

ReactJS.

The Component model

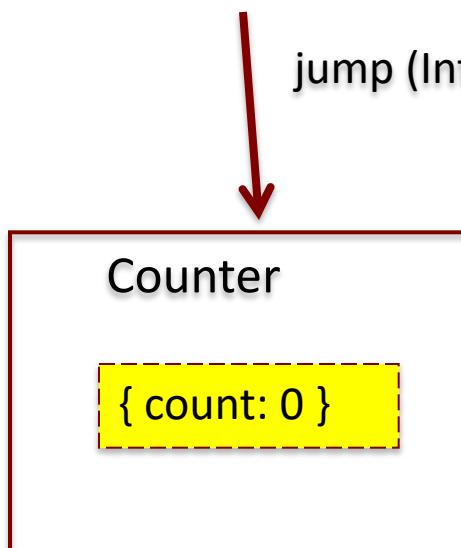
Topics

- **Component State.**
 - Basis for dynamic, interactive UI.
- **The Virtual DOM.**
- **Data Flow patterns.**
- **Lifecycle methods.**

Component DATA

- **Two sources of data for a component:**
 1. **Props - Passed in to a component; Immutable; an object (this.props).**
 2. **State - Managed internally by the component; Mutable; an object (this.state)**
 - ***** The basis for dynamic and interactive Uis *****
- **Props-related features:**
 - **Default values.**
 - **Type-checking.**
- **State-related features:**
 - **Initialization.**
 - **Mutation - the setState() method.**
 - **Performs a merge operation, not an overwrite.**
 - ***** Automatically causes component to re-render. *****

Component Data- Example

- **The Counter component.**
 - **Ref.** samples/06_state.js
 - **Coding features:**
 1. **Custom function/method, e.g.** incrementCount().
 2. **Static class property, e.g.** defaultProps.
 3. **Class instance property, e,g.** state.
- 
- A red arrow points from the word 'jump (Integer)' to a box labeled 'Counter'. Inside the 'Counter' box is a yellow box containing the code '{ count: 0 }'.

React's event system.

- **Cross-browser support.**
- **Event handlers receive SyntheticEvent – a cross-browser wrapper for the browser's native event.**
- **React event naming convention slightly different from native:**

React	Native
onClick	onclick
onChange	onchange
onSubmit	onsubmit

- See <https://reactjs.org/docs/events.html> for full details,

Automatic Re-rendering

- **EX.: The Counter component.**

User clicks ‘increment’ button

→ onClick event handler (incrementCounter) executed
→ state is changed (setState())
→ render() method executed

Modifying the DOM

- DOM – an internal data structure; mirrors the state of the UI; always in sync.
- Traditional performance best practice:
 1. Minimize access to the DOM.
 2. Avoid expensive DOM operations.
 3. Update elements offline, then reinsert into the DOM.
 4. Avoid changing layouts in Javascript.
 5. Etc.
- Should the developer be responsible for low-level DOM optimization? Probably not.
 - React provides a Virtual DOM to shield developer from these concerns.

The Virtual DOM

- **Consequence:** Re-render everything on every update.
 - Sounds expensive!
- **How?**
 1. Create a lightweight, efficient form of the DOM – the Virtual DOM.
 2. React app changes V DOM
 3. React engine:
 1. Perform *diff* operation between current and previous V DOM state.
 2. Compute the minimal set of changes to apply to (real) DOM.
 3. Batch execute all updates to real DOM.
- **Benefits:**
 - a) Clean – Clean, descriptive programming model.
 - b) Fast - Optimized DOM updates and reflows.

Automatic Re-rendering (detail)

- **EX.: The Counter component.**

User clicks ‘increment’ button

→ *onClick event handler (incrementCounter) executed*

→ *state is changed (setState())*

→ *render() method executed*

→ *The Virtual DOM has changed*

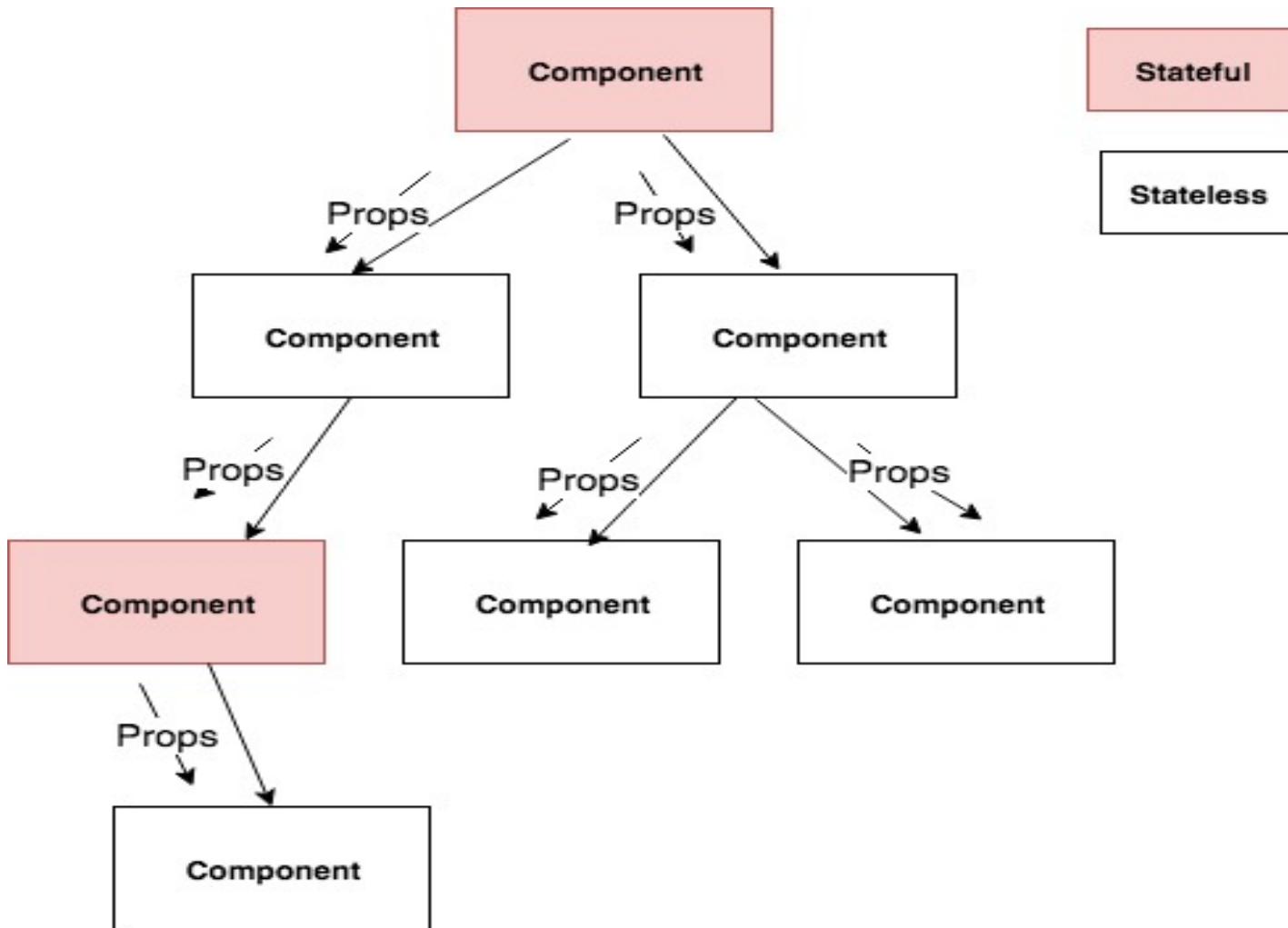
→ *React diffs the changes (between the current and previous Virtual DOM)*

→ *React batch updates the Real DOM*

Topics

- Component State. ✓
- The Virtual DOM. ✓
- Data Flow patterns.
- Lifecycle methods.

Unidirectional data flow



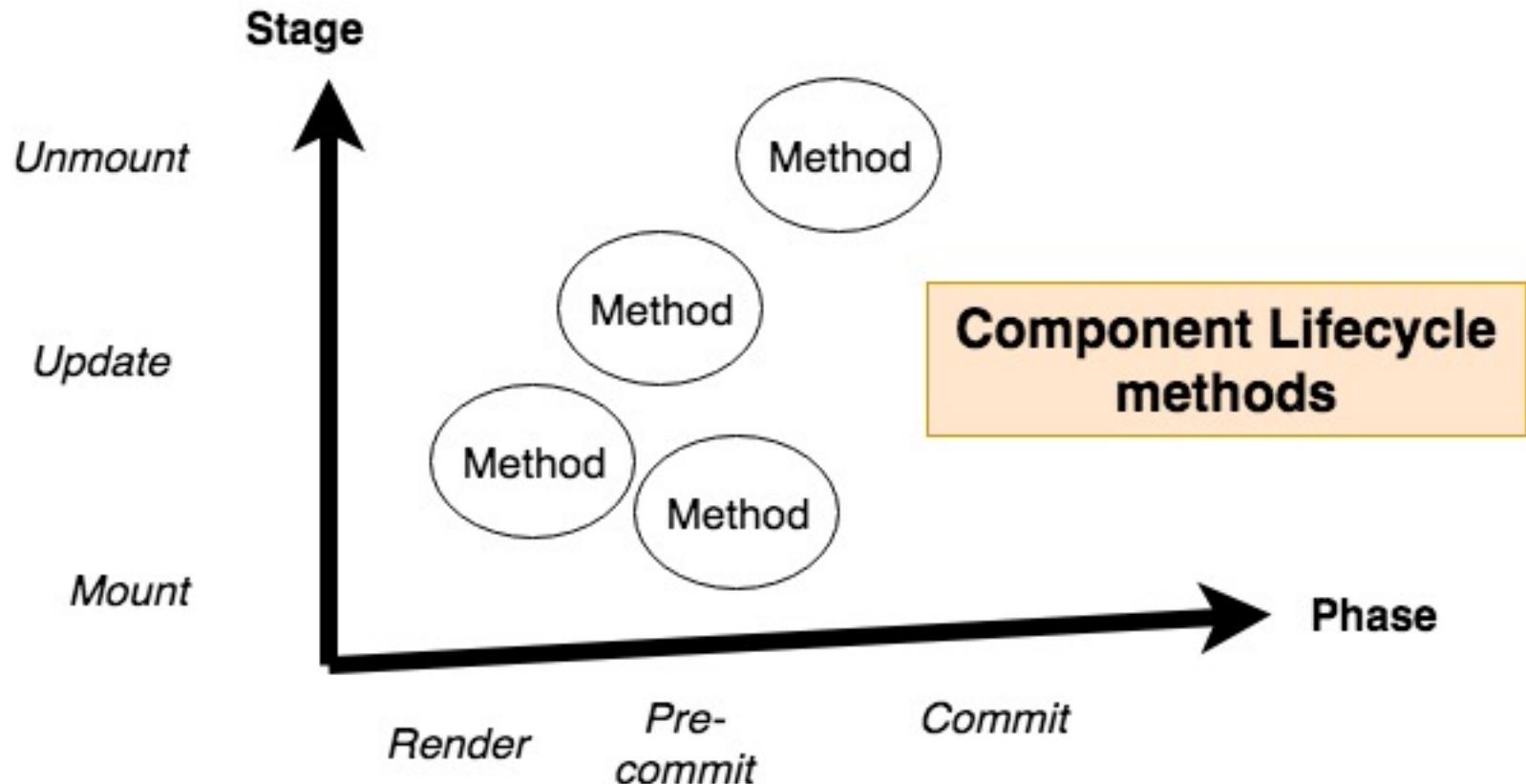
Unidirectional data flow

- In a React app, data flows uni-directionally **ONLY**.
 - Most other SPA frameworks use two-way data binding.
- In a multi-component app, a common pattern is:
 - A small subset (maybe only 1) of components will be statefull – the rest are stateless.
- Statefull components:
 - Call `setState()` to update its state.
 - Re-renders itself.
 - Pass updated props to subordinate components.
 - *React guarantees subordinate components are re-rendered with updated prop values.*

Topics

- Component State. ✓
- The Virtual DOM. ✓
- Data Flow patterns. ✓
- Lifecycle methods.

Component *Lifecycle methods*



Component *Lifecycle methods*

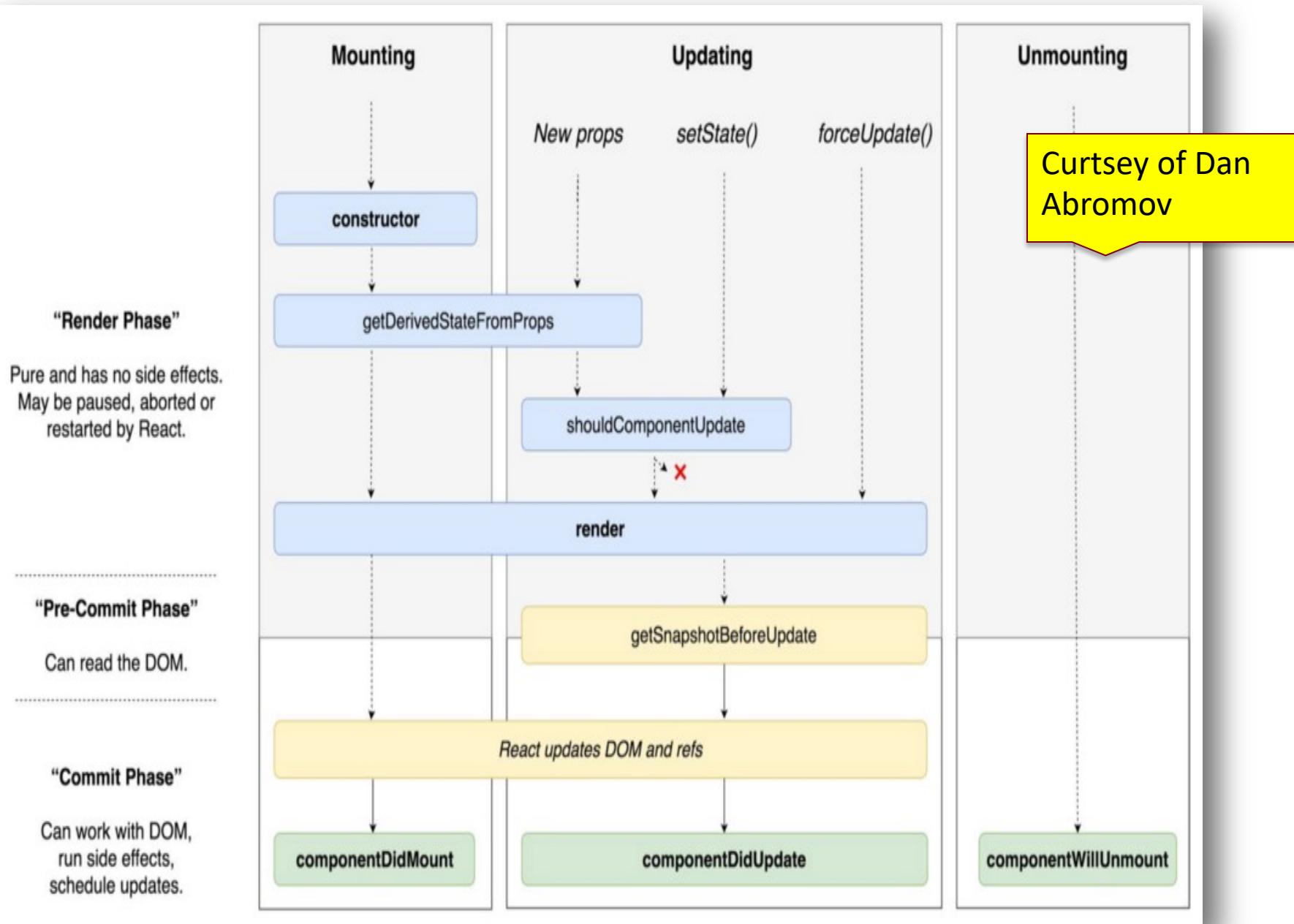
- Methods invoked by React at specific times in a component's lifecycle (Most are optional).
- Lifecycle stages:
 - 1 Mounting (Initialization).
 - 2 Update.
 - a) New props (even when value is unchanged!!).
 - b) `setState()`.
 - c) `forceUpdate`.
 - 3 Un-mounting.
- Phases:
 - Render phase.
 - Pre-commit phase (Pre DOM update).
 - Commit phase (Post DOM update).

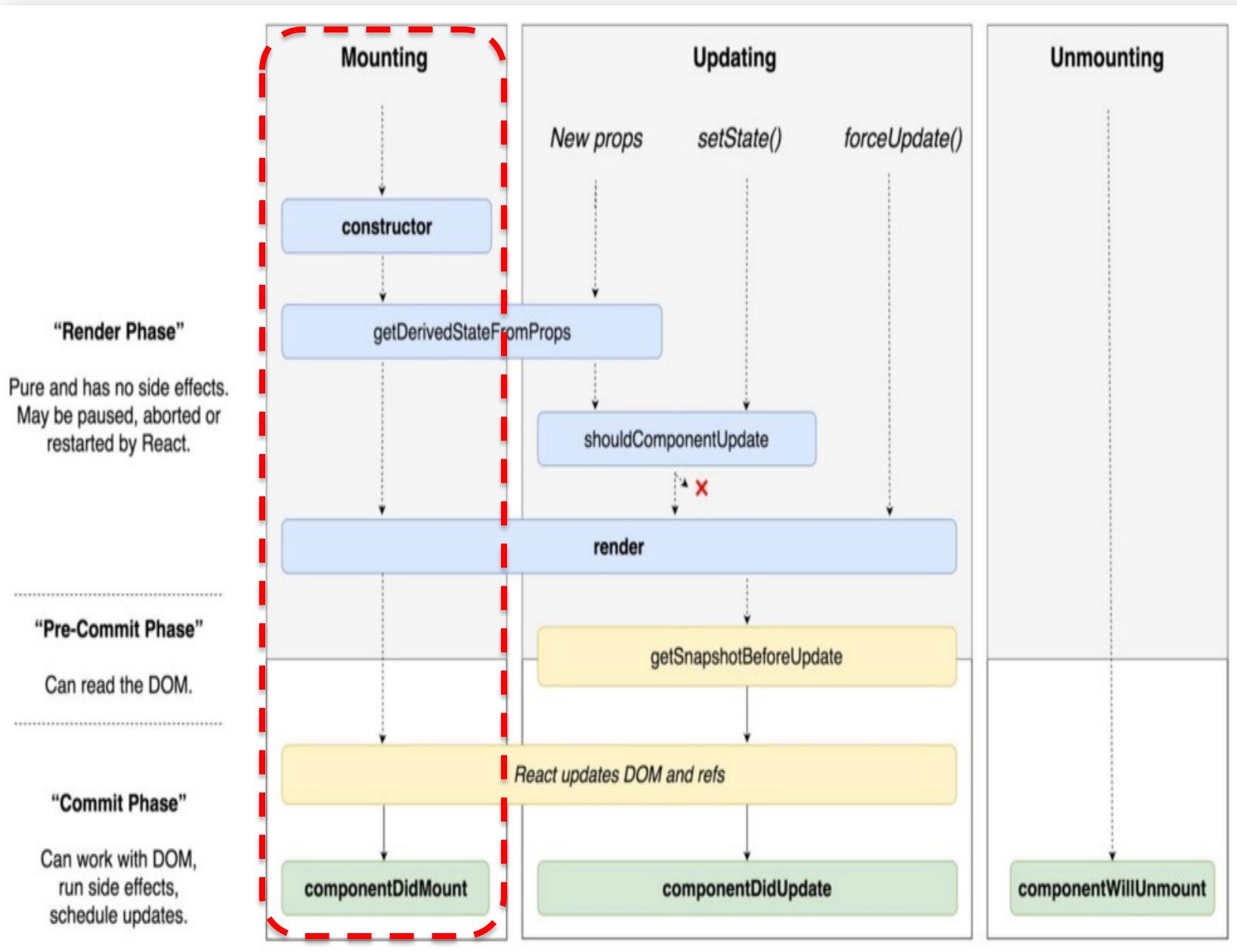
The Lifecycle methods

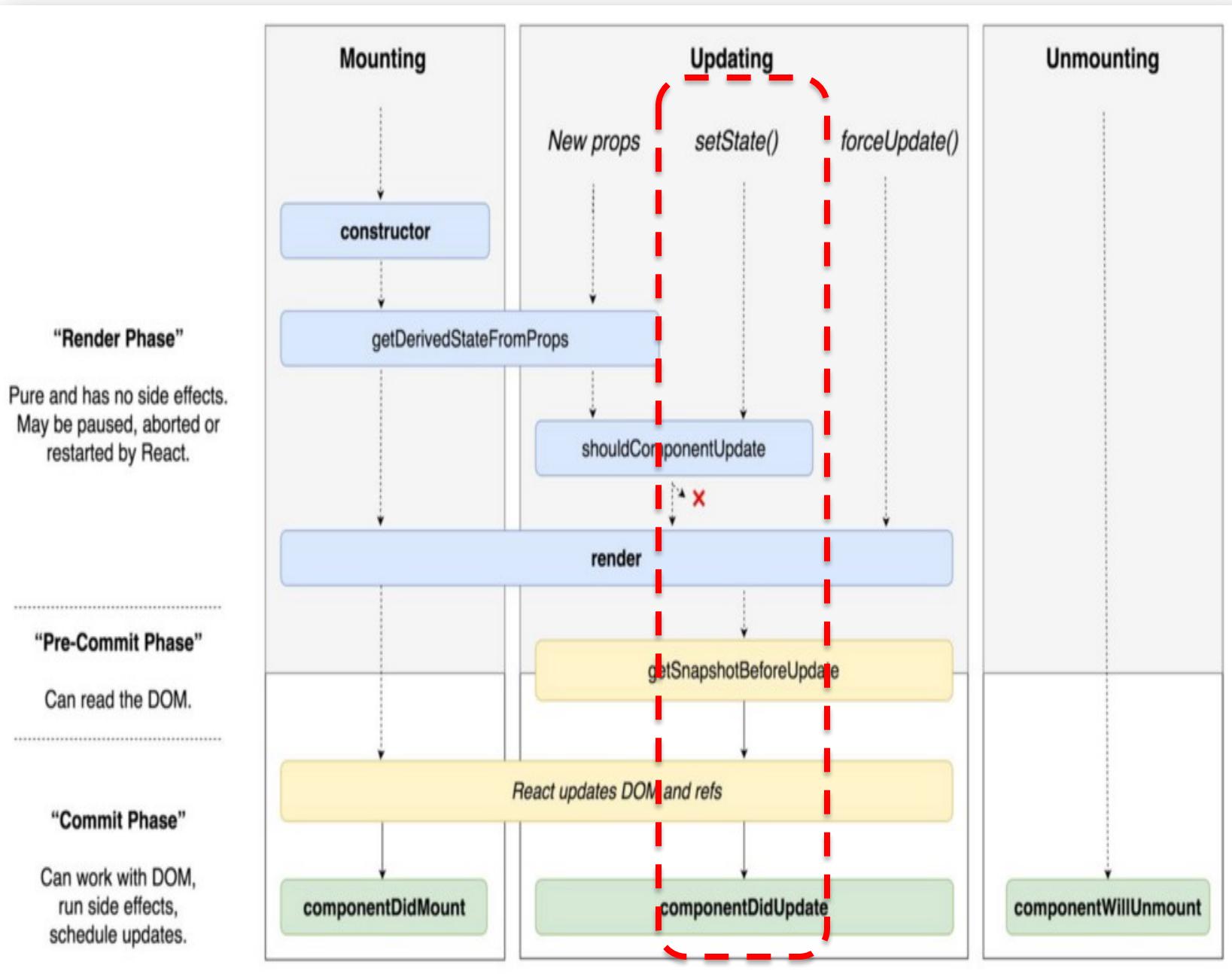
1. `shouldComponentUpdate()` – returns boolean – can cause a component to skip re-rendering.
2. `getDerivedStateFromProps()` - when a component state object is computed from its prop values.
3. `componentDidUpdate()` – executed after a rendering has updated the real DOM, e.g. perform real DOM manipulation, set up external subscription, cause side-effect.
4. `componentDidMount()` – executed after component has mounted (see later)
5. `componentWillUnmount()`; executed before a component is about to unmount; Perform cleanup operations, e.g. remove external subscription.

The Lifecycle methods

- *shouldComponentUpdate()* $\diamond\diamond\diamond$
 - *getDerivedStateFromProps()*.
 - *render()*. $\diamond\diamond\diamond$
 - *componentDidUpdate()*
 - *componentDidMount()* $\diamond\diamond\diamond$
 - *componentWillUnmount()*
- $\diamond\diamond\diamond$ used most frequently

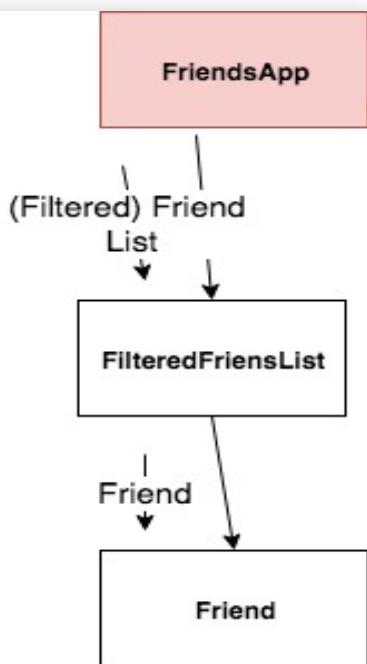






Sample App

Component hierarchy diagram



Friends List

- **Joe Bloggs**

jbloggs@here.con

- **Paula Smith**

psmith@here.con

- **Catherine Dwyer**

cdwyer@here.con

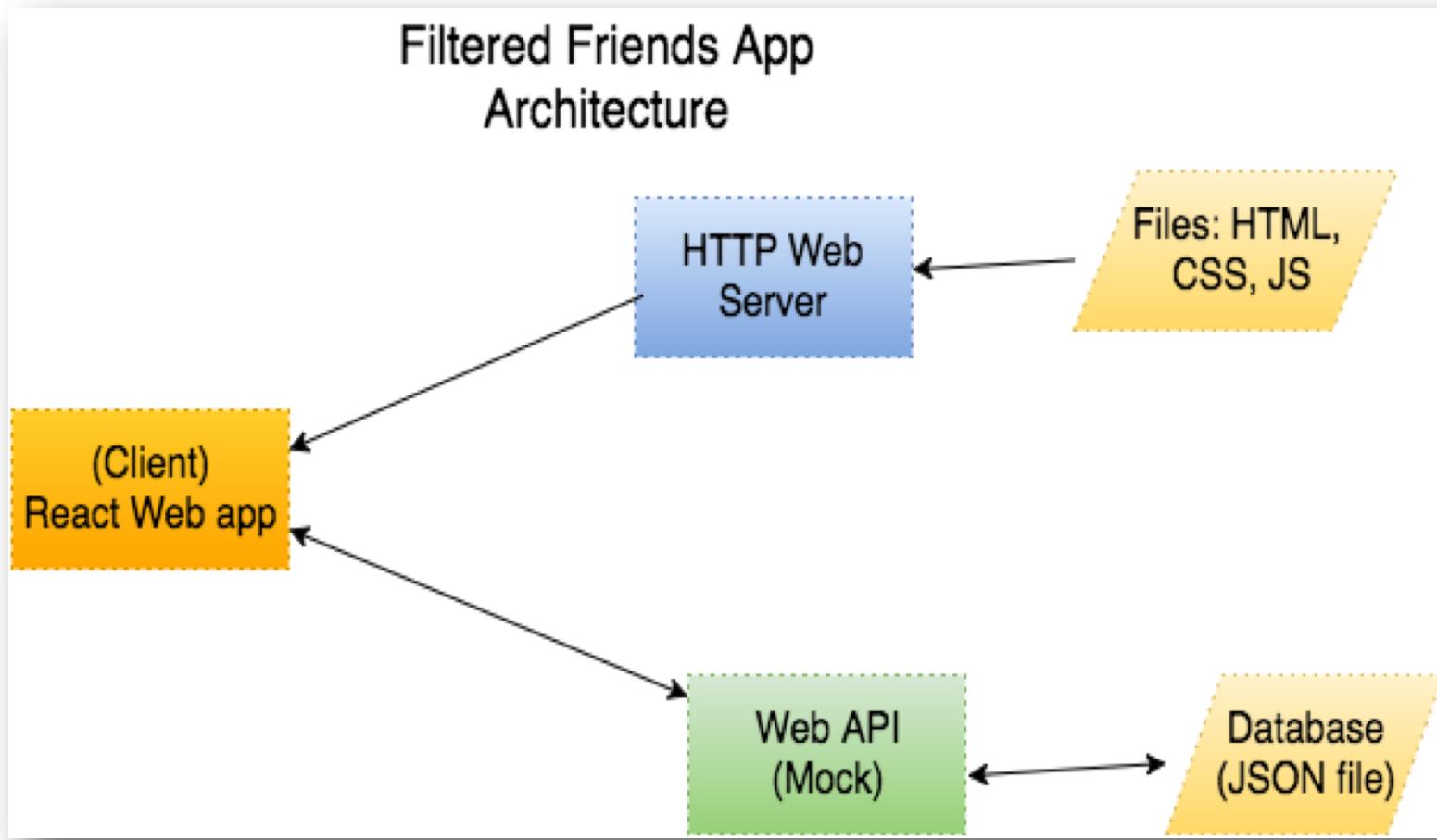
- **Paul Briggs**

pbriggs@here.con

FriendsApp component:

1. Manages app's state (i.e. text box content).
2. Computes matching friends list.
3. Controls list re-rendering.

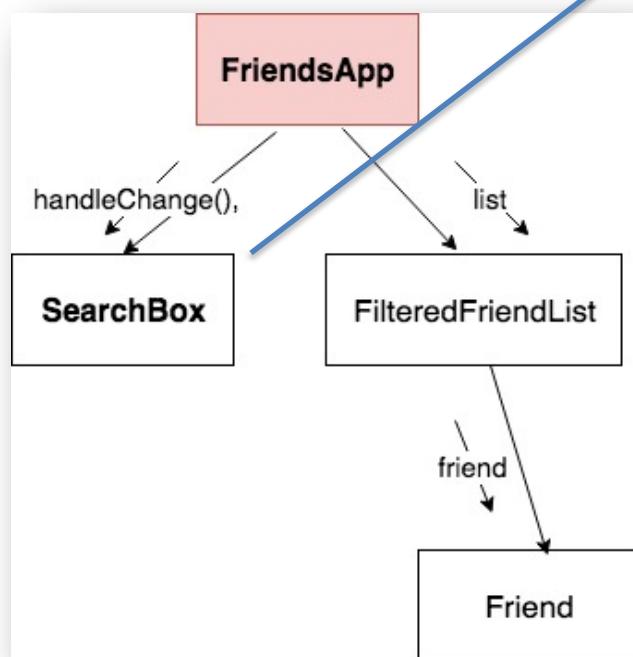
Sample App – Architecture..



DEMO

Inverse data flow

- What if a component's state is effected by an event in a subordinate component?
- Solution: The inverse data flow pattern.



..... back to Lifecycle methods

Sample App – Execution trail (Mounting & setState)

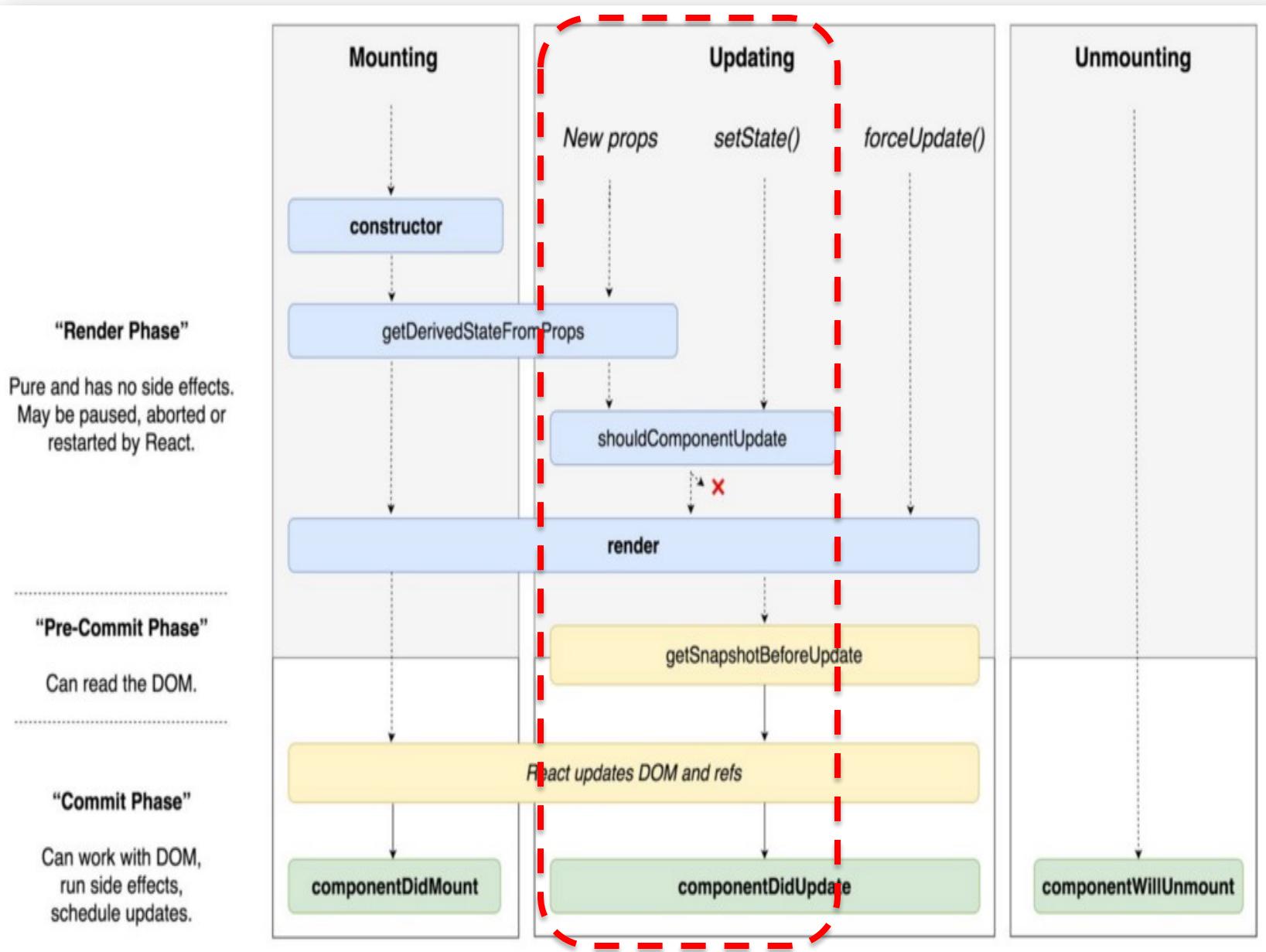
The screenshot shows a web browser window with the URL `localhost:3000`. The main content is a "Friends List" application. On the left, there is a search bar labeled "Search". Below it, a list of friends is displayed with their names and email addresses:

- Joe Bloggs
jbloggs@here.con
- Paula Smith
psmith@here.con
- Catherine Dwyer
cdwyer@here.con
- Paul Briggs
pbriggs@here.con

On the right, the browser's developer tools are open, specifically the "Console" tab. The console output shows the execution trail of the application:

```
render of FriendsApp
render of FilteredFriendList
componentDidMount of FriendsApp
render of FriendsApp
render of FilteredFriendList
render of Friend (Joe Bloggs)
render of Friend (Paula Smith)
render of Friend (Catherine Dwyer)
render of Friend (Paul Briggs)
```

A red oval highlights the entry "componentDidMount of FriendsApp".



Sample App – Execution trail (Update on new props & setState)..

The screenshot shows a browser window with a "Friends List" application on the left and a developer tools console on the right.

Friends List Application:

- A text input field containing "paula" with a red arrow pointing to it.
- A list item: ". • Paula Smith"
- A link: "psmith@here.com"

Console Output:

```
Console was cleared
< undefined
  render of FriendsApp
  render of FilteredFriendList
  render of Friend (Paula Smith)
  render of Friend (Paul Briggs)
  render of FriendsApp
  render of FilteredFriendList
  render of Friend (Paula Smith)
  render of Friend (Paul Briggs)
  render of FriendsApp
  render of FilteredFriendList
  render of Friend (Paula Smith)
  render of Friend (Paul Briggs)
  render of FriendsApp
  render of FilteredFriendList
  render of Friend (Paula Smith)
  render of Friend (Paul Briggs)
  render of FriendsApp
  render of FilteredFriendList
  render of Friend (Paula Smith)
  >
```

Note: The text box event handler calls `setState()` on FriendsApp

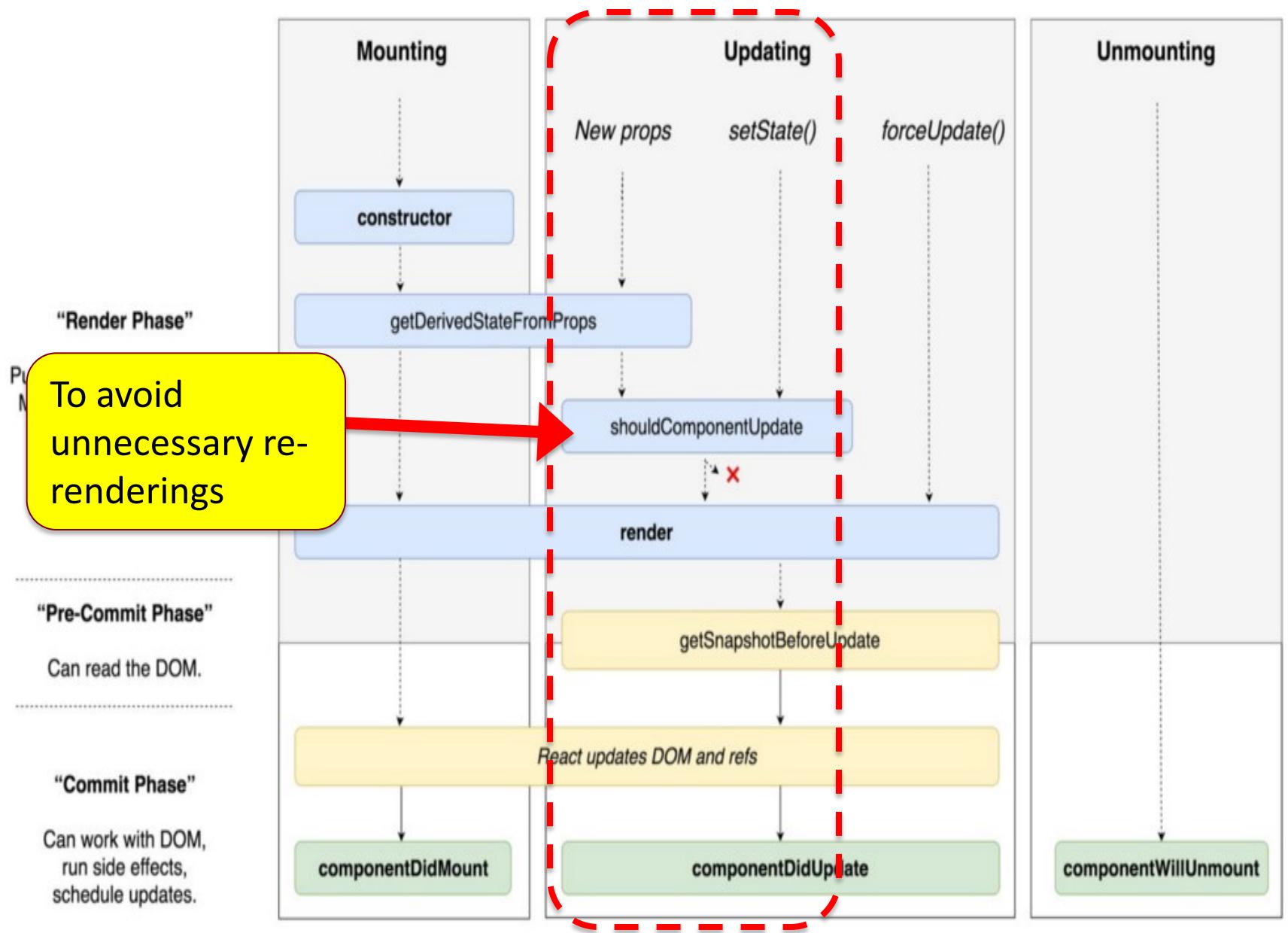
Unidirectional data flow & Re-rendering

- What happens when user types in text box?

User types a character in text box

→ *onChange event handler executes*

- *Handler calls setState() (FriendsApp component)*
 - *React calls FriendsApp render() method*
 - *React calls render() method of children (FilteredFriendList) with new prop values*
 - *React calls render() method of children of FilteredFriendList.*
 - *(Pre-commit phase) React re-computes the new Virtual DOM*
 - *React diffs the new and previous Virtual DOMs*
 - *(Commit phase) React batch updates the Real DOM*
 - *Browser repaints screen*



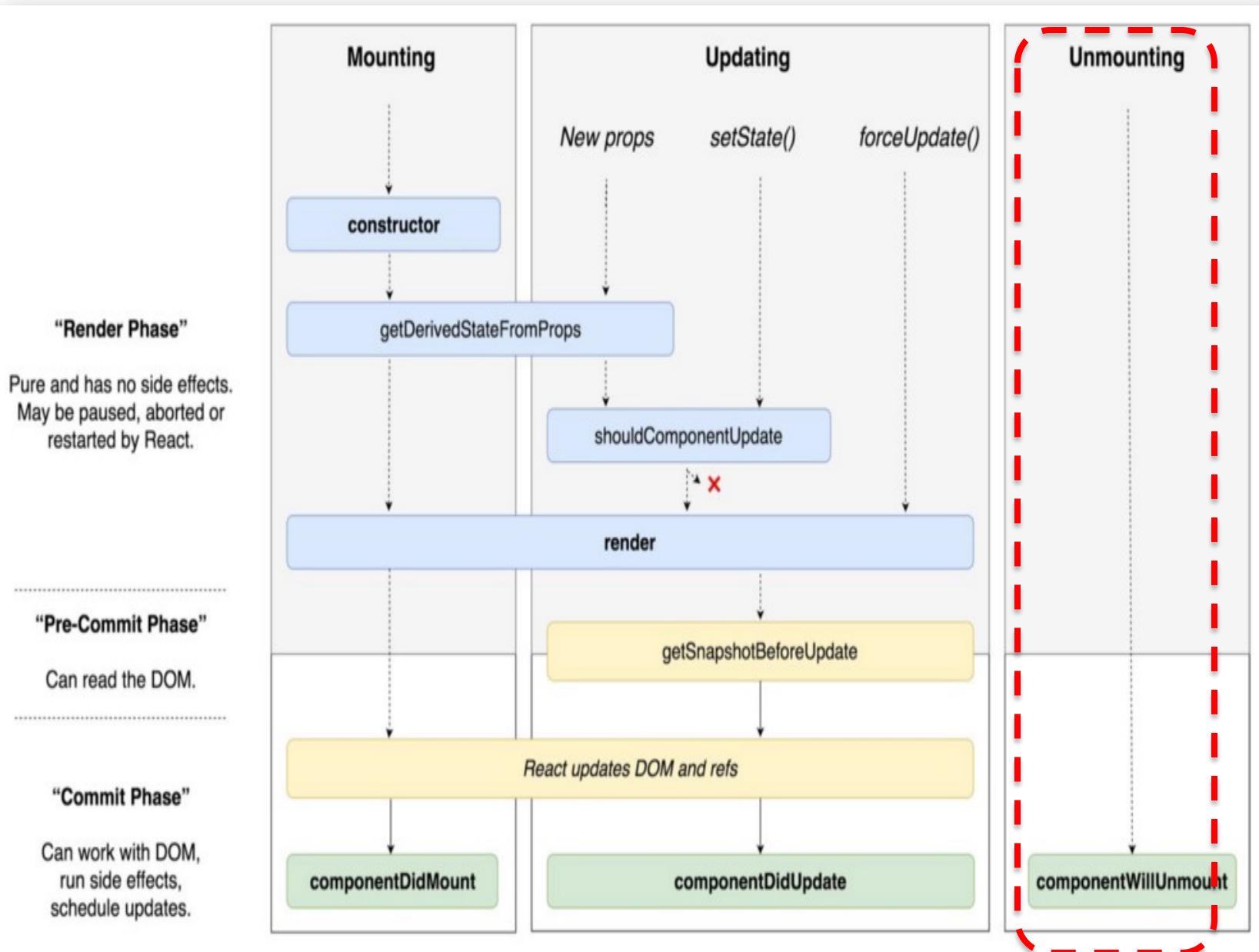
Sample App – Execution trail (Update on new props & setState)..

FilteredFriendsList should NOT re-render if the length of the array of matching friends (its prop) has not changed.

Friends List

- Paula Smith
 - psmith@here.con

```
Console was cleared
< undefined
p render of FriendsApp
    shouldComponentUpdate of FilteredFriendList
        render of FilteredFriendList
            render of Friend (Paula Smith)
            render of Friend (Paul Briggs)
a render of FriendsApp
    shouldComponentUpdate of FilteredFriendList
        render of FilteredFriendList
u render of FriendsApp
    shouldComponentUpdate of FilteredFriendList
        render of FilteredFriendList
l render of FriendsApp
    shouldComponentUpdate of FilteredFriendList
        render of FilteredFriendList
a render of FriendsApp
    shouldComponentUpdate of FilteredFriendList
        render of FilteredFriendList
            render of Friend (Paula Smith)
>
```



Sample App

– Execution trail (Un-mounting)...

The screenshot shows a developer tool's execution trail for a React application. The trail consists of a vertical list of log entries, each representing a component's lifecycle event. A red oval highlights the initial rendering phase, labeled 'App start-up'. A second red oval highlights the phase where a friend named 'Paul' is selected, labeled 'Typed 'p''. A third red oval highlights the phase where a friend named 'Joe' is deleted, labeled 'Typed ''. The log entries are:

- render of FriendsApp
- render of FilteredFriendList
- componentDidMount of FriendsApp
- render of FriendsApp
- shouldComponentUpdate of FilteredFriendList
- render of FilteredFriendList
- render of Friend (Joe Bloggs)
- render of Friend (Paula Smith)
- render of Friend (Catherine Dwyer)
- render of Friend (Paul Briggs)
- render of FriendsApp
- shouldComponentUpdate of FilteredFriendList
- render of FilteredFriendList
- shouldComponentUpdate of Friend (Paula Smith)
- shouldComponentUpdate of Friend (Paul Briggs)
- componentWillUnmount of Friend (Joe Bloggs)
- componentWillUnmount of Friend (Catherine Dwyer)
- render of FriendsApp
- shouldComponentUpdate of FilteredFriendList
- render of FilteredFriendList
- render of Friend (Joe Bloggs)
- shouldComponentUpdate of Friend (Paula Smith)
- render of Friend (Catherine Dwyer)
- shouldComponentUpdate of Friend (Paul Briggs)

Summary

- For interactive apps we record the user's input/interaction in component(s) state object.
 - The interaction may cause UI changes – dynamic app.
- React achieves DOM update performance improvements by managing an intermediate data structure, the Virtual DOM.
- Data only flows downward through the component hierarchy – this aids debugging.
- A component's life-span includes stages, from mounting to un-mounting, and phases, including render, pre-commit and post-commit.
 - We can hook logic in to the life span at prescribed times using lifecycle methods.

