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

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

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

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

MORE TIPS

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