

REACT.JS

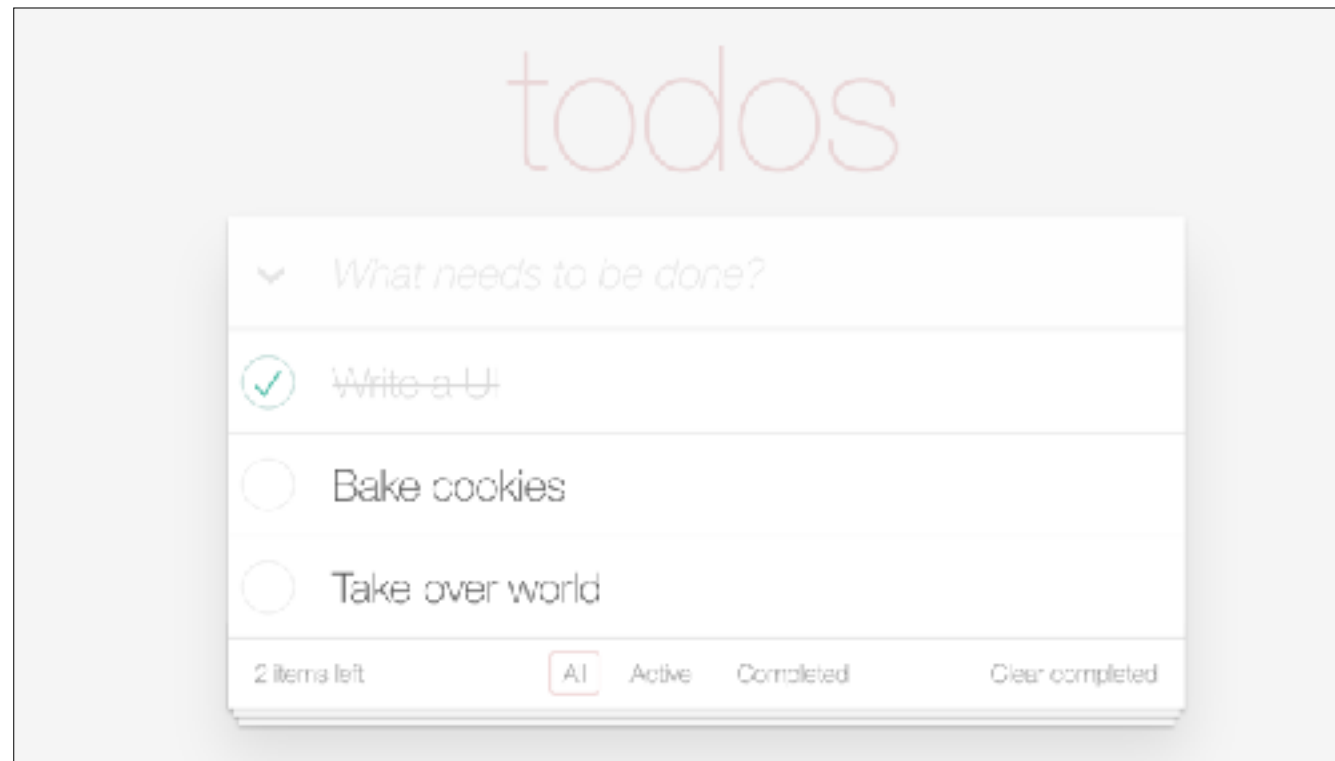
A JavaScript library for building user interfaces

PROBLEM: MAKING A USER INTERFACE IS HARD

WHY: SO MUCH STUFF TO REMEMBER!

STATE + VIEW

- **Every user interface has data (which we often call “state”), and presents something based on that data**
- **Any change to the underlying data should be reflected in the view**



What is the “data” in this app

- list of todos
 - whether a todo is active or completed
- whether we’re showing all, active, or completed todos
- number of items (2 items left)
- even the text that’s entered in the input

```
{
  "todos": [
    {"text": "Write a UI", "status": "completed"},
    {"text": "Bake cookies", "status": "active"},
    {"text": "Take over world", "status": "active"}
  ],
  "showStatus": "all"
}
```

Here is how we could represent this app as data, or “state”
(total items is the length of todos after we filter for just the active)

Question: how to represent this data in the DOM?

ADD A TODO

- **Push to our “todos” array**
- **Append a new element to the DOM**

WAIT

- **Push to our “todos” array**
- **Does the “status” of the todo match what we’re currently showing?**
 - **Yes: Append a new element to the DOM**
 - **No: Okay, don’t append a new element to the DOM**

WAIT

- Push to our “todos” array
- Does the “status” of the todo match what we’re currently showing?
 - Yes: Append a new element to the DOM
 - No: Okay, don’t append a new element to the DOM

Let’s say that we’re currently only showing completed, and we add a new, active todo

OKAY, NOW WE SWITCH BACK TO ALL

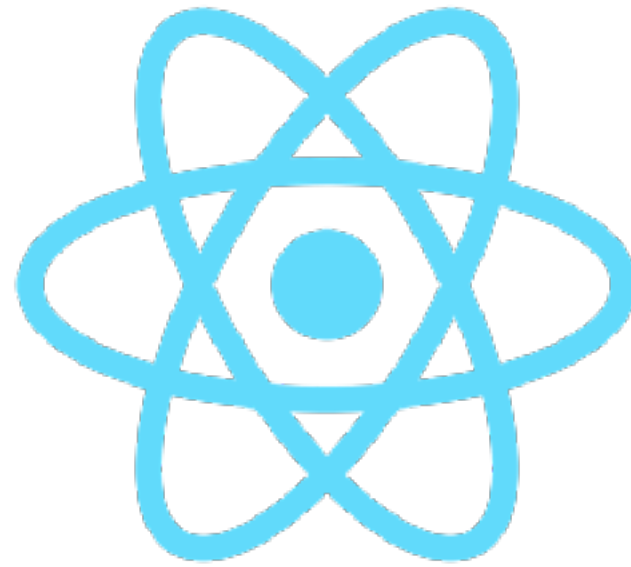
- Change the “showStatus” to “all”
- Check our list of “todos” and add any that aren’t showing back in
 - But make sure they’re in the right order!
 - Don’t just wipe the slate clean though - that’s bad for performance!
 - Also make sure to change the highlighting so that “All” looks selected

**OKAY, SO REMEMBERING TO
CHANGE THE VIEW AFTER WE
CHANGE THE DATA IS HARD...**

**OKAY, SO WHAT IF WE COULD
JUST CHANGE THE DATA, AND
THE VIEW WOULD...**

...REACT?





REACT

- **Data is called “state”**
 - More precisely: “state” is data that *changes*
- **When you update “state”, React re-renders the view for you**
- **React re-renders the view in a performant way**

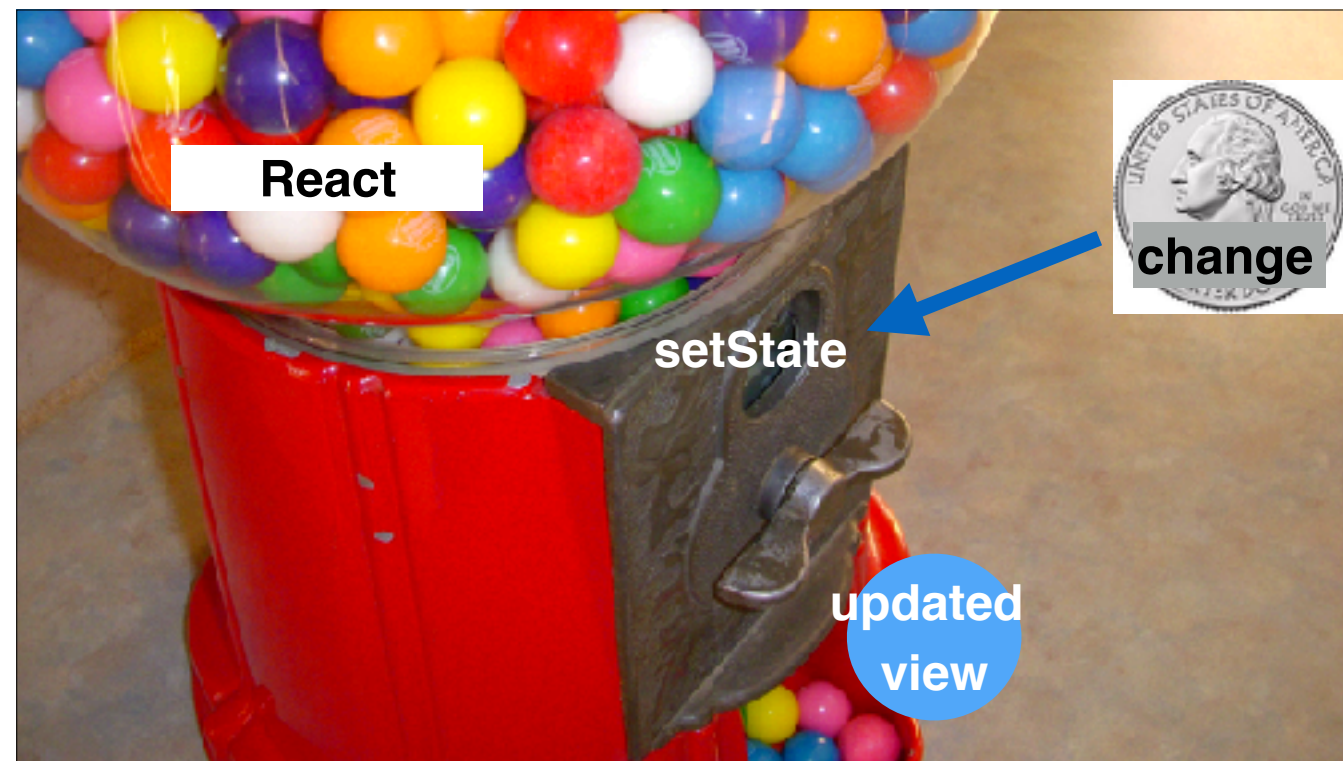
SET STATE

- **We do not directly mutate “state”**
 - **Ex: we DO NOT say** `state.showStatus = “all”`
- **We use a method called** `setState`
 - **Ex. we DO say** `this.setState({showStatus: “all”})`
- **Why? So that React can handle things intelligently if multiple updates happen all at once!**

Think of it as “requesting an update” from React, as opposed to trying to force it.

Other reasons:

- single gateway easier than many gateways
- immutable data easier to reason about
- batch updates



WHEN YOU SET STATE, REACT WILL UPDATE THE VIEW

And this is the **only** way for the view to change



LET'S WRITE YOUR FIRST COMPONENT!

