AUTHENTICATION / AUTHORIZATION

- Authentication (Auth)
    - Who are you?
- Authorization (Authz)
    - What are you allowed to do?

FACTORS

A way of proving auth/authz

- Something you know
  - passwords, PIN
- Something you have
  - keycards, yubikey, RSA token, cellphone
- Something you are
  - fingerprints, iris, face

2FA is "two factor auth", MFA is "multi-factor (2+) auth"

LOGIN

Authenticates, possibly authorizes

- Username
- Password

Send both. Per security discussion, server will compare hashed password+salt to stored salt+hash for that username.

But then what?

BEYOND STATELESS

Web requests are stateless

How do you let the server know a later request is from someone that has already authenticated?

SESSION ID

How to solve with session id

When user successfully authenticates:

- create a random string (session id)
- on the server:
    - connect the username and authz info with id
        - often a DB entry. For class just keep in memory
    - return this session id to the client
- on the client:
    - send this session id with any later requests

OTHER TOKENS

Session Id is a "token" that by itself is random

Other tokens contain usable info directly, but are "signed" to prove who created them

Example: JWT (JSON Web Token) ("jot")

Basically a text file saying "Whoever has this token has proven themselves to be X or be allowed to Y", with a digital signature to prove it came from the owner of some private key.

JWT

Advantages

- No DB check each time used
- Can be passed to others
    - This is how many 3rd party login systems work

Disadvantages

- Good for their lifetime, even if user "logs out"
- Don't want to store changing info in them

SENDING AUTH TOKEN BACK AND FORTH

- Cookies
- Forms
- Headers

SENDING AUTH TOKEN BACK AND FORTH

- Cookies
  - header of a request/response
  - connected to a domain
  - server can "set" in a response
  - stored in browser
  - browsers auto-send on later requests
  - Works across tabs/browser windows
    - Not across profiles/incognito/private
  - Can have an automatic expiration time
  - Can resume after closing window/browser

SENDING AUTH TOKEN BACK AND FORTH

- Forms
    - send as hidden field for body
    - send as query param in url
    - Only in same tab
    - Must be set up for each request
    - Query params impact sharing links
        - Query params not even slightly secure

SENDING AUTH TOKEN BACK AND FORTH

- Non-cookie header
    - Requires front-end JS to send the request
    - ...not a page load
    - Allows you to send custom headers and header values
    - Must be set up for reach request
    - Only on same tab (in-memory JS must know the value)
        - Can store in browser to share across tabs

COOKIE DETAILS

- "Remember me"?
  - sets cookie to not expire/after a long time
  - Otherwise cookie gone when browser exits
- Logout
  - Server can overwrite
- Other cookie config options
  - A path root (rare)
  - Secure - HTTPS only r(ecommended)
  - HttpOnly - FE JS can't see (confusing name)
  - SameSite - 3rd party pages (recommended)

EXPRESS COOKIE EXAMPLE

```
// express "middleware", this time as an extra library
const cookieParser = require('cookie-parser');
app.use(cookieParser()); // use with all requests

// (skipping over other express stuff)
app.get('/', (req, res) => {
  const store = req.query.store;
  if(store) {
    res.cookie('saved', store, { maxAge: 15000 });
  }

  const saw = req.cookies.saved;
  res.send(`<p>Request had cookie "saved": ${saw}</p>`);
});
```

STEPS

1. Inside a new project directory:
    - `npm init -y`
    - `npm install express`
    - `npm install cookie-parser`
2. Create the `server.js` (or whatever you call it) file
3. run `node server.js`
4. go to `localhost:3000` in the browser
5. use `?store=SOMEVAL` at end of url to set the cookie
6. DevTools-Network-Headers to see the `Set-Cookie` in the response and the `Cookie` in the request
7. DevTools-Application-Cookies (left) to see cookies
8. Remember this is a 15 sec expiration time

CHANGING THE COOKIE EXAMPLE

Do you know how to:

- Store the cookie under a different name than `"saved"`?
- Change the expiration time of the cookie?
- Change the name of the query param you are sending to set the cookie value? (instead of `"store"`)
- Redirect the user to `'/'` (no query param) after setting the cookie?