DOM (Document Object Model)

* Documents object DOM is the tree of elements making up a web page
  + Connected nodes of the web page
* JS exists to manipulate the dom.
* This manipulation allow us to make web pages dynamic and responsive.
  + With just html CSS pages can’t really respond to users

AJAX

* Asynchronous JavaScript and XML
  + Almost no one uses this for xml anymore. Predates JSON.
* JavaScript is an asynchronous event drivel language.
* JS is single threaded and can only compute 1 thing at time.
  + If a piece of JS takes a while your application cannot do anything else.
* HTTP Requests are something that can take time.
  + Most processing of code is in the order of nanoseconds.
  + Getting a file or message from another computer on the other side of the world could take several seconds.
    - We do not want our JS to stop working while this is happening.
* Async await and promises
  + Syntactic sugar (abstraction of more complicated callbacks in my opinion)
  + Tell JS to delay processing something until is get the information from the back.

CORS

* Cross Origin Resource Sharing
* It is very, very easy to make JS just in a loop make an infinite amount of http requests to a server.
* It is also very easy to make malicious JS that could make requests that read your cookies and send them to hackers.
* The web has a standard that JS CANNOT make http requests and have them be processed unless the Server allows that to happen.
  + Servers can specify what origins (ip addresses) are allowed to send JS http requests to them.

Web Security

* People have access to what they should and do not have access to what they should not.
  + The main principle of security.
* All security should be a holistic approach.
  + There is no one perfect solution.
  + You should add security throughout your application.
* JWTs are one way in which we can add security to our application.
  + JSON Web Tokens.
  + JWTs are essentially unchangeable documents you can give to users.
    - When you give someone a JWT it is impossible for that person to edit the JWT without you noticing.
    - This is the feature that gives JWTs the ability to be used for security.
  + JWTs
    - Are NOT ENCRYTED
    - Must be kept private for each user.
    - Think of JWTs as a government document like a Social Security card.
      * The Server can tell if you edited it or if it is fake.
      * It cannot tell if it was stolen.
* JWTs are encoded
  + Encoded means turned into another format.
    - Turing a Java object into JSON is ‘encoding’ the object into JSON format
    - ANYONE could read an encode JWT
    - JWTs are encoded in base64 text
      * + Done purely for utility reasons
* A JWT is comprised of 3 parts
  + 1st part header
    - Meta information about the JWT
  + 2nd part body/payload
    - Data that is in the JWT
    - Should be in JSON format
  + 3rd part signature
    - How the magic of an un-editable document is created.
    - The signature is created by using the payload body and combing it in an algorithm to create a signature that cannot be reverse engineered.

How to give a JWT?

* There are a few approaches.
* My flow for giving a JWT to a person.
  + The client makes a login request ex… /users/login
    - In the body you put the username and password
  + The server will generate an appropriate JWT for that user.
    - Maybe includes their role or username.
      * NEVER THEIR PASSWORD
  + The HttpResponse will give back that JWT as a cookie.
* A cookie is a small text file stored in a browser.
  + Often used to store user information.