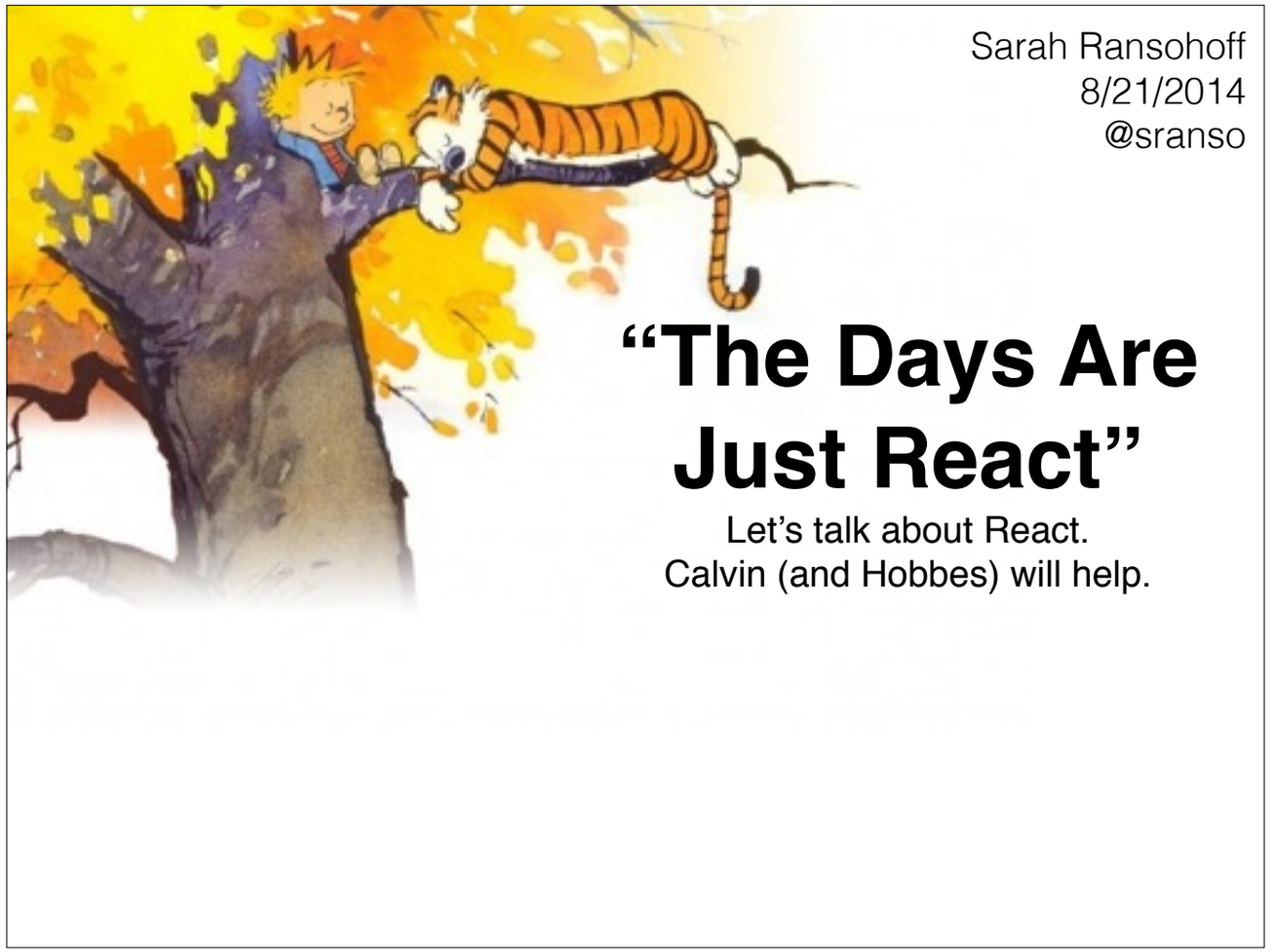
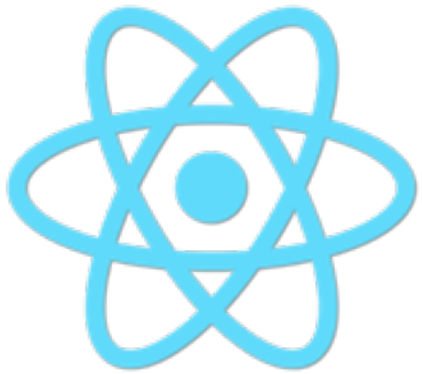


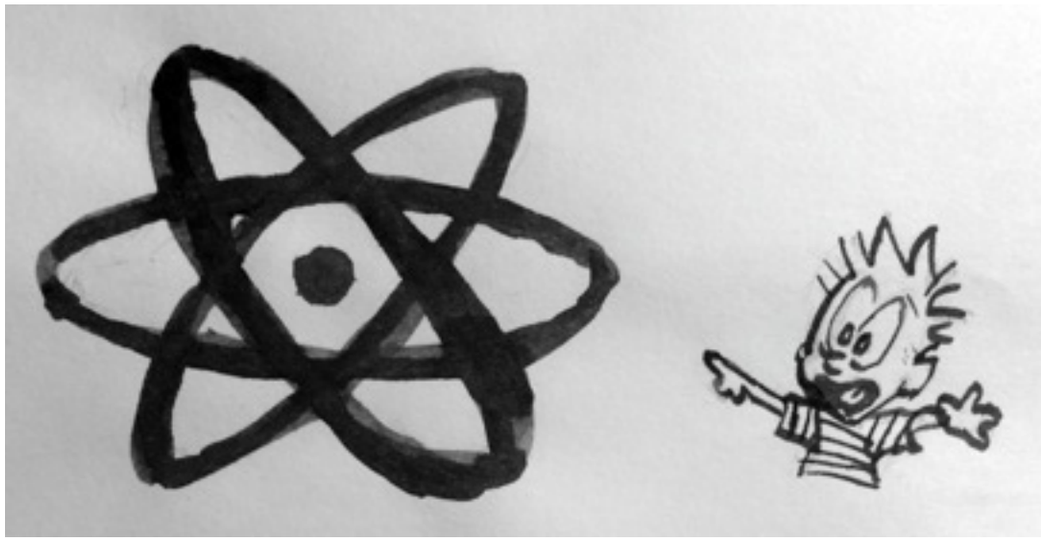
hey everyone i’m sarah, graduated from flatiron, work on web team at a venmo. we’re gonna talk about react and calvin is gonna help us understand what’s going on

WHERE WE'RE GOING

1. React
2. Virtual DOM Diff
3. Components
4. Data Flow
5. Recap



WHAT IS REACT



WHAT IS REACT

a JavaScript library for building
user interfaces

(user interface: the thing(s) a user sees, clicks on, drags, etc in a web application)

WHY USE REACT

there are lots of JS UI interface options —
why use React?


more…

why use react over the other options?

- angular, webix, backbone

WHY USE REACT

- chunks of reusable UI
- large modular apps
- easy to follow data flow (one-way data flow)

enables you to make chunks of UI (components)

HOW DOES IT WORK

- virtual DOM diff implementation
- components on components
- implements a one-way data flow

- build with components — think of your app in components, then determine the component hierarchy

HOW DOES IT WORK

- virtual DOM diff implementation
- components on components
- implements a one-way data flow

- build with components — think of your app in components, then determine the component hierarchy

HOW DOES IT WORK

- virtual DOM diff implementation
- components on components
- implements a one-way data flow

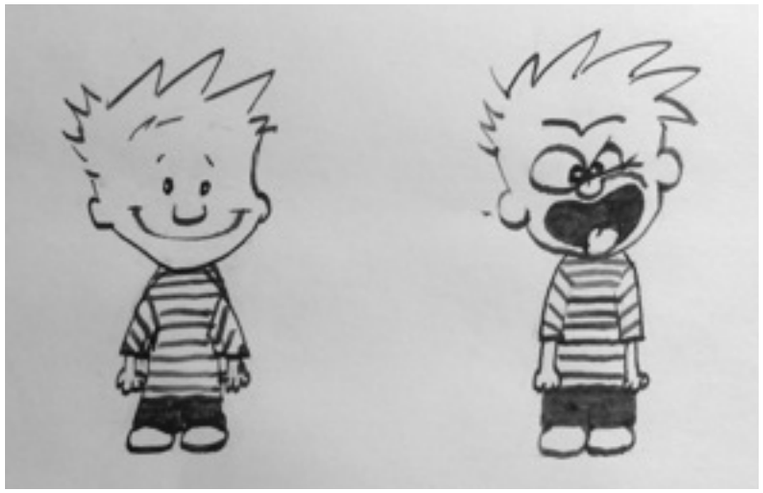
- build with components — think of your app in components, then determine the component hierarchy

VIRTUAL DOM DIFF

- React keeps track of a *Virtual DOM* and a *Previous Virtual DOM*
- when it's time to render…
  - diff = Virtual DOM <—> Previous Virtual DOM
  - update Real DOM with diff

the VDOMD comparison is between the latest virtual DOM and the previous virtual DOM. The actual DOM just gets updated according to what changed between the most recent two virtual DOMs.

VIRTUAL DOM DIFF



PREV. VIRTUAL DOM

VIRTUAL DOM

- let’s have Calvin help us understand this. let’s say Calvin’s doing well, he’s getting along with his mom today. That’s what React has in its VDOM — happy Calvin. But then she offers him some broccoli which he HATES so Calvin’s face changes. React notes this change in the VDOM, compare it to its PVD, and takes that diff, and then updates the RD. So in a way, there is a third Calvin out there (not pictured) who is “the real Calvin”.
-

VIRTUAL DOM DIFF

- remove?

VIRTUAL DOM DIFF

Why use it?

- faster — no re-render of whole DOM
- faster — look to Virtual DOM rather than Real DOM

- faster — don’t have to re-render the entire DOM or even the entire component
- examining/updating the real DOM is very slow, so the virtual DOM allows for speed

HOW DOES IT WORK

- virtual DOM diff implementation
- components on components
- implements a one-way data flow

WHAT ARE COMPONENTS

- elements that are parts of a whole
- chunks of reusable UI

- chunks of reusable UI that work the same way no matter where you (re)use them

WHAT ARE COMPONENTS

- elements that are parts of a whole
- chunks of reusable UI
- *dare I say…*

WHAT ARE COMPONENTS

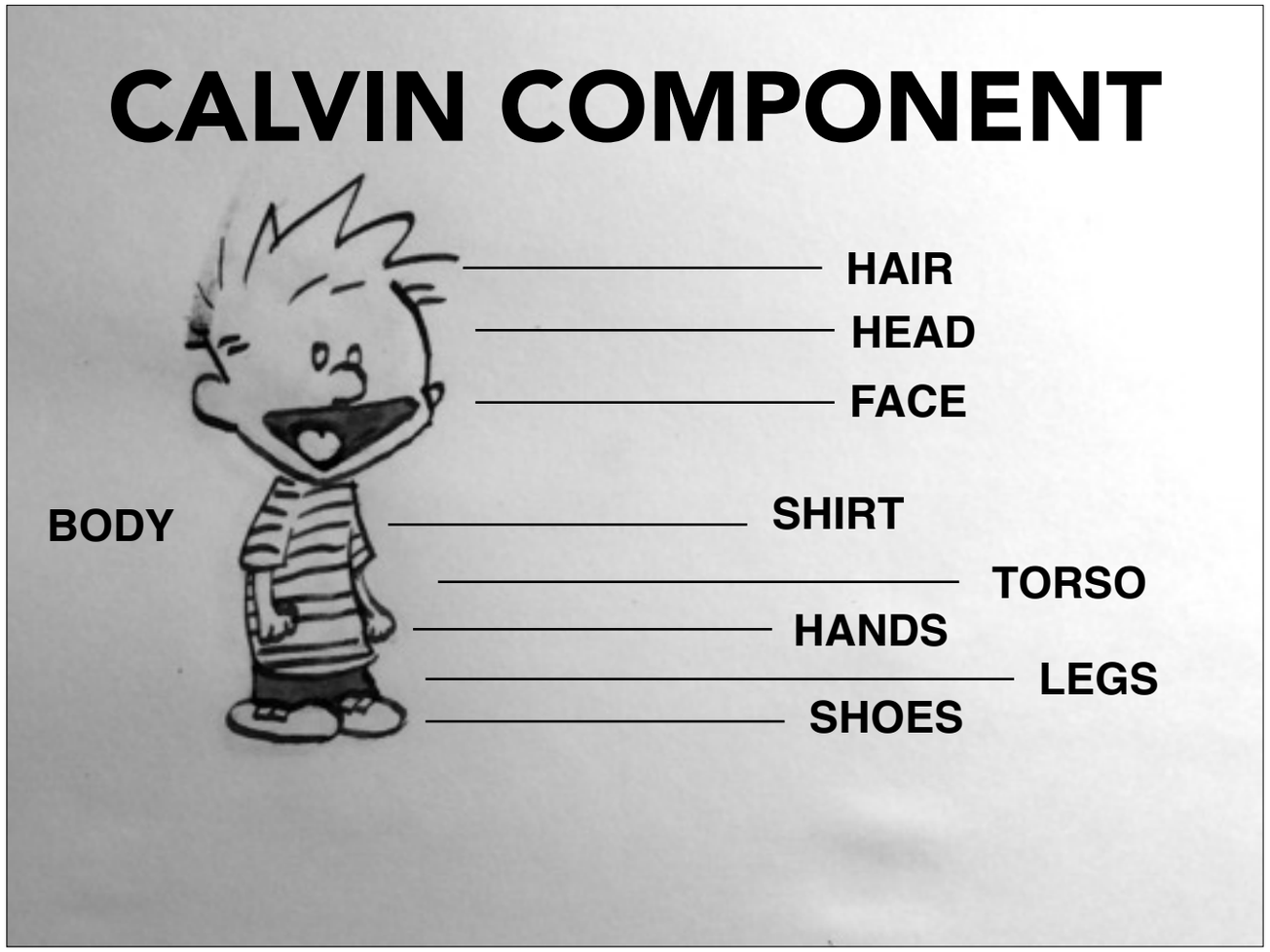
- elements that are parts of a whole
- chunks of reusable UI
- something with a single responsibility!?!!?

WHAT ARE COMPONENTS

- elements that are parts of a whole
- chunks of reusable UI
- something with a single responsibility!?!!?

CALVIN COMPONENT

- calvin is one whole being but made up of diff parts, components. what makes up calvin?
-



- ex of components — e.g. shoes and hands, more than one. React allows us to reuse that component twice. all these components (and more) make up Calvin.
-

- ok we've got our components, what is the hierarchy? which components are parents to the others? in many ways, react is like a tree: there's a larger component that calls other components that calls others etc, passing off necessary data as it goes

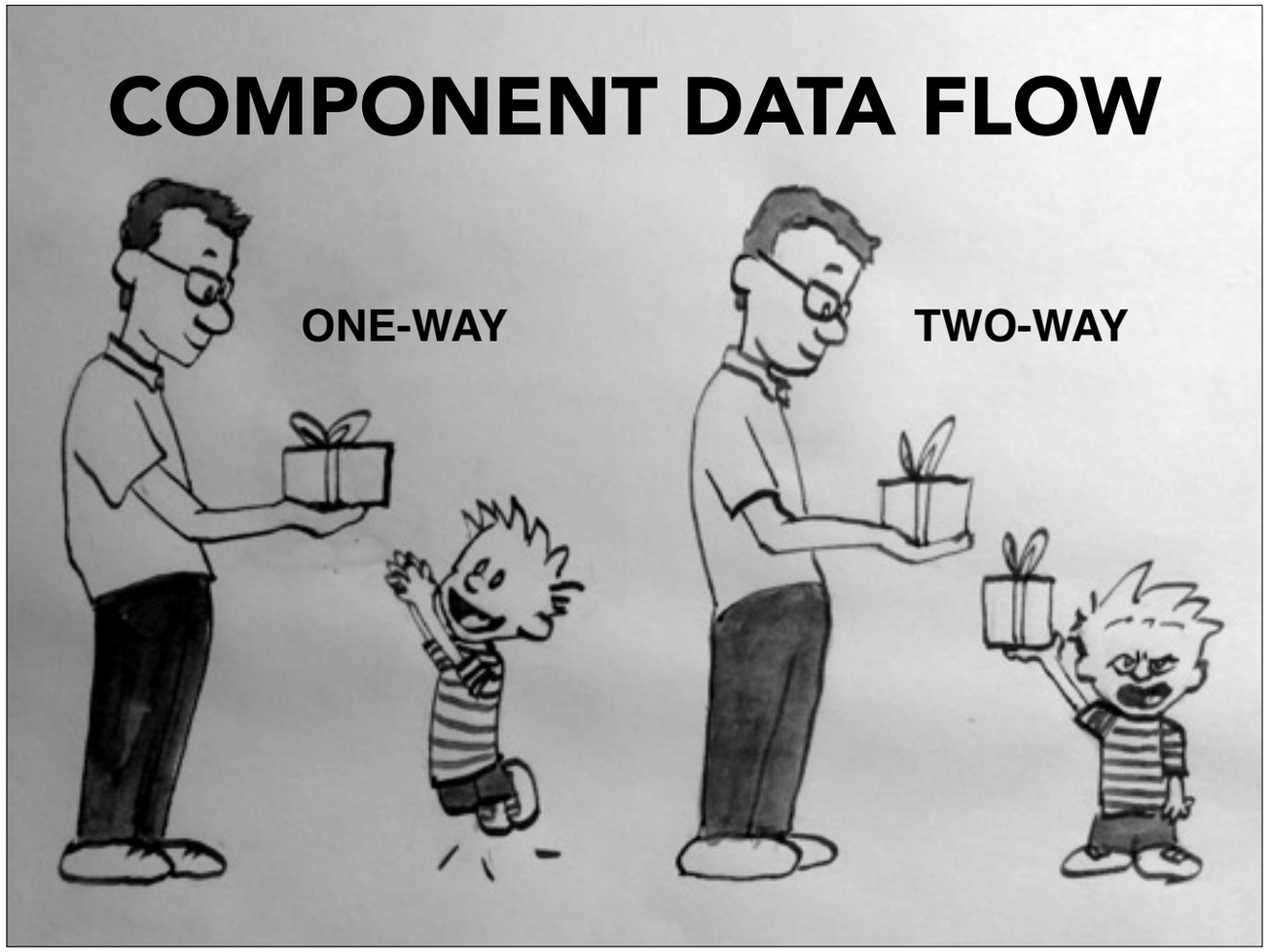
HOW DO COMPONENTS WORK

each component takes an input of data and returns what to display (html)

- which leads me to how they work — each component etc… input data

HOW DOES IT WORK

- virtual DOM diff implementation
- components on components
- implements a one-way data flow



- in react data flows from parent to child — one-way data flow. the other option is two-way data flow — from parent to child, and child to parent. that’s how angular works for example.

COMPONENT DATA FLOW

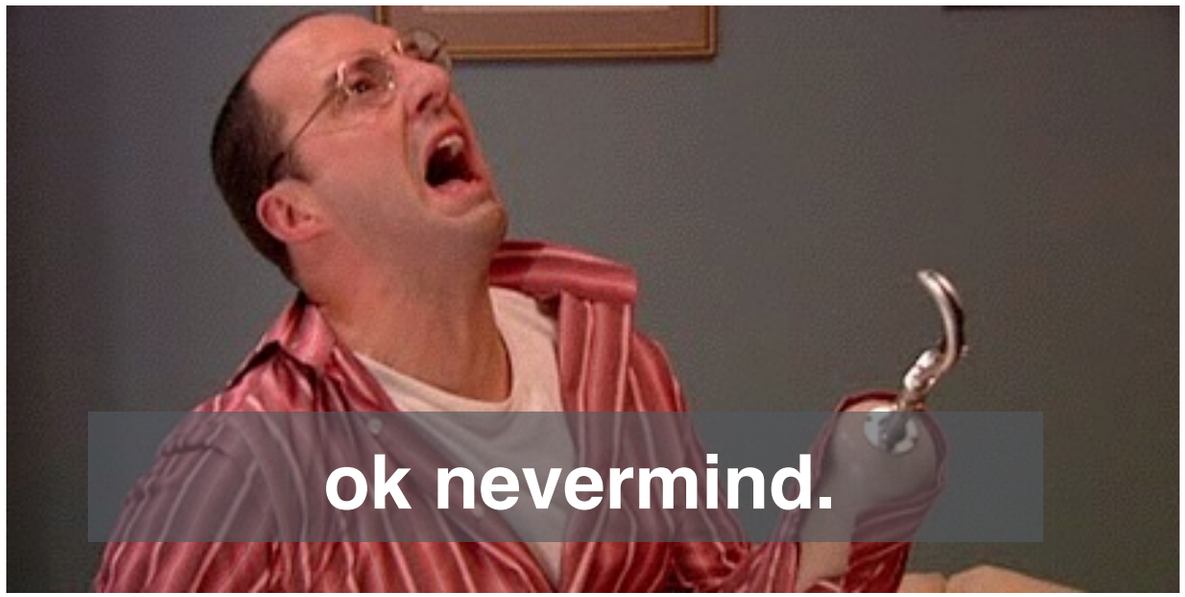
Why use one-way?

- keep things modular
- easier to reuse children if they don't care who their parents are

COMPONENT DATA FLOW

COMPONENT DATA FLOW

COMPONENT DATA FLOW



TWO KINDS OF DATA IN REACT

1. `this.props`
2. `this.state`

color code for next few slides :)

-

this.props.genes(function(){});

this.props

- props - data passed from parent to child. note that children aren’t supposed to change this.props — they technically can but if they do it complicates data flow
-

```
this.setState({plotting: true});
```

this.state

- state - data reserved for interactivity, data that changes over time. this.setState causes a re-render of the react component. so, for ex, if Calvin's expression changes when he's "already rendered" that would cause his face to be re-rendered. nevertheless, state + virtual DOM is still really helpful for that reason — can pinpoint just the thing we want to change
-

COMPONENT EXAMPLE

```
119   React.renderComponent(
120       <Body sitting="false" awake="true" />,
121       document.getElementById('container')
122   );
```

- render into dom. classroom vs snowy hill. at very beginning, body component rendered into DOM with two props: sitting and awake (ignore though).
-

COMPONENT HIERARCHY

```
/**
 * @jsx React.DOM
 */

var Body = React.createClass({
  render: function() {
    return (
      <ul>
        <li>BODY:
          <p>This is his body.</p>
            <Head awake={this.props.awake}/>
        </li>
      </ul>
    );
  }
});
```

```
var Head = React.createClass({
  render: function() {
    return (
      <ul>
        <li>HEAD:
          <p>This is his head.</p>
          <Face awake={this.props.awake}/>
        </li>
      </ul>
    );
  }
});
```

- when you render body, body renders head, head renders face, etc etc. data passed down
-

LIVE EXAMPLE

- let's see an example of event flow. here is the result of a react app — the one we've been looking at. in a list form so we can really emphasize the fact that body is parent to head, head to face, etc. we can see that all components know that calvin is awake b/c that's important to all of them, and that data was passed down as a prop

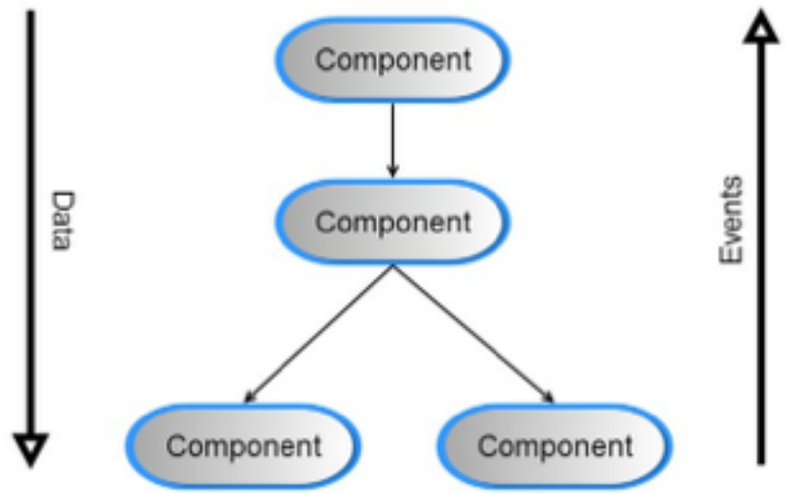
BEGINNER PITFALLS

(aka my pitfalls)

BEGINNER PITFALLS

- always add `/** @jsx React.DOM */` at beginning of js file if you're using jsx — you'll get funky errors if you don't
- each `return` function can only return *one* DOM element
- listener events like `onChange` are often passed up in components — be careful to not to override a listener event. you can avoid this by having a local `handleChange` that calls `this.props.onChange` (looks up to another `onChange`)
- remember React uses one-way data flow! you can't pass data up a tree. pass it on down.

DATA & EVENT FLOW



IN SUM...

- React is a JavaScript library for building user interfaces
- virtual DOM diff implementation
- components on components
- one-way data flow

AND WE USE REACT B/C...

- chunks of reusable UI
- large modular apps
- easy to follow data flow (one-way data flow)

THANKS!

Sarah Ransohoff
@sranso