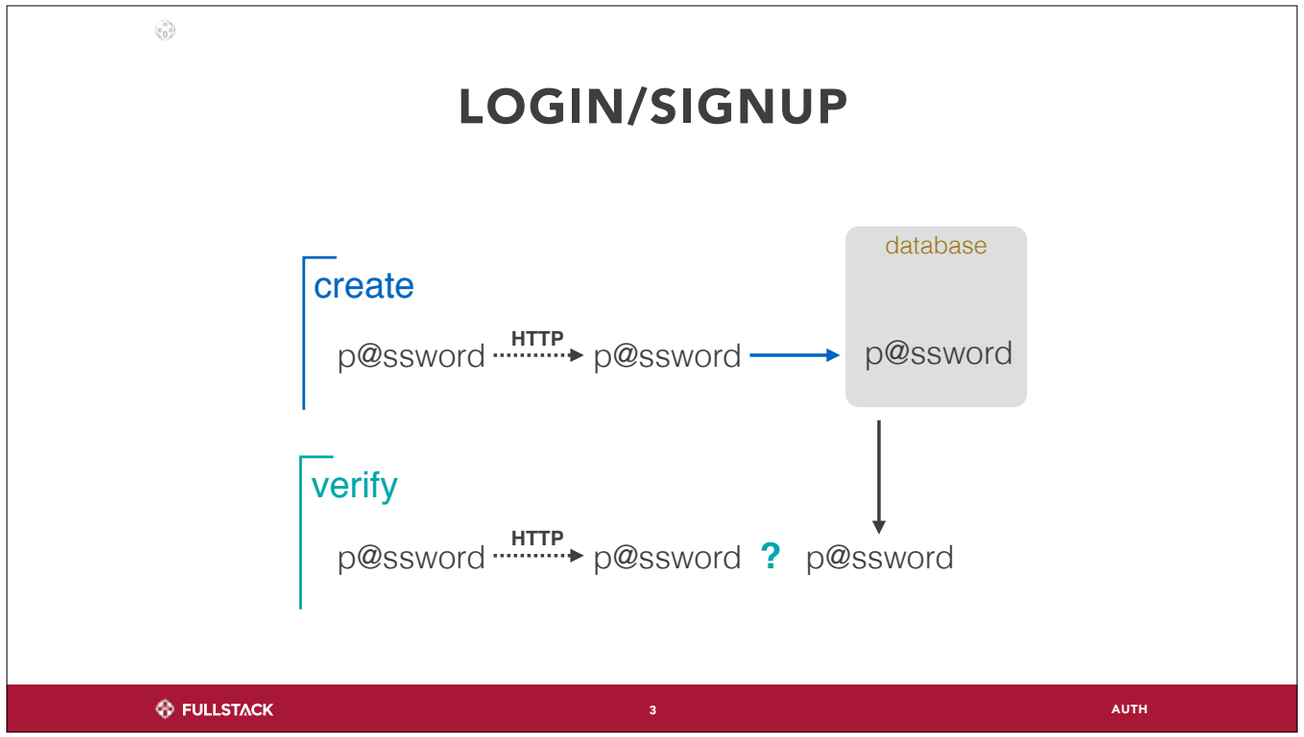
Let’s talk about authentication data flow. First of all, disclaimer: we’re not talking about web security today. We’re just going to talk about the flow of data involved in logging in users and keeping them logged in.

AAA

- **Authentication**
  - “this person is who they say they are”
- **Authorization**
  - “this person is allowed to do X, Y, Z”
- **Accounting**
  - “who the heck is using all our bandwidth?”

You may have heard of the three A’s of web development….

What we’re dealing with today is Authentication…



Here’s a general outline of the login/signup workflow, at least the way we understand it right now. It’s more or less what you’d expect.

STAYING LOGGED IN

However, once we log in, we have this new problem - how to keep that user logged in. (next)

THE PROBLEM

What if we want to store some information about each client/server relationship (session)?

We know what doing this looks like - when you're logged into Gmail or Spotify and you refresh the page, you're not asked to log in again. Some kind of persistent state is maintained so that Google or Spotify continues to know who you are for a while even after you refresh the page.

However, HTTP as we know it is a stateless protocol - one request, one response.

IDEAS?

- script variables
- database documents

What are some ideas for addressing this?

SCRIPT VARIABLES?

Data will not persist across browser pages

We can’t just put the user data in a variable somewhere in our Javascript, because that gets wiped out when we refresh the page.

DATABASE DOCUMENTS?

Yes, but login credentials would have to be sent with EVERY request

And we can store the login credentials in our database, but that means we need to send those credentials with every request.


COOKIE!

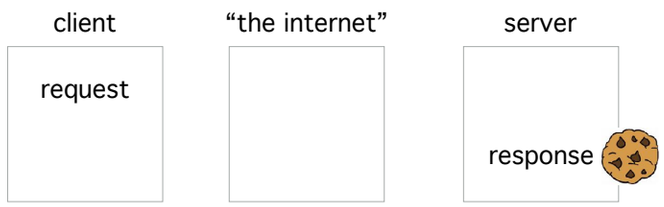
= small text file

What is a cookie? It’s just a small text file.

HTTP

Before we look at how cookies come into play, here’s the HTTP request-response cycle as we know it

WITH COOKIES



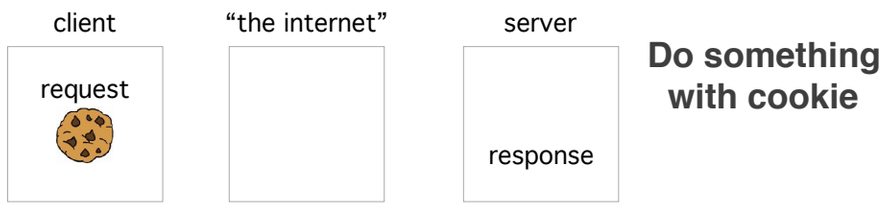
Now let’s introduce cookies to the mix

WITH COOKIES

client “the internet” server

…Now the client has this cookie on it, so the next time it makes the request, it will include it in the request

WITH COOKIES



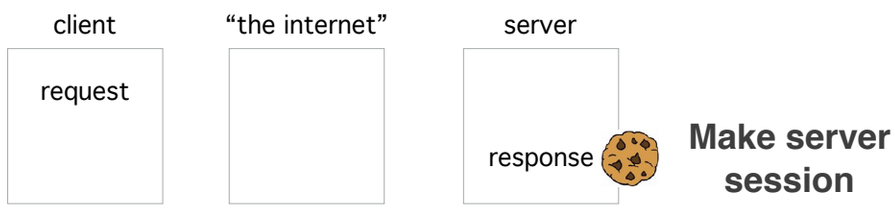
The server will check to the cookie, which will help it remind itself about the person making the request.

COOKIES & SESSIONS

No need to authenticate for every request

The big advantage that this gives us that we don’t need to actually authenticate with every request. Our server can use the cookie to remind it about who’s logged in

SESSIONS



Now we can use cookies to implement “stateful” sessions on the server

SESSIONS

client

“the internet”

server

SESSIONS

client

request

"the internet"

server

response

Look up session

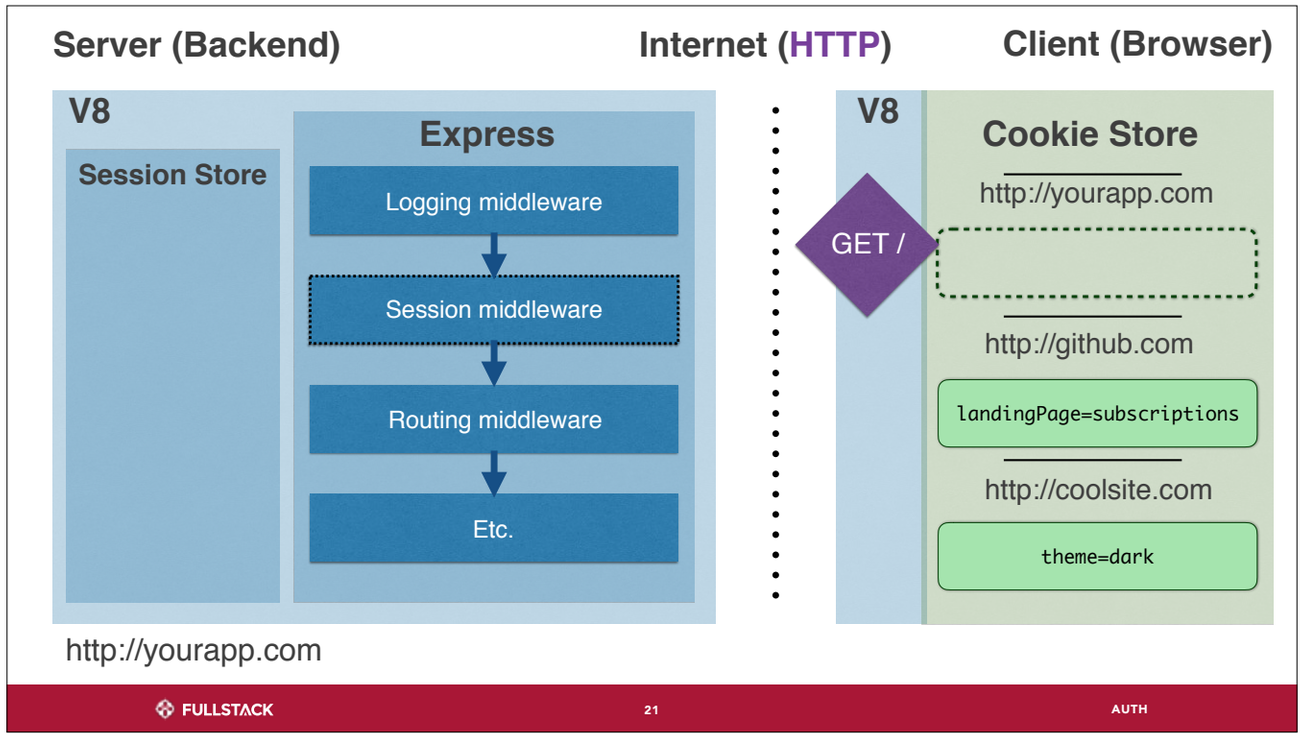
req.session = sessions[id]

COOKIES & SESSIONS

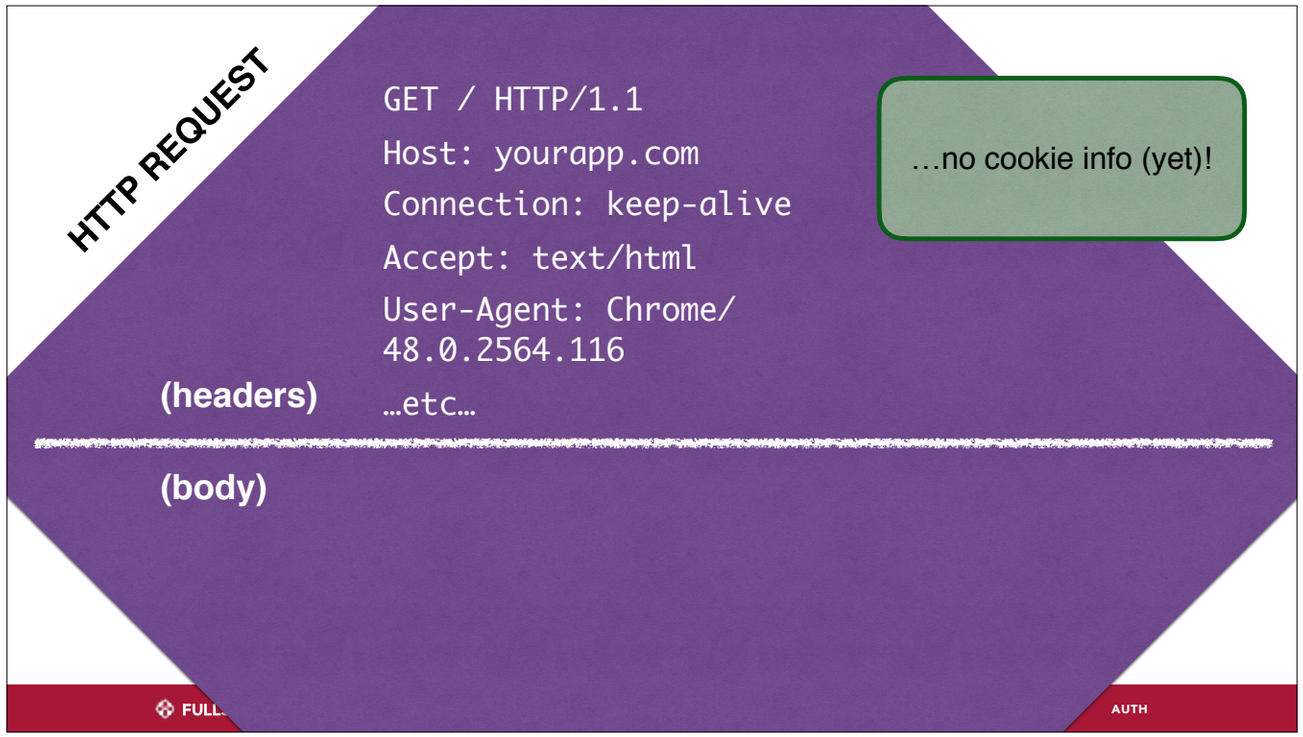
- Server gives client a cookie with ID only
- Client keeps cookie and sends with all requests
- Server loads request-specific session (stored in RAM)

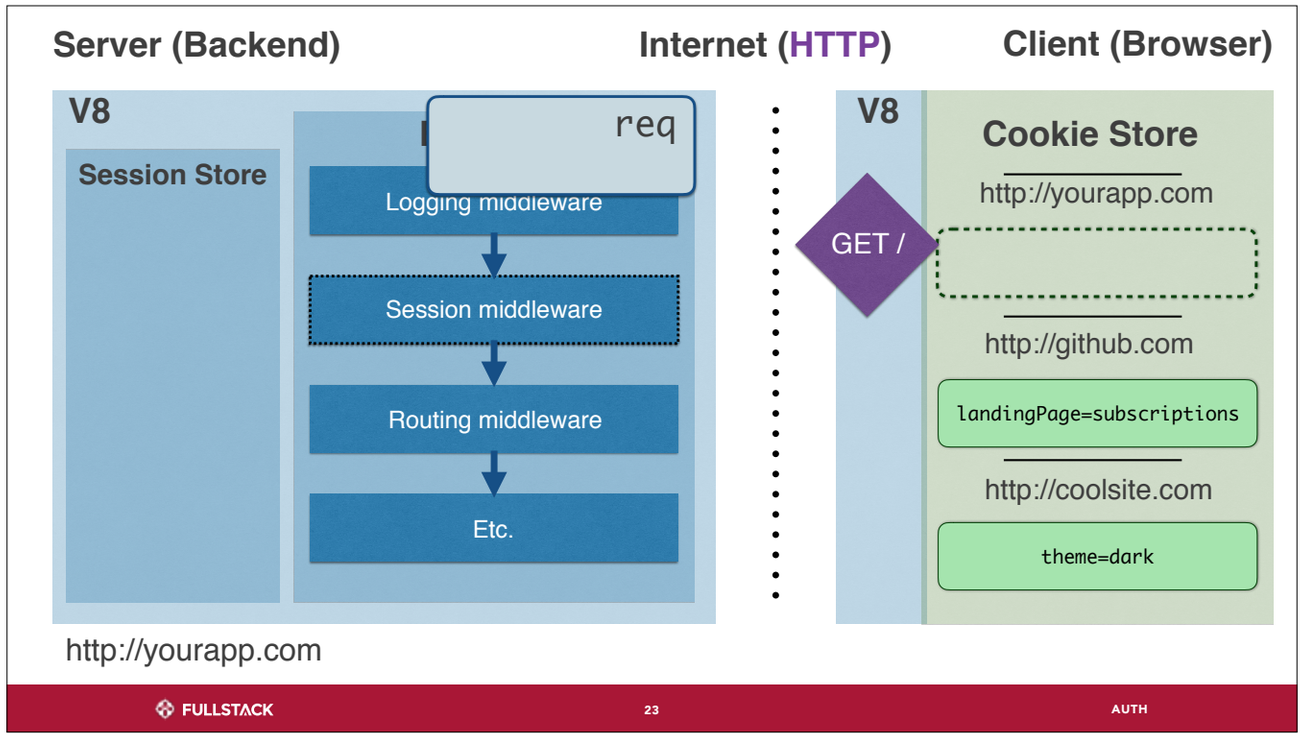
…and on this session object, we can keep track of a lot of things, but probably the most important is the identity of the user making the request.

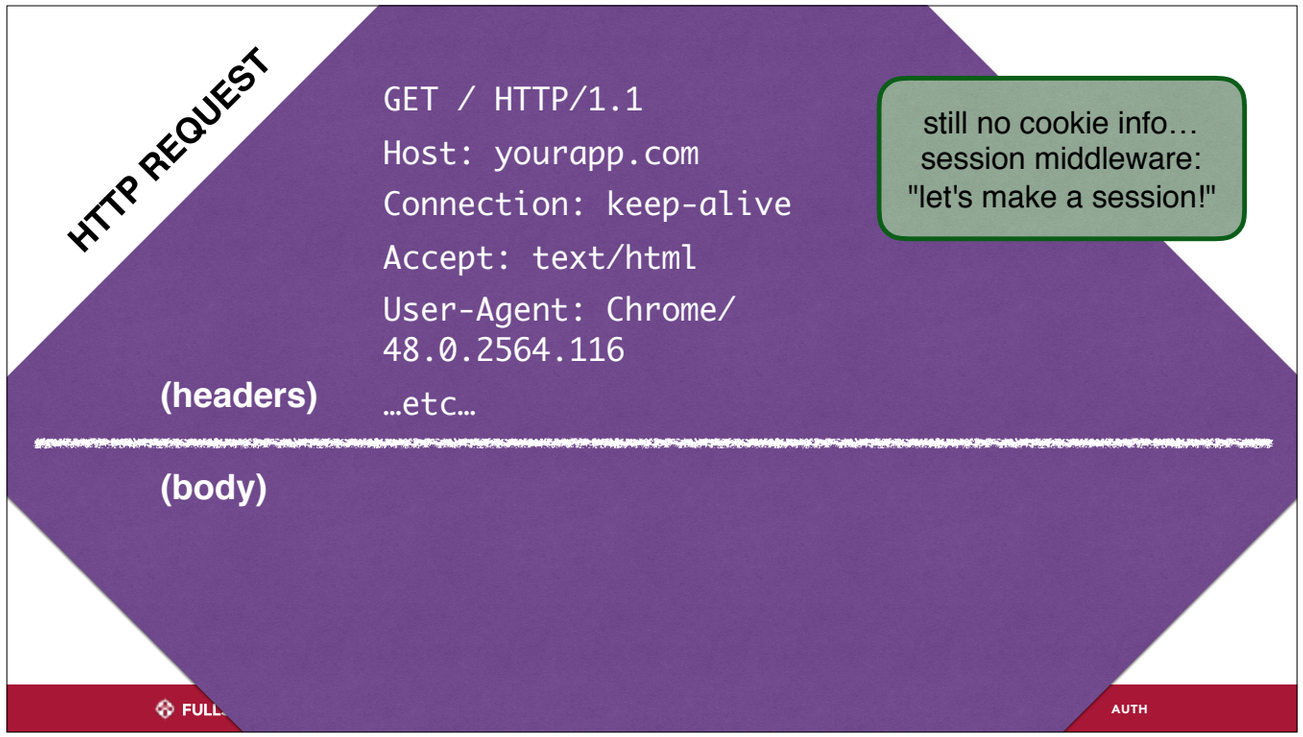

SESSIONS
IN
DETAIL



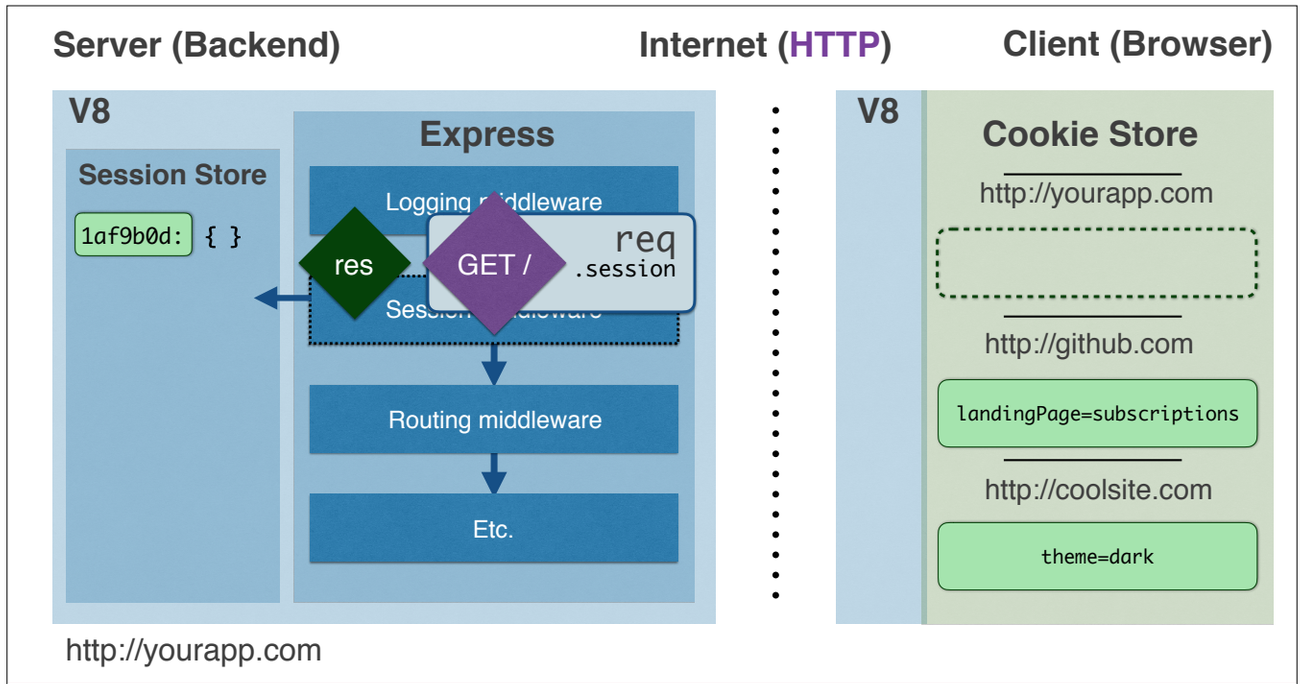
This is going to be the same data flow we just looked at, only in a little bit more detail
Here we have our server, which has our express app on it, which contains our logging middleware, our session middleware. etc
And here we have a browser, which has never visited our site before. We've visited other websites, like GitHub and coolSite, and we've got cookies from then, but they're only going to be sent for requests to GitHub or coolSite. We don't have any cookie for yourapp.com


HTTP REQUEST
GET / HTTP/1.1
Host: yourapp.com
Connection: keep-alive
Accept: text/html
User-Agent: Chrome/
48.0.2564.116
…etc…
(headers)
(body)
…no cookie info (yet)!
FULL
AUTH

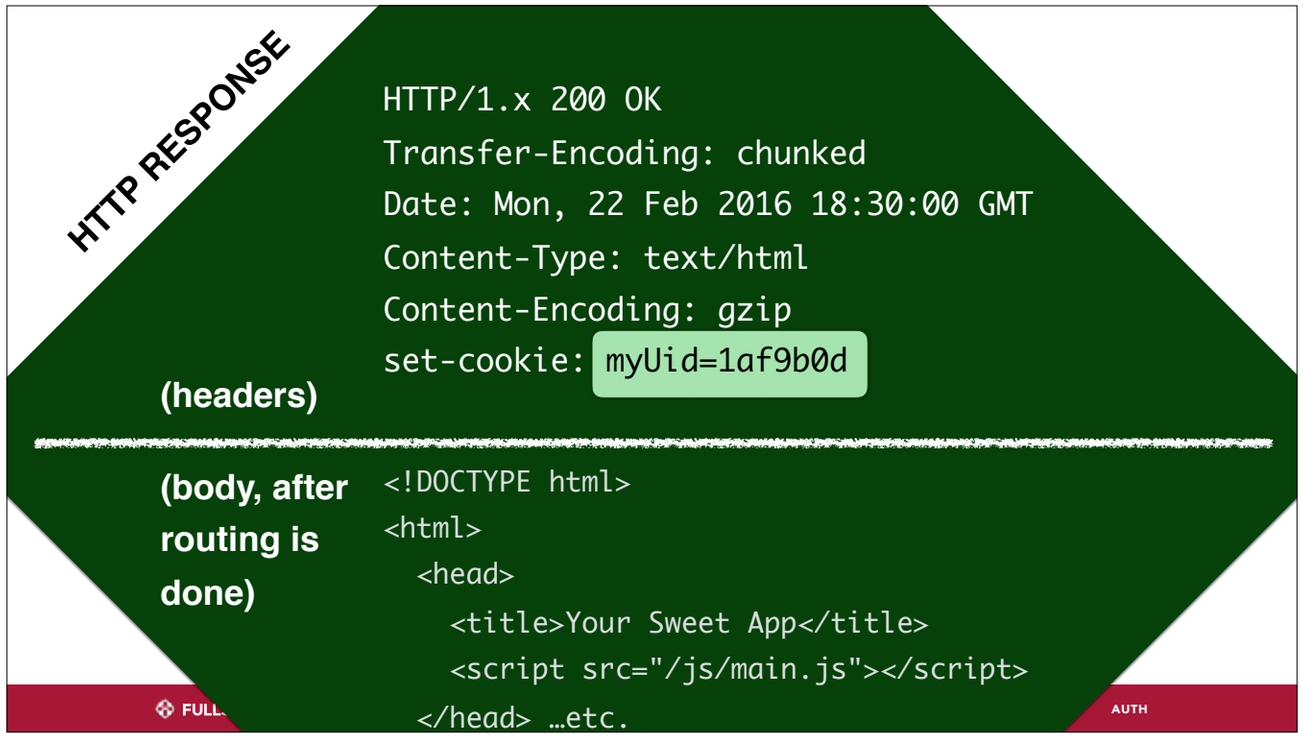

Server (Backend)
Internet (HTTP)
Client (Browser)
V8
Session Store
req
Logging middleware
Session middleware
Routing middleware
Etc.
http://yourapp.com
V8
GET /
Cookie Store
http://yourapp.com
http://github.com
landingPage=subscriptions
http://coolsite.com
theme=dark



When your session middleware sees this request, and notices that there's no cookie on it, it's like…let's get baking.

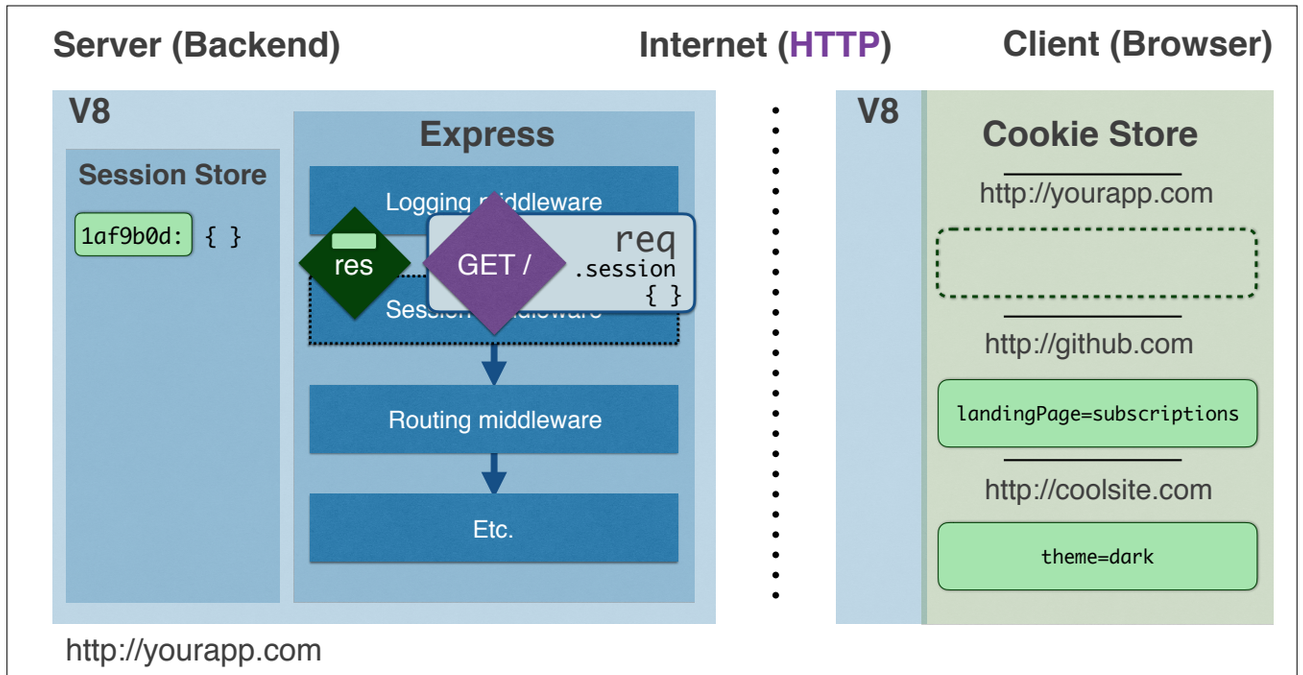


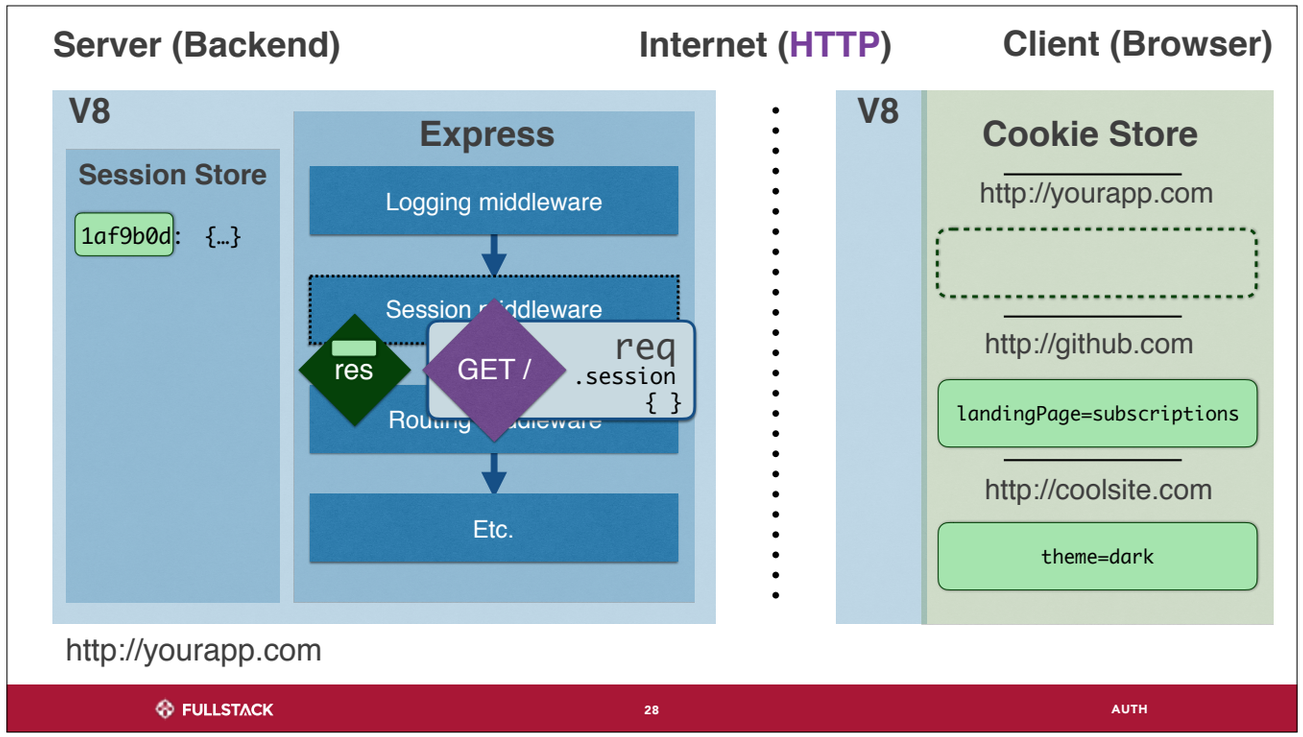
So the session middleware creates a new entry in its session store, which is just an object/hash table in memory holding key value pairs. The key here is going to be your session id, and the value is going to be any information that we want to persist across various requests.
It attaches these values to the request object as req.session, and it adds the session id to the cookie on the response object



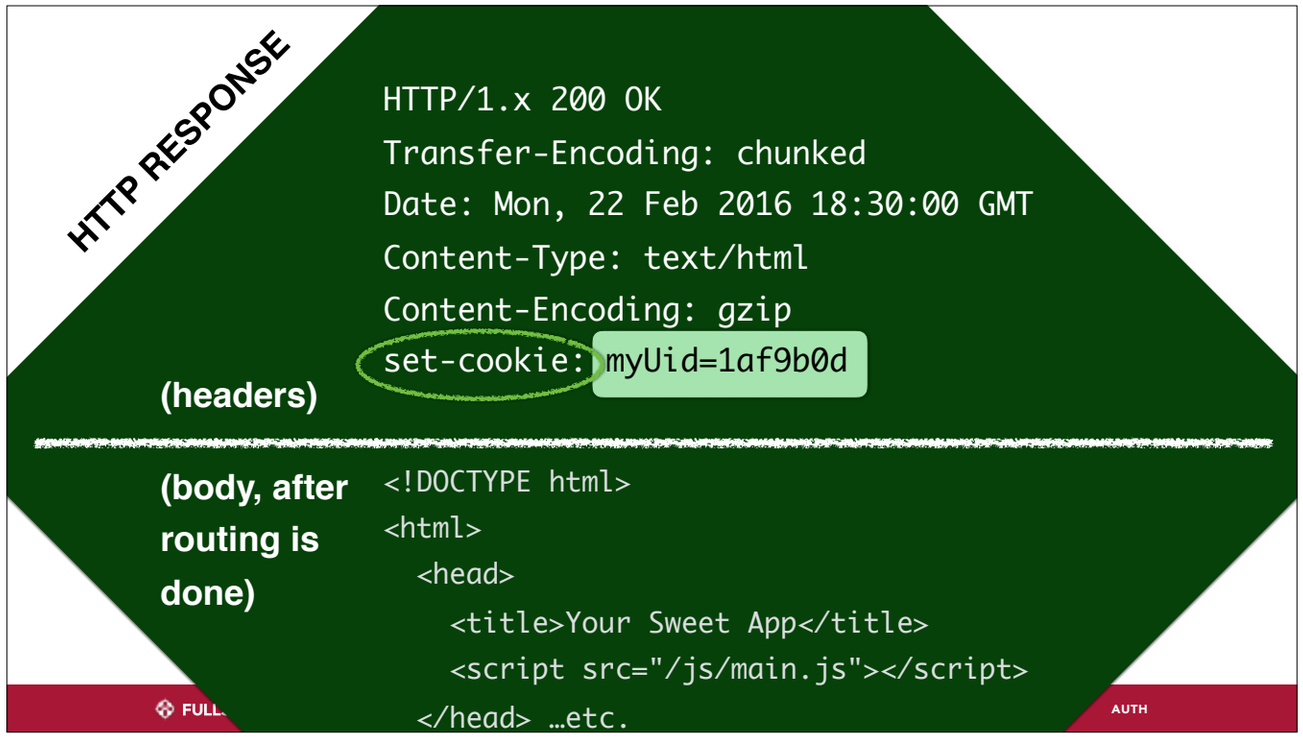
Remember that an HTTP responses (and requests) are just text that's formatted in a certain way, but thanks to express we can interface with them through javascript objects.

When we add the cookie containing our session id, it looks like this

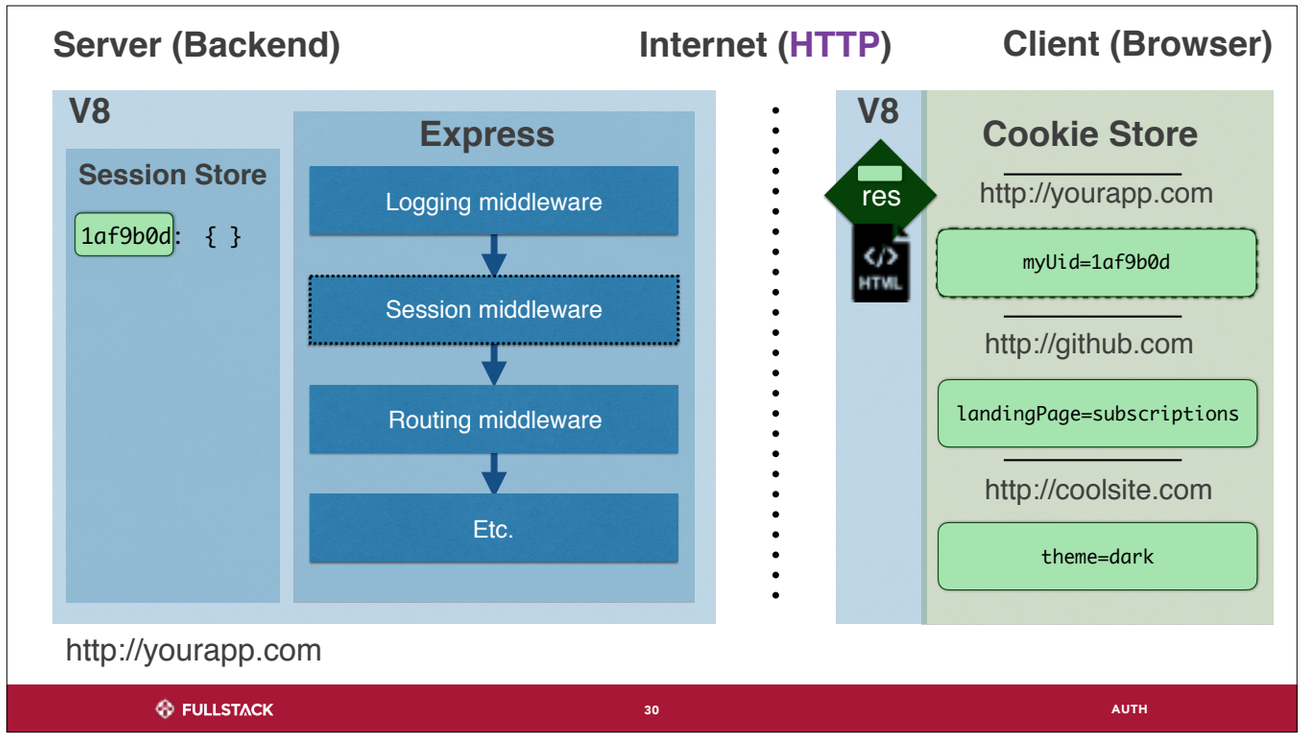

Server (Backend)
Internet (HTTP)
Client (Browser)
V8
Session Store
1af9b0d:
{ }
Express
Logging middleware
res
GET /
req
.session
{ }
Routing middleware
Etc.
http://yourapp.com
V8
Cookie Store
http://yourapp.com
http://github.com
landingPage=subscriptions
http://coolsite.com
theme=dark

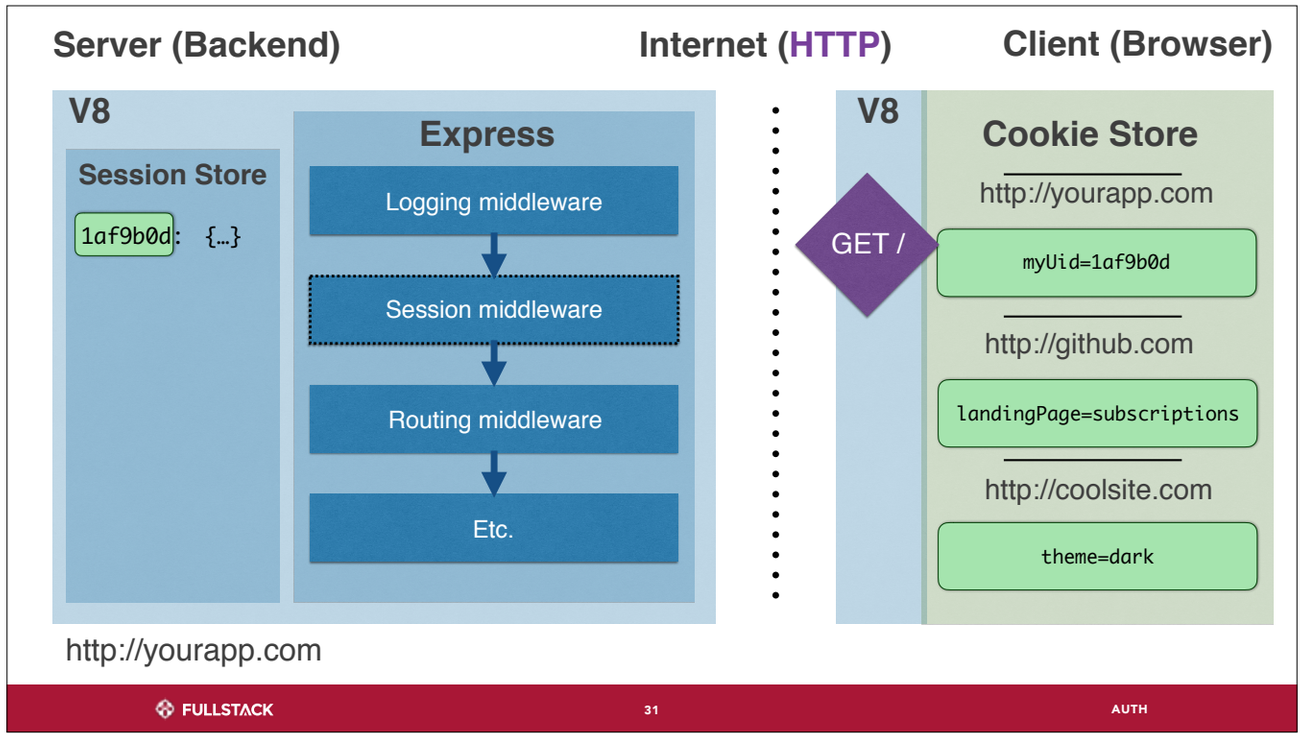


Now this response is ready to go back to the client…, and the big difference from what we're used to is now our response has this set-cookie header, which your browser knows what to do with.

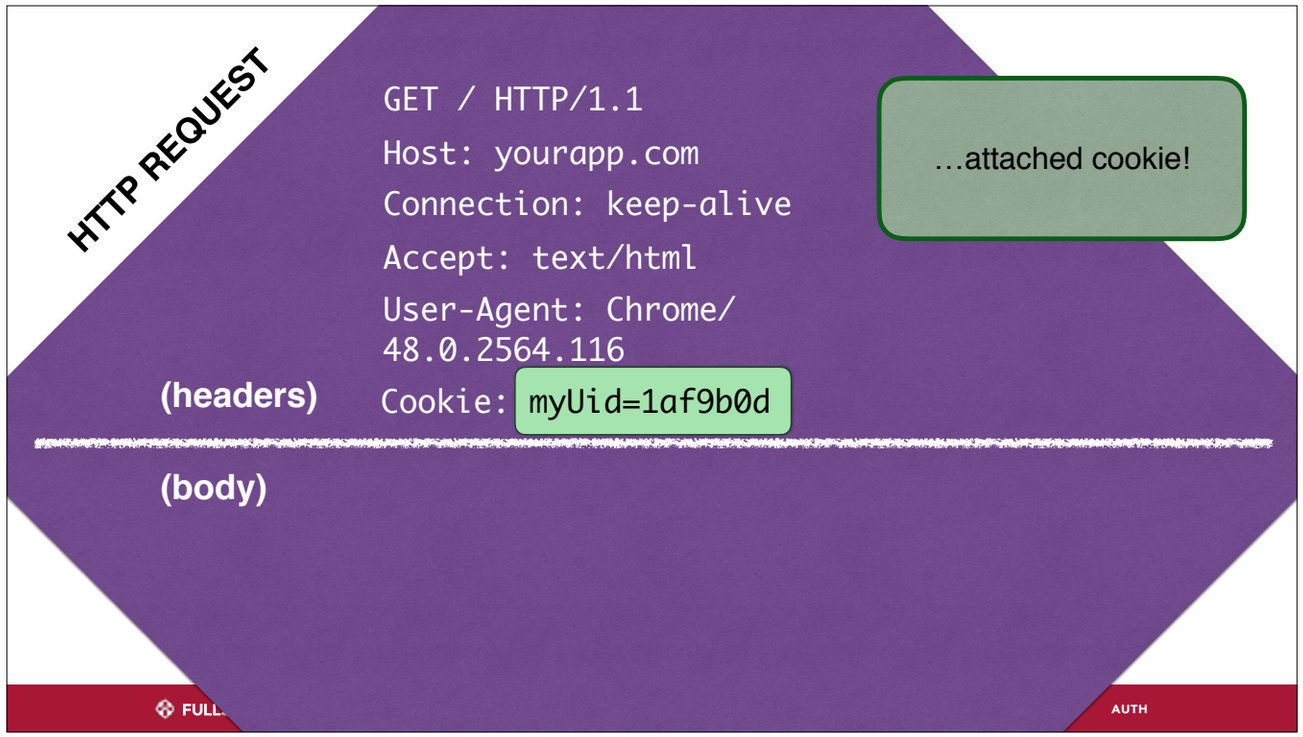


Your browser sees the set cookie header, and says, “awesome, I got a cookie”. I’m going to put this away in my storage.

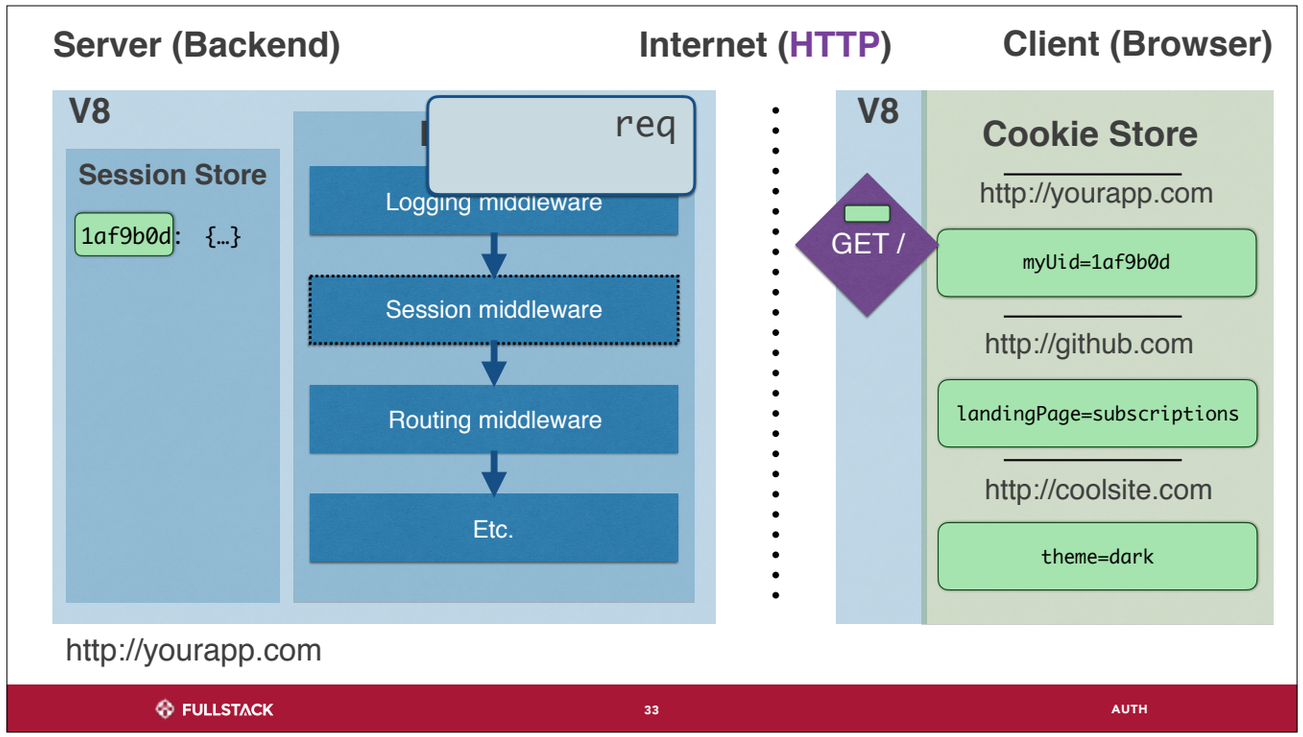

Server (Backend)
Internet (HTTP)
Client (Browser)
V8
Session Store
1af9b0d: { }
Express
Logging middleware
Session middleware
Routing middleware
Etc.
http://yourapp.com
V8
res
HTML
Cookie Store
http://yourapp.com
myUid=1af9b0d
http://github.com
landingPage=subscriptions
http://coolsite.com
theme=dark

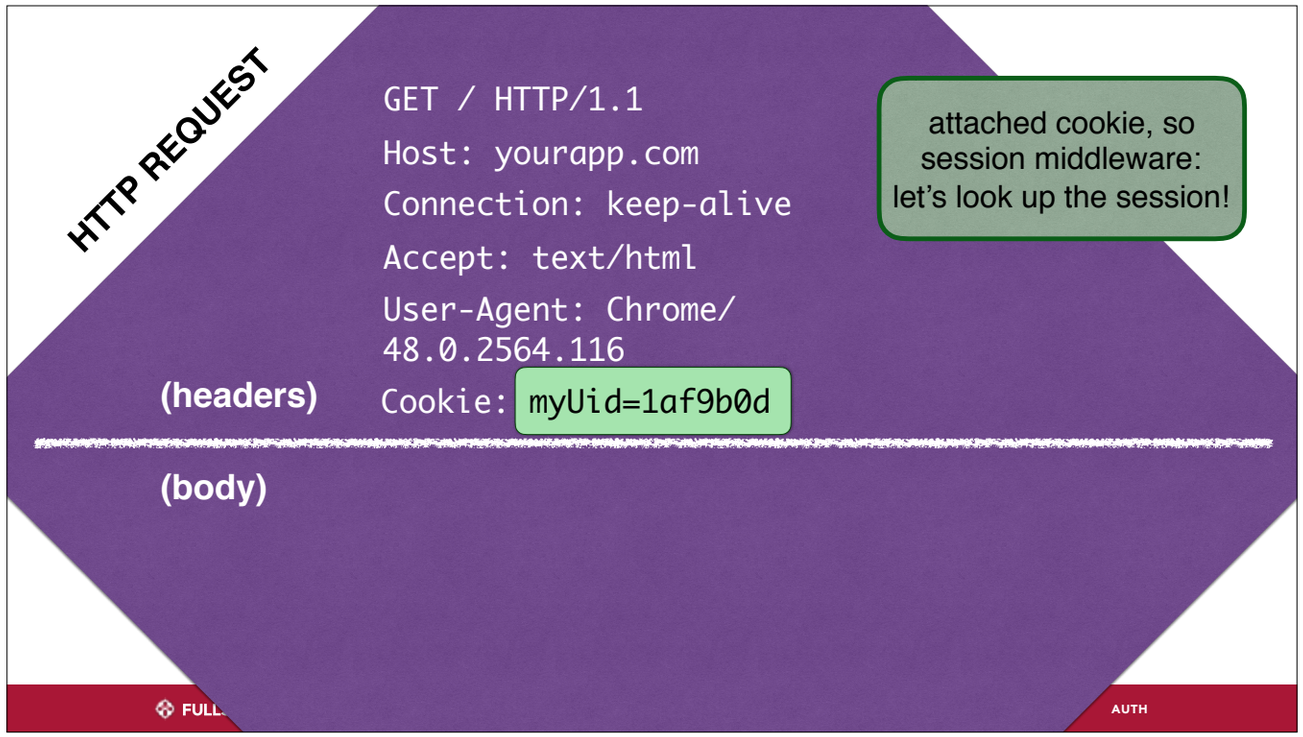


Now let’s imagine that this browser makes another request.

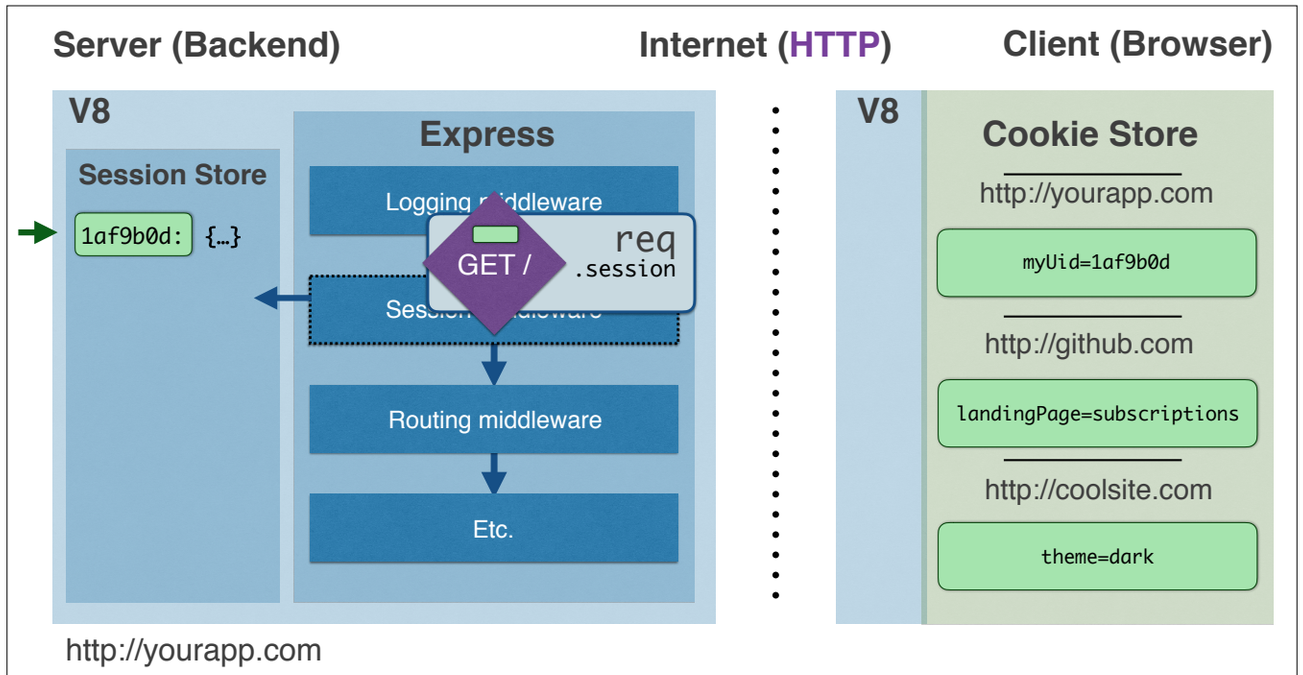


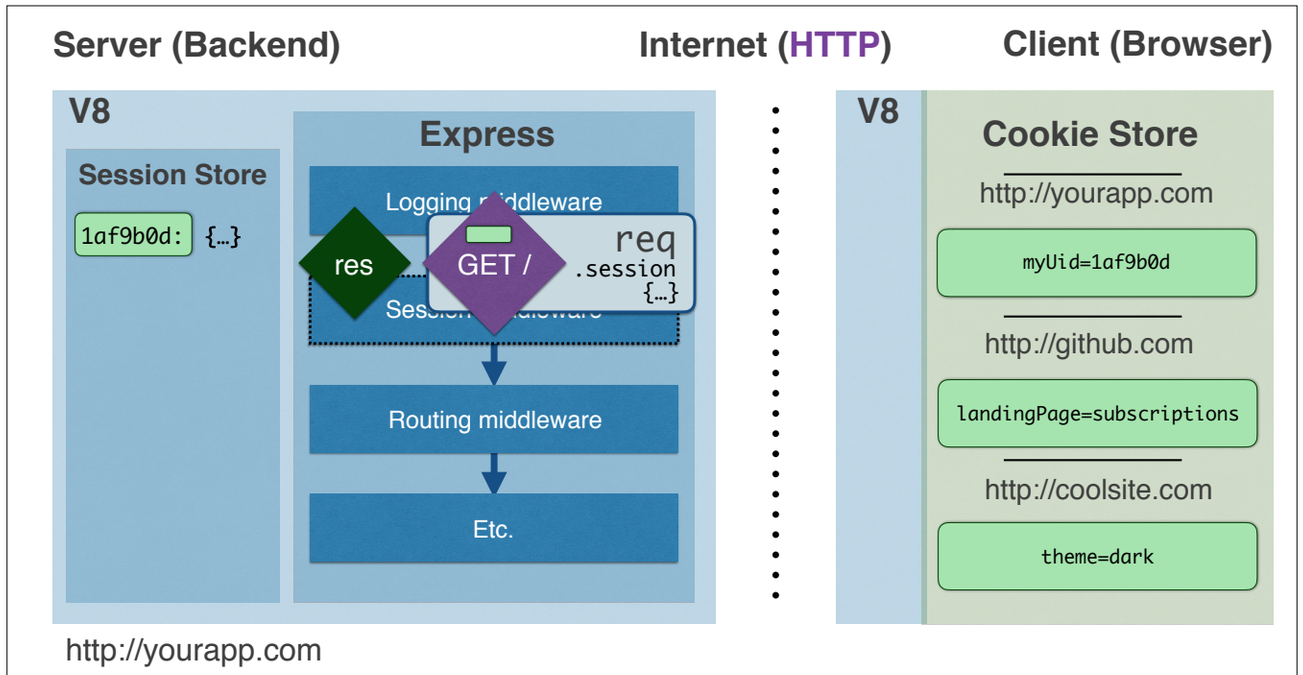
Only this time, because that browser has a cookie stored for this host, it includes it in the request.


Server (Backend)
Internet (HTTP)
Client (Browser)
V8
Session Store
1af9b0d: {…}
req
Logging middleware
Session middleware
Routing middleware
Etc.
http://yourapp.com
V8
GET /
Cookie Store
http://yourapp.com
myUid=1af9b0d
http://github.com
landingPage=subscriptions
http://coolsite.com
theme=dark



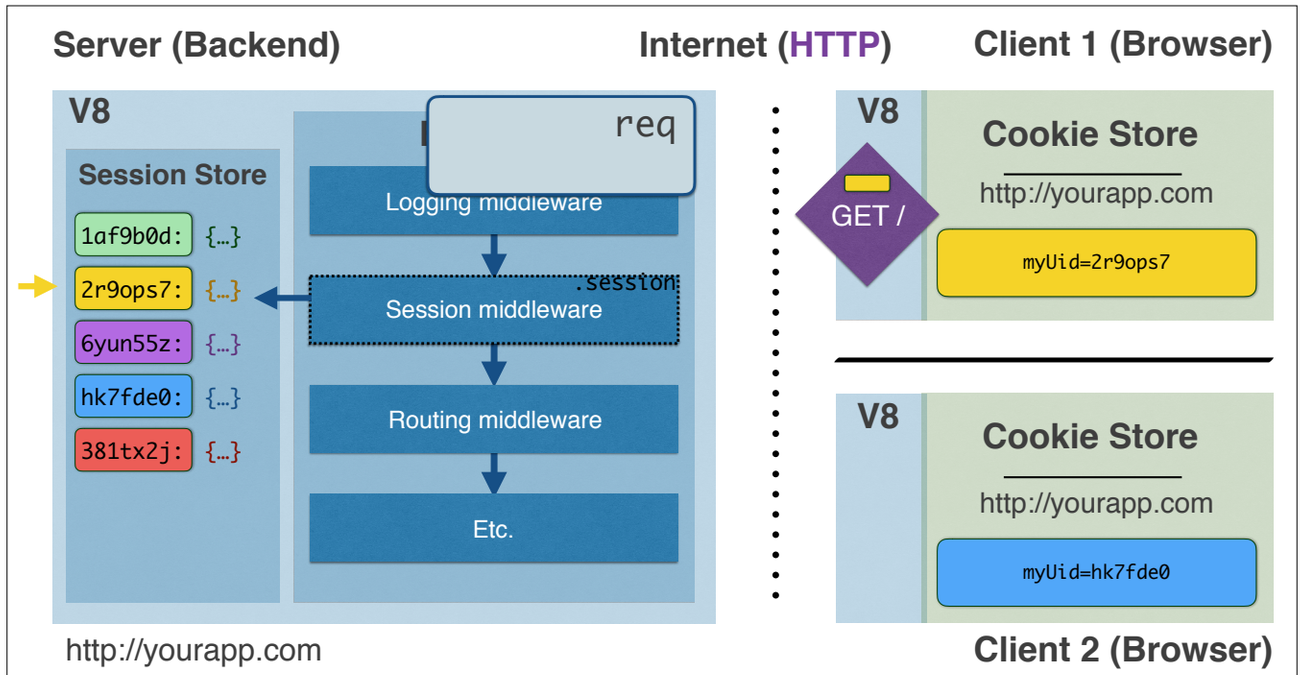
Now at this point, our session middleware will be like - sweet, there's a cookie. And what's that…yes, there's a session id. I'm going to look you up in my session store.

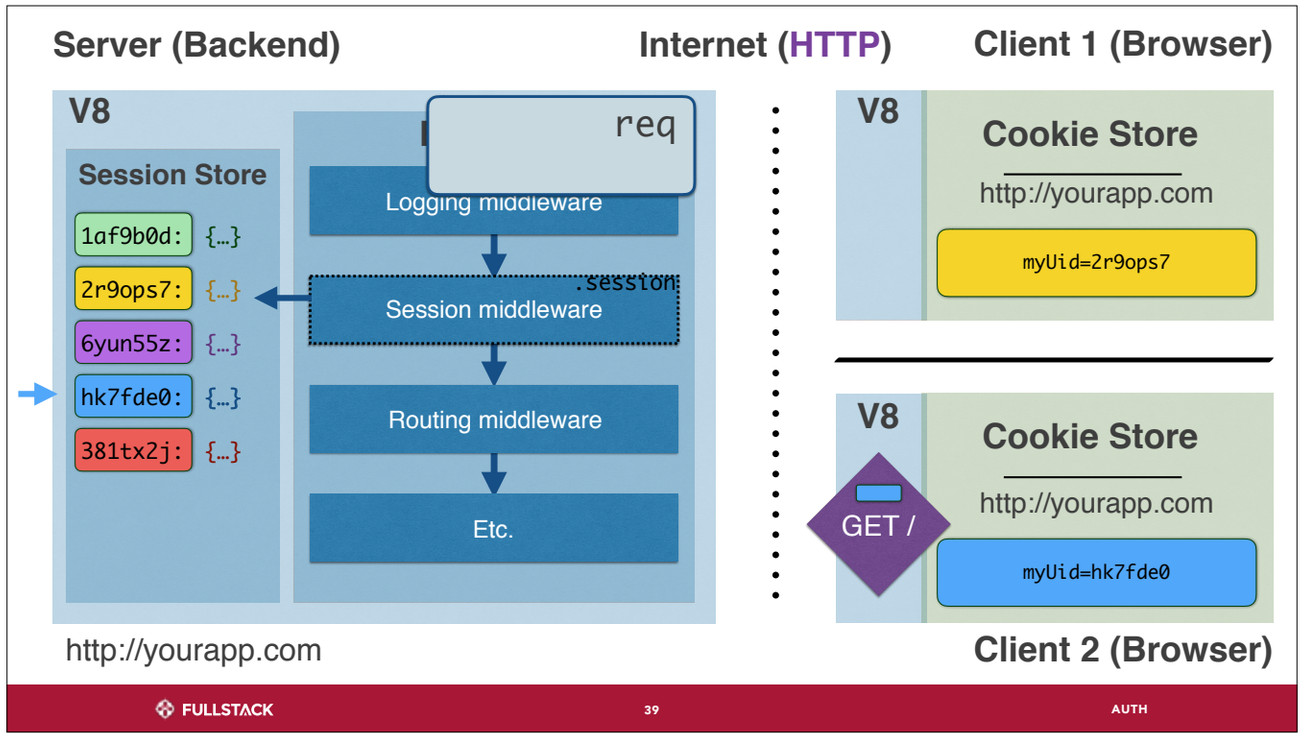

Server (Backend)
Internet (HTTP)
Client (Browser)
V8
Express
Session Store
1af9b0d:
{…}
Logging middleware
GET /
req
.session
Session middleware
Routing middleware
Etc.
http://yourapp.com
V8
Cookie Store
http://yourapp.com
myUid=1af9b0d
http://github.com
landingPage=subscriptions
http://coolsite.com
theme=dark


Server (Backend)
Internet (HTTP)
Client (Browser)
V8
Session Store
1af9b0d:
{…}
Express
Logging middleware
res
GET /
req
.session
{…}
Routing middleware
Etc.
http://yourapp.com
V8
Cookie Store
http://yourapp.com
myUid=1af9b0d
http://github.com
landingPage=subscriptions
http://coolsite.com
theme=dark

Multiple Sessions

Let’s look at how this works when we have many different browsers making requests - they each get their own session and session id


Server (Backend)
Internet (HTTP)
Client 1 (Browser)
V8
Session Store
1af9b0d: {…}
2r9ops7: {…}
6yun55z: {…}
hk7fde0: {…}
381tx2j: {…}
req
Logging middleware
.session
Session middleware
Routing middleware
Etc.
http://yourapp.com
GET /
V8
Cookie Store
http://yourapp.com
myUid=2r9ops7
V8
Cookie Store
http://yourapp.com
myUid=hk7fde0
Client 2 (Browser)


Server (Backend)
Internet (HTTP)
Client 1 (Browser)
V8
Session Store
1af9b0d: {…}
2r9ops7: {…}
6yun55z: {…}
hk7fde0: {…}
381tx2j: {…}
req
Logging middleware
.session
Session middleware
Routing middleware
Etc.
http://yourapp.com
V8
Cookie Store
http://yourapp.com
myUid=2r9ops7
V8
GET /
Cookie Store
http://yourapp.com
myUid=hk7fde0
Client 2 (Browser)

"You get a session, you get a session, everybody gets a session!"

–NOT OPRAH