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=somevalue`)

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

Multiple Page Application (MPA)

Multiple "pages"

- Distinct urls
- Some dynamic, some static
- Navigate between pages
    - request/response/full-page render

Single Page Application (SPA)

- One base url
    - "distinct urls" requires server configuration
        - passes through same dynamic handler
    - can be done with one `.html` file!
- Different "views"/"screens"
    - no full-page renders
    - changed HTML via front end JS
- Web Service calls (AJAX) to save/read from server
    - Page state (in JS) remains

In Between Apps

- No hard requirements
- Loading time of JS data and page rerenders
- Backend vs Frontend regularly swings back and forth
    - Currently shifting away from Frontend peak
    - Server Side Rendering (SSR) and Server Side Generation (SSG)

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

Some Tips

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

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

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

Simple polling

- use `setTimeout()` or `setInterval()`
  - call a callback that does `fetch()` every N milliseconds
- save the id of any interval to cancel it later
  - "id" here is unrelated to html id

Summary - Concepts

- Front End JS vs Server-side JS
- MPA vs SPA and points between
- Progressive Enhancement
    - more work, more accessible
- Indicate delay with spinners
- basic polling, long polling, websockets