How do we persist session data and keep them logged in?

1. Cookie
   1. Small string (4kb) stored in the browser.
   2. Browser can store ~50 per domain.
   3. Browser can store at most ~3000 cookies total.
   4. With each request/response cycle, **all** cookies are sent.
   5. They are stored in the header request object.
   6. Server sends a “set-cookie” header in its response
      1. If you send 4 cookies,
      2. The server might add another cookie saying (server was visited)
      3. And now 5 cookies are sent back.
2. What do cookies look like on the client?
   1. Formatted as a series of k-v pairs on request and response headers
   2. Semicolon separates each pair
   3. Cookies typically store encrypted info that is used to keep track of who you are.
3. Types of cookies
   1. Persistent cookies (aka tracking cookies)
   2. Secure (can only be transferred over HTTPS)
   3. HttpOnly (can only be transferred via HTTP and HTTPS, **UNABLE TO BE ACCESSED BY JAVASCRIPT)**
   4. If session ID is stolen, that session ID can be put into a response header, which would enable a user to impersonate another.
4. Cookie metadata
   1. “expires” is deprecated
   2. “Max-Age” is what we’re looking for as far as how long the cookie should persist.
   3. Generally, we want to use http only.
5. Cookies for authentication:
   1. When a user logs in, we create a session cookie, which is a unique id associated with the user.
      1. The cookie is saved to the browser
      2. We can also save cookie values in our database and have it correspond to a user.
   2. The next time a user visits the website, it will use the session cookie to identify the user and persist the session.
      1. “SSID” – Session ID.

*React Portals – something cool to look at*

1. Seting cookies:
   1. In express: make sure you install cookie-parser
   2. And make sure you encode it (like app.use(json()))
   3. Res.cookie -> creates cookie
2. Cookies have other uses other than authentication.
   1. It can be used for things like
      1. Identifying recurring users
      2. Maintaining user settings info
      3. Advertising
   2. Third-party advertising cookies
      1. Accepting all cookies will track activity across websites.
      2. Iframe is an html element that allows another html doc to be embedded.
      3. Websites contain iframes that belong to third-party adverts.
      4. When a browser loads a third-party iframe, it sends an additional request to that party
      5. And the cookie we sent with that request is used to identify you and to serve you personalized advertisements.
      6. So when these iFrames, which is essentially real estate purchased by other companies, are loaded:
         1. A get request will be made for that content (even if the iFrame is set to hidden)
         2. And thus your cookies will be sent to that third party and you may be identified.
3. Cookie theft
   1. Cookies are freely editable
   2. They are not a source you can explicitly trust
   3. Attackers can steal cookies from a user’s browser, which can be used to impersonate the user if an active cookie session has been stolen.
4. Network Eavesdropping
   1. Network traffic is intercepted and a request is received by a 3rd party
   2. Use secure connections (https)
5. HTML5 LocalStorage
   1. An alternative to cookies
   2. A mini-database
   3. Arbitrary k-v store for the browser
   4. localStorage API
   5. window.localStorage.setItem(‘key’, ‘value’)
   6. window.localStorage.getItem(‘key’) => ‘value’
   7. we do NOT want to store sensitive data here!!
   8. It can be easily read or written.
   9. This can be done in either front-end or back-end.
6. Cookies vs LocalStorage
   1. localStorage has no built-in security
   2. and it will last forever, unless you remove it.
7. sessionStorage
   1. only lasts for the session.
8. JSON Web Token
   1. JWT (jawts)
   2. JWTs allow us to avoid continually querying a database to authorize users.
   3. Token authentication is an approach used in place of session authetnciation.
   4. Tokens are generated for users
   5. And they store the user’s identifying info on the client side rather than server side.
   6. JWT is an open standard that defines a compact, self-contained way for transmitting info between parties as JSON object.
      1. Payload
         1. Info about who this is.
         2. The JWT is checked to identify the user.
         3. But this payload is easy to fake – how do we ensure that it’s not fake?
      2. Signature
         1. We use a signature to ensure the payload is authentic.
         2. This is a one-way hash of the payload
         3. Plus a **secret** stored on the server.
         4. We take the payload + secret and hash the resulting string
         5. This ensures that a forged payload will be read differently from an authentic one.
      3. The header indicates what hashing algorithm to use
      4. The payload is our data
      5. The signature is our salt.
         1. Payload and Header are encoded into base-64.
         2. A “secret” is added to the encoded string
         3. And this result is hashed.
   7. Cookies can be used to store JWTs.
   8. We can add HTTPOnly and Secure options to your cookie, you can block it from access by JS.
   9. This approach is limited to one domain, but it can help avoid XSS attacks.