TAKEAWAYS

- Page Loads vs DOM Manipulation
- Progressive Enhancement
- Error Handling

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`)

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

- loads data from `/foo` in **background**
- doesn't require `<form>`
- doesn't use `name` attributes

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

TIPS

- Remember to `preventDefault` on
  - form submissions
  - button clicks
  - link navigation
- Disable/enable buttons
- tooltips on hover
- modal windows
  - full page div
  - translucent background
  - form in div
  - stop even propagation
- Remove/hide elements that the JS makes redundant/unhelpful

ERROR HANDLING

`throw new Error("poop")` was intended to have you fill it in

- Not copy it and leave it there

Give the user useful information!

Remember the difference between

- a network error (fetch will reject promise)
- server error (fetch will resolve with response, `response.ok` is false)