Front End
Software Development

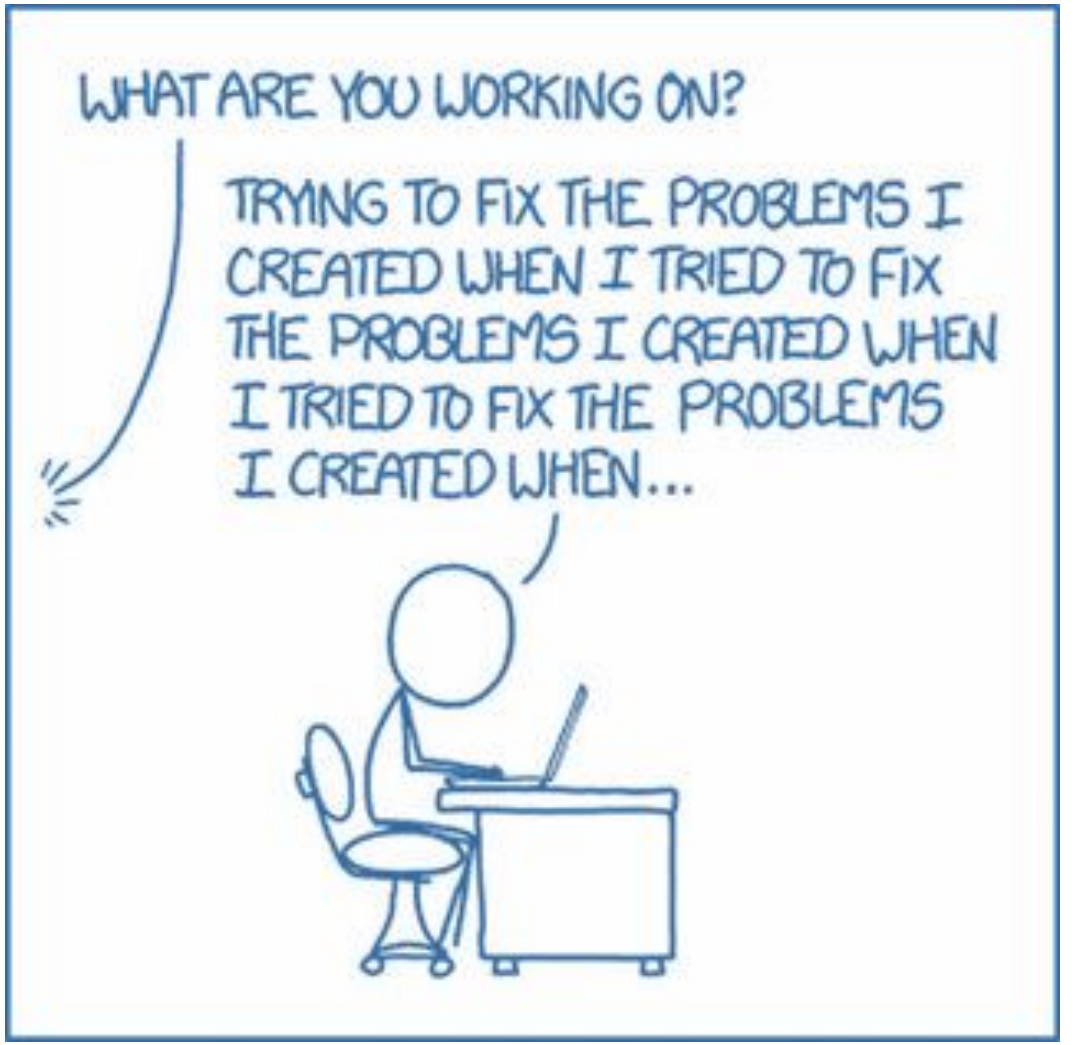
Web App Design w/React (weeks 13 - 18)

Week 02

Agenda

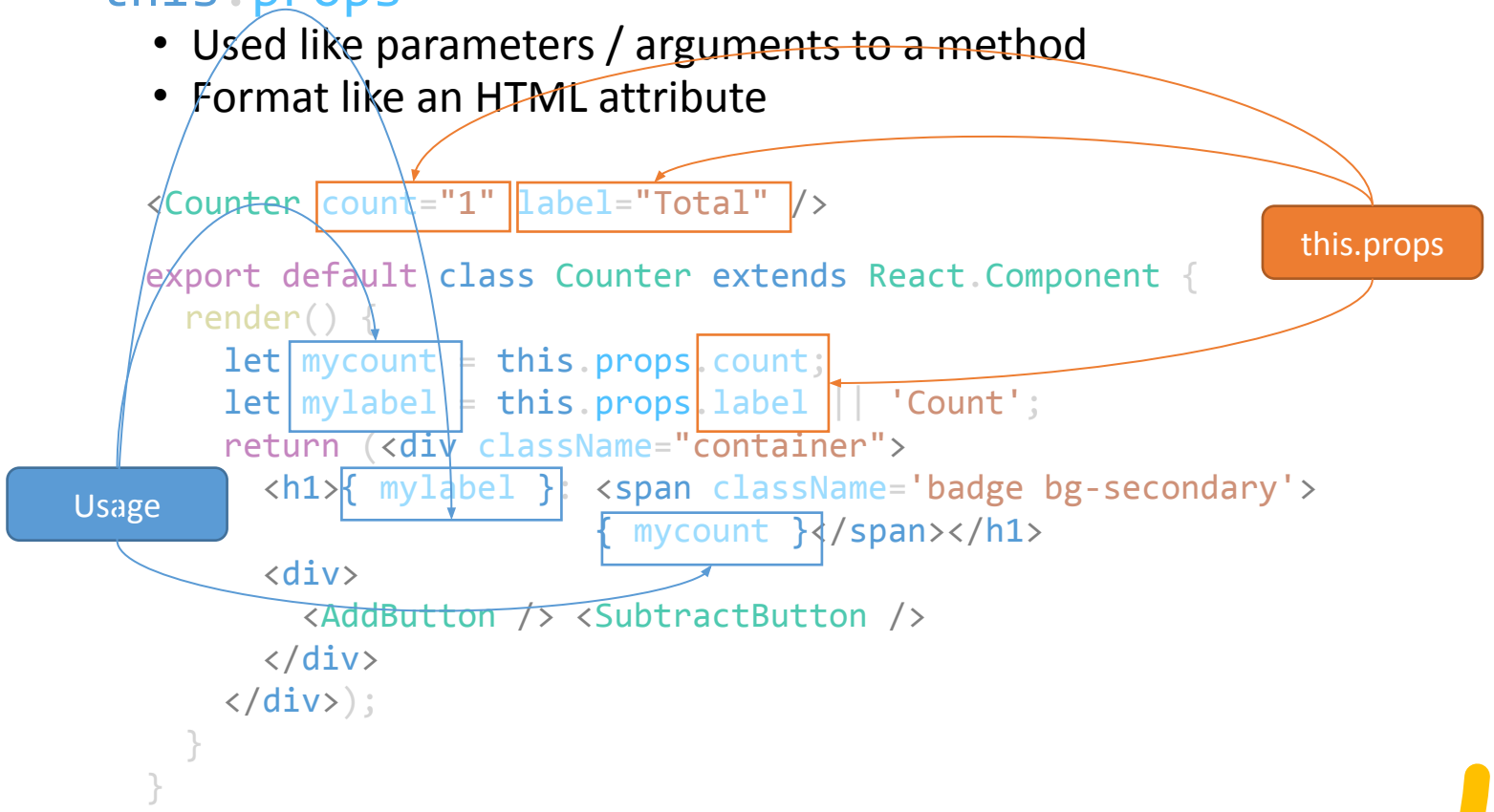
- Questions
- Props
- State
- Updating State
- Unique Key Prop
- Events
- Intro to Lifecycle Methods

Questions

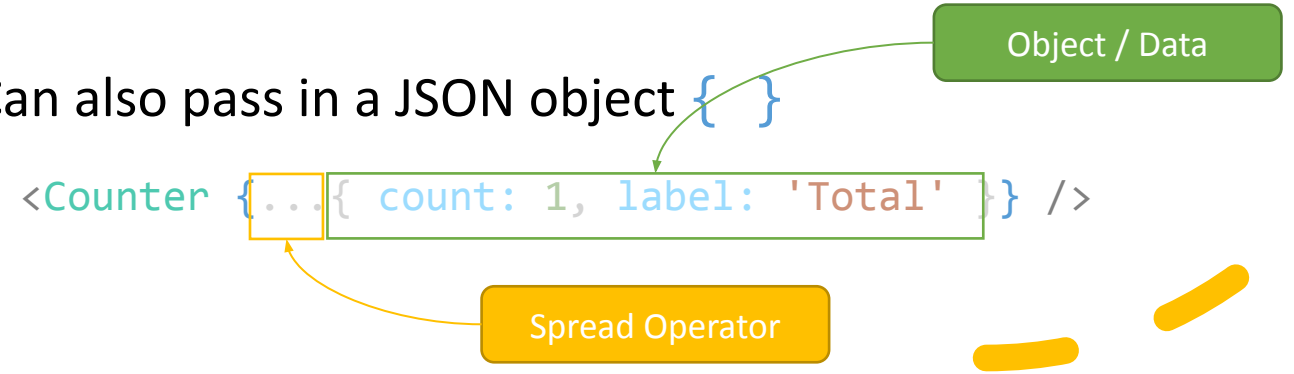

WHAT ARE YOU WORKING ON?
TRYING TO FIX THE PROBLEMS I CREATED WHEN I TRIED TO FIX THE PROBLEMS I CREATED WHEN I TRIED TO FIX THE PROBLEMS I CREATED WHEN...

Props

- this.props
  - Used like parameters / arguments to a method
  - Format like an HTML attribute



- Can also pass in a JSON object { }



State

- `this.state = { };`
  - Controls "current" values
  - Initialize within class `constructor(props)`

```
export default class Counter extends React.Component {
  constructor(props) {
    super(props);
    this.state = {
      count: props.count || 0,
      label: props.label || 'Count'
    };
  }

  render() {
    let mycount = this.state.count;
    let mylabel = this.state.label || 'Count';

    return (<div className="container">
      <h1>{ mylabel }: <span className='badge bg-secondary'>
                       { mycount }</span></h1>
      <div>
        <AddButton /> <SubtractButton />
      </div>
    </div>);
  }
}
```

Initalize "state"

Use "state"

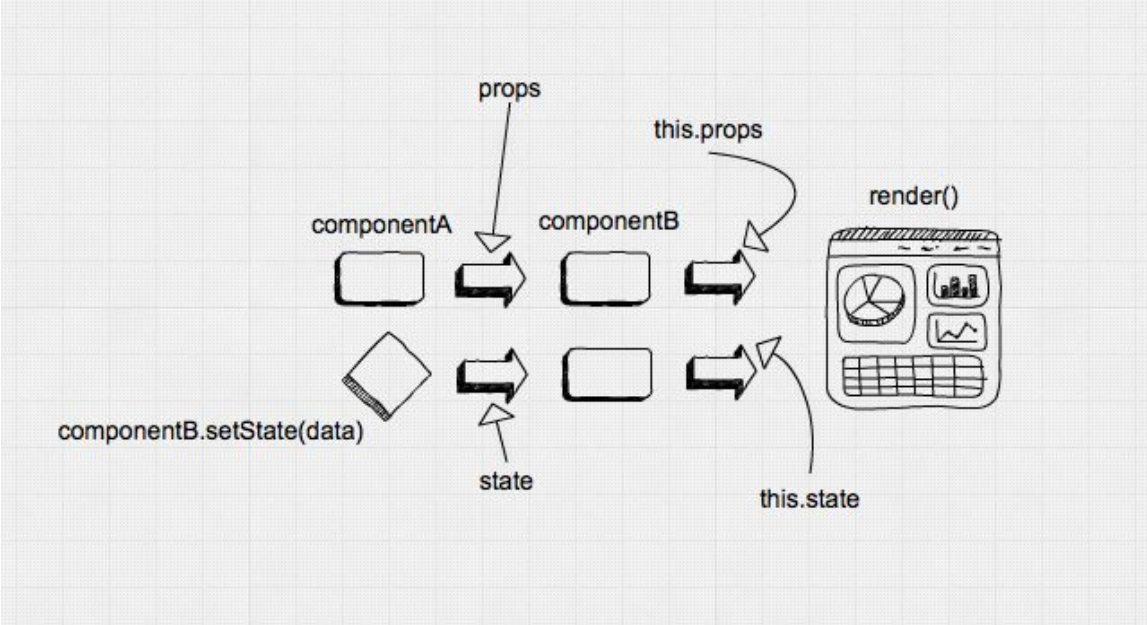
Called when object created

Updating **State**

- Changing state triggers, a component to render itself.
- **DO NOT** set state directly.

```
this.state.count =
this.state.count++;
```

  - Doing so prevents the render lifecycle from triggering.
- Use **`this.setState();`**



Upating State

(Continued)

- `this.setState();`
  - Pass in object containing updated or changed values.
  - Can pass callback to immediately access state.

```
render() {
  setTimeout(() => {
    let currentCount = this.state.count;
    this.setState((s,p) => ({ count: currentCount + 1 }));
  }, 1000);
  let mycount = this.state.count;
  let mylabel = this.state.label || 'Count';

  return (<div className="container">
    <h1>{ mylabel }: <span className='badge bg-secondary'>
                     { mycount }</span></h1>
    <div>
      <AddButton /> <SubtractButton />
    </div>
  </div>);
}
```

Use setState()

Only pass in the properties / values that have changed.

Wrapper method to avoid infinite loop error. State should **NOT** be modified in the render method.

Unique Key Prop

- Repeated items *(i.e. list, row)*
  - Requires unique id per element (i.e. like primary key in a database)
  - `key={ }` property
- **Why**?
  - Allows React to keep track of elements to determine changes.

Events

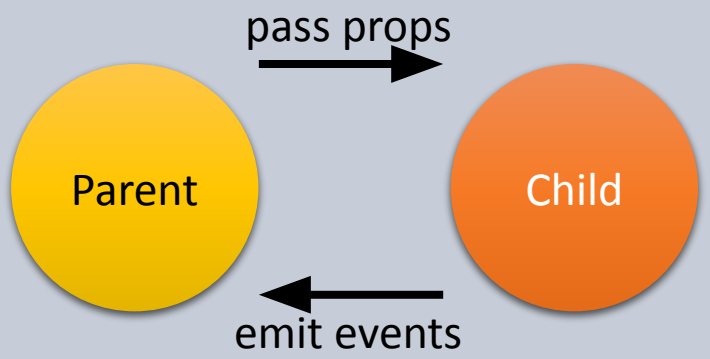
```
export default class Counter extends React.Component {
  constructor(props) {
    super(props);
    this.state = { count: props.count || 0 };
    this.onIncrement = this.onIncrement.bind(this);
    this.onDecrement = this.onDecrement.bind(this);
  }

  onIncrement(event) {
    let count = this.state.count;
    this.setState({ count: count + 1 });
  }
  onDecrement(event) {
    let count = this.state.count;
    this.setState({ count: count - 1 });
  }

  render() {
    let mycount = this.state.count;

    return (<div className="container">
      <h1>Count: <span className='badge bg-secondary'>
                 { mycount }</span></h1>

      <div>
        <button onClick={ this.onIncrement }> +1 </button>
        <button onClick={ this.onDecrement }> -1 </button>
      </div>
    </div>);
  }
}
```

HTML like events

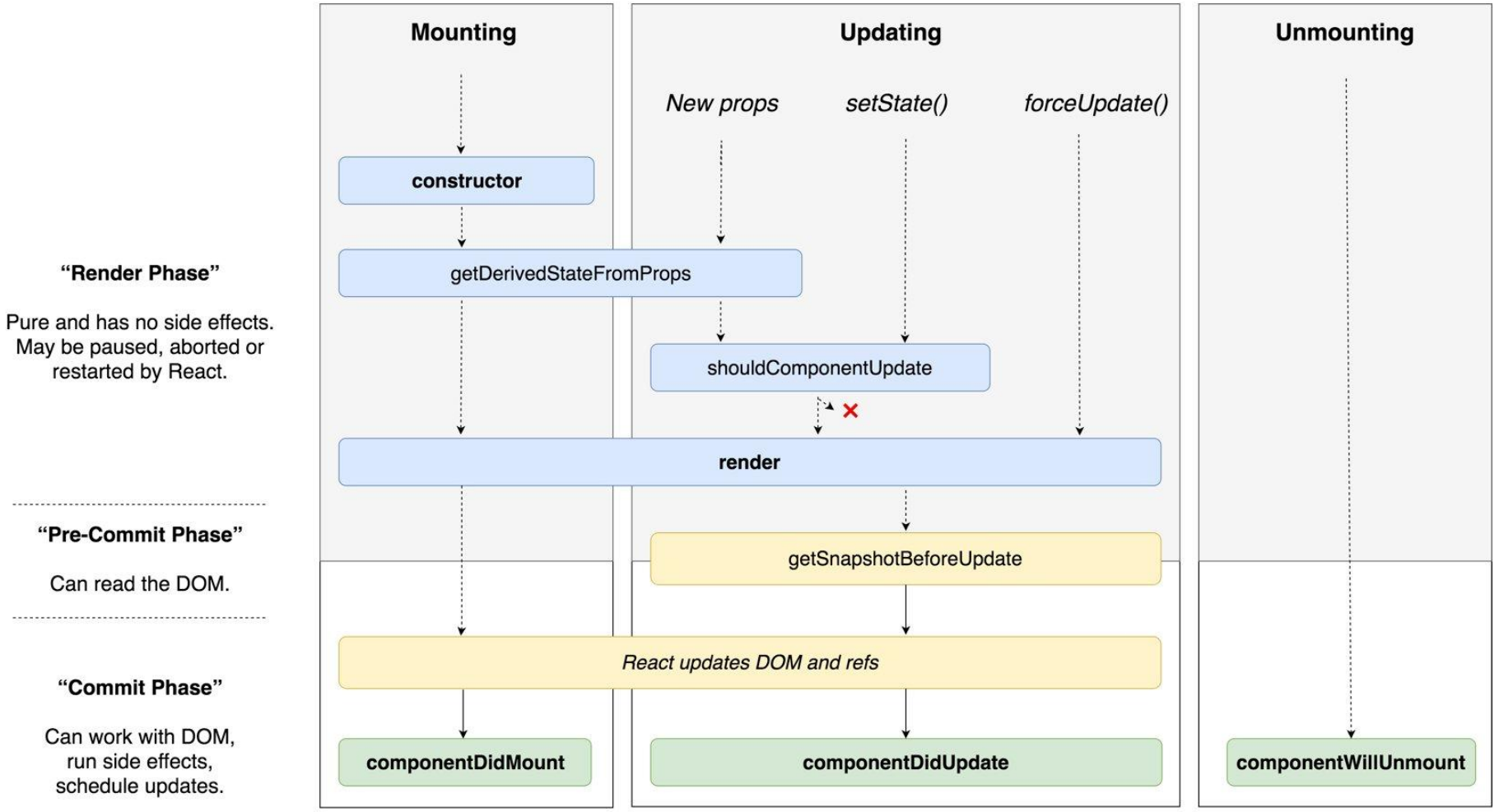
Events

(Continued)

- State flows down (one-way)
- "Raising the state"
    - Container class



Note: If you're getting double events, check to see if you have `<React.StrictMode>` set in your App.js. If so, remove them!

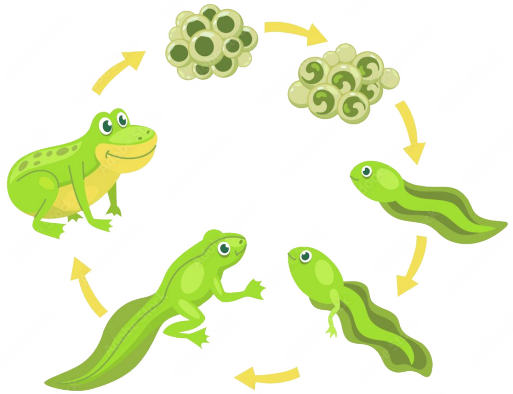


Lifecycle Methods

(Introduction)

Lifecycle Methods

(Summary)



- `componentDidMount()`
  - Called only after mounting (inserted into the tree).
  - Used for:
    - DOM node initialization
    - Load data from remote endpoint, make network request, etc.

  Note: You may call `setState()` immediately in `componentDidMount()`.

- `componentDidUpdate()`
  - Called after updating occurs.
  - Not called for the initial render.
  - Used for:
    - Manipulate DOM elements
    - Network requests (if needed)

- `componentWillUnmount()`
  - Called after unmounting and destruction.
  - Used for:
    - Cancel network requests
    - Invalidate timers
    - Unsubscribe from events

DEMO

All The Components,
Let's Talk & Play Nice!