

Start of Class Prep

In a directory where you would create projects for labs

Run this:

```
npx create-react-app test-app
```

This will take a few minutes

- Start it up and we'll look at it later in class

JS-Chat

- RESTful Service-based backend
- "Vanilla JS" Single Page Application
- Last backend-focused lesson of the course
 - You can do everything for backend of final!
 - You ARE working on your final, right?

Pain Points to Learn from

- Defining and transferring state
 - Backend vs payloads vs frontend(s)
- Polling
 - What service to poll?
 - What to do with results?
- Rendering
 - Scrolling
 - Interruption
- Estimation
 - Will not magically get better

State Complexity

- Every App benefits from thinking about state
 - Data Models
- Apps have different needs!
 - Similar data, may be different states
- Data sent may not be the same as data stored
 - Ex: Array of active users vs Object of users

Multiple Clients

We wrote FE and BE for this app

- But many services have *multiple* clients!
- Service should be "kind" to clients
 - But clients can have different needs!
 - Don't tightly couple service and client
- We move to React for FE starting today
 - But services don't change!
 - Just a new client

Service devs think in terms of *usable data*

- Not specific app consumer

What to return on login?

- Check for existing session on login
 - What to return?
- Logging in to create a session
 - What to return?

Should you assume to send all the needed data?

- Or should they need to make extra requests?
- More requests = extra delay
- Send all data = complex to add more services

Generally, better to separate login from data

Users vs Messages

"active users" and "messages"

- 1 service or 2 services?

Answer is surprisingly nuanced

- Is this data coupled for all clients?
 - ALL clients?

Service Granularity

More services (and more granular) good because

- More flexible for clients
- Easier to add changes over time
 - Ex: Pagination

More services (and more granular) bad because

- More network overhead
- More work if we always call them all anyway

Data Model Assumptions

A service sends named, known fields

- Will it add fields in future?
- Services want to avoid version bump
 - Often add new object fields
 - New feature = *Minor* version
 - Which means all clients get immediately
 - Write your clients safely!

Security

- We didn't filter the message text
 - Vulnerable to injection attack!
 - Hard to allowlist open text
 - Not limited like username/phone
 - HTML Entities are a mess
 - `<`, `>`, and `&` in particular

Security Options for Plain Text

- Option 1: Ban entities
 - Works great for security
 - Users may wish to use `<`, `>`, or `&` in text
- Option 2: Translate Entities when stored
 - Convert to `<`, `>`, `&`
 - Data is stored safely
 - Requires conversion to use outside of HTML
 - Causes problems if reconverted (`& &`)
- Option 3: Translate Entities on display
 - Data is stored unsafely!
 - Everyone. Must. Remember. When. Using.

Polling

- Surprise difficulty for many!
- What service do you poll?
 - What does it return?
 - How does it know?
- What do you do with results?

Think about how a service dev would think

- WITHOUT a specific client

Most Basic Option

- Poll service to get all messages
 - Always replace list of messages
 - Rerender

Basic Pro Cons

Pro:

- Very simple to implement
 - Like a refresh button, but automatic
- Already have "get all messages/users" service(s)

Cons:

- Rerender even without change
 - Rendering issues covered separately
- Lots of network overhead (all messages)
 - What if big data?
 - What if pagination?

Medium Complexity Option

- Poll a "are there new messages" service
 - Use results (one of:)
 - Display an indicator to refresh
 - OR, Request message list and rerender
- Requires a "new messages check" service
 - Requires a way to identify new messages!
 - Messages get ids?
 - Timestamp?

New Message Check Benefits/Costs

- Separates complexity of IF new messages
 - from HANDLING of new messages
- New Service
 - New functionality to offer client flexibility!
 - New commitment to client devs!
 - New maintenance costs!
- If (*IF*) client always requests new messages
 - Why force another service call?

Advanced Messages Option

- Poll a "get new messages" service
- Same requirement to identify if new messages
- Returns only new messages
- Can be same service as "get all messages"
 - With optional param
 - indicates most recent known message

Advanced Get Messages

- No new services, just new features
 - Fewer services means controlled commitments
 - More complex services means more complex code
- Adds flexibility for clients
- Reduces unnecessary service calls

Polling Active Users

What about changes in active users?

- In same service as messages?
 - See previous discussion
- Similar, separate service for users vs messages?
 - Double the polling?
- Clients refresh users when new messages?
 - Reasonable(?)
 - Clients may want more

Rendering Results of Polling

Surprising issues!

- Showing newest messages?
 - Scrolling is complicated
- Auto-rendering can interrupt

Showing Newest Messages

How to show newest messages if too many for screen?

- Option 1: New messages on top
 - Clever "cheat"
 - Can't always do so
 - "Later" usually below "Earlier"
- Option 2: Scroll to bottom
 - Scrolling is complex!
 - Takes time and testing!
 - Keep Accessibility in mind
 - Research!
 - "Works for me" isn't enough

Rendering can interrupt

- We replace most/all of the HTML on rerender
 - Including form elements!
 - Even mid-typing
- Fine for this assignment
 - But not what you'd want for a real project

Avoiding Interruption

- Solution is doable, but lots of details
 - Need to be more nuanced in replacing HTML
 - We will let React do that for us
 - But you know enough to do it without!
 - Detailed
 - Time-consuming to write/test

Estimation

- I got a LOT of requests for extension
- Some of you have been warned
 - Point is to prepare for working
 - Normal to occasionally be behind
 - Not good to always need more time
 - Not good to notice too late to matter
 - Workplaces often change non-deadline parts
 - You need time to make changes

Are you learning from Estimation Problems?

I'm gracious (and humble!)

- Teams, clients, managers may not be
- Other Instructors may not be

Learn from my generosity so you don't need theirs

Final Project has no non-emergency extensions