# SPA Details

Compared to server-generated dynamic pages, single-page applications (SPA) have:

- One core concept to keep in mind
- Details about how to handle common needs

# SPA Core Concept

The Core Concept:

- SPA: updates page content without a page load
- A non-SPA: a page load on any server update

Lots of middle ground and mixing

- Can be a few "pages" but many "screens"

# Page Loads vs DOM Manipulation

```
<form action="/foo" method="POST">
Word: <input name="something">
<button>Submit</button>
</form>
```

- Causes a **page load** on `/foo`
- Sends params based on input `name` attributes
- Sends params as url-encoded string (`something=some%20value`)

# DOM Manipulation

```
fetch('/foo',  {
  method: 'POST',
  body: JSON.stringify({ something: somevalue })
});
```

- loads data from `/foo` in **background**
- doesn't require `<form>`
- doesn't use `name` attributes
- no default body syntax (JSON is one option)
    - Should send header to indicate content type

# SPA Details

Common questions to answer in SPA:

- using Progressive Enhancement?
- indicating wait (spinners)
- doing login/logout via service calls?

# Progressive Enhancement

Taking a non-client-side JS web app and augmenting it with JS

- Remains working if no JS (no client-side JS)
- Great for search engines
- Great for accessibility and various devices
- Great for ensuring backend is secure (no assumptions)
- Fairly rare due to extra effort

# Techniques

PE techniques include:

- Form validation before submit
- Autocomplete
- Form submission hijacking
- Pulling in functionality from other pages

# How to Progressively Enhance

- If no JS, page works using form submits
- If JS, add to/replace/turn-off/override DOM to use JS instead/also

Example:

- A form submission sends to backend, gets new page
- JS turns off submission, sends as background call and replaces form once sent

# Biggest Lie in Web Apps

- Add to page before starting a long async action
- Remove when complete
- If something breaks and you don't remove it
- ...it keeps spinning
- ...does NOT indicate anything is "thinking".
- It is just an animated image

# Sessions

Web is stateless

Login creates a "session" on the server

- usually stored in DB

Session needs to expire

- and be cleaned up (DB?)

Session is NOT your info

- that is tied to your userid

# Polling

The web request/response cycle:

- means the client has to ASK for an update
- ...even if there isn't one yet

This can feel (and be) inefficient

- But is also very common
- We'll do basic polling because it's simple
- ...not because it is better

# Polling methods

- Polling
  - periodic web requests
- "Long Polling"
  - Server keeps res open, trickling empty data
  - Server finishes res once there is an update
  - Client immediately opens new request
- Websockets
  - Not HTTP
  - A different protocol started *from* HTTP
  - Allows server "push" actions

# CRUD

- Create
- Read
- Update
- Delete

Basic interactions with records (usually a database)

Majority of Webapps revolve around this functionality

# Some Tips

- Remember to `preventDefault` on
    - form submissions
    - button clicks
    - link navigation
- Disable/enable buttons
- Tooltips on hover

# More Tips

- Modal windows
    - full page div
    - translucent background
    - form in div
    - stop event propagation
- Remove/hide elements that the JS makes redundant/unhelpful