React Lifecycle

Basic Component Definition ...

- **`getDefaultProps()`** not typically used
  - return object describing initial props for component
  - access with **`this.props`**
  - only needed for props that might not be passed in by parent components
  - called once regardless of # of instances created
  - in components implemented with an **ES6 class**,
    set **`defaultProps`** property on class
    instead of implementing this method
- **`getInitialState()`** not typically used
  - return object describing initial state for component
  - access with **`this.state`**
  - only needed in components that maintain their own state
  - in components implemented with an **ES6 class**,
    set **`this.state`** in constructor
    instead of implementing this method

... Basic Component Definition

- **`render()`** must have!
  - returns component markup, typically specified with JSX
    - alternative is to specify by calling JavaScript functions
  - return **`false`** or **`null`** to render nothing
  - should be a pure function
    - return same thing for same values of **`this.state`** and **`this.props`**
    - do not modify DOM or cause other side effects
  - cannot modify **`this.state`** or **`this.props`** here

Component Life Cycle

- Three main parts
- **Mount**
  - component initialized and inserted into DOM
  - includes initial render
- **Update**
  - component re-rendered to virtual DOM and actual DOM updated if needed
  - triggered by state or prop changes
- **Unmount**
  - component removed from DOM

Changes in React 16.3+

- New lifecycle methods
  - static `getDerivedStateFromProps(nextProps, prevState)`
    - "invoked after a component is instantiated as well as when it receives new props"
    - "should return an object to update state, or null to indicate that the new props do not require any state updates"
    - "if a parent component causes your component to re-render, this method will be called even if props have not changed"
  - `getSnapshotBeforeUpdate(prevProps, prevState)`
    - "invoked right before the most recently rendered output is committed to e.g. the DOM"
    - "enables your component to capture current values (e.g. scroll position) before they are potential changed"
    - "any value returned by this lifecycle will be passed as a parameter to componentDidUpdate()" in the newly supported 3rd parameter
- Deprecated lifecycle methods
  - `componentWillMount` - use constructor instead
  - `componentWillReceiveProps` - use new `getDerivedStateFromProps` method instead
  - `componentWillUpdate` - use new `getDerivedStateFromProps` method instead
  - these methods will be removed in React 17

Order of Invocation

- Mount (initial render)
  - `getDefaultProps`
  - `getInitialState`
  - `componentWillMount` deprecated!
  - **`render`**
  - **`componentDidMount`**
- Prop Change
  - `componentWillReceiveProps` deprecated!
  - **`shouldComponentUpdate`**
  - `componentWillUpdate` deprecated!
  - **`render`**
  - **`componentDidUpdate`**
- Unmount
  - `componentWillUnmount`
- State Change
  - **`shouldComponentUpdate`**
  - `componentWillUpdate` deprecated!
  - **`render`**
  - **`componentDidUpdate`**

Lifecycle Methods ...

I wish these methods did not have "component" in their name ... too verbose!

- **`componentWillMount()`** deprecated! `componentWillUpdate` is a related method
  - invoked immediately before initial render
  - can create new state and pass to `this.setState` without triggering another render
  - cannot perform DOM manipulation here

- **`componentDidMount()`** not typically used `componentDidUpdate` is a related method
  - invoked immediately after initial render
  - can perform setup such as
    loading initial data from an Ajax service
    or subscribing to a data source
    - when data is returned, pass to `this.setState`
  - can perform **DOM manipulation** on what was rendered
  - when using Flux architecture, can subscribe to store changes here

… Lifecycle Methods …

- **`componentWillReceiveProps(nextProps)`** deprecated!
  - not called before initial render, but before others
  - can update state from props here, but this is not common
  - can call `setState` here without triggering another render
  - cannot perform DOM manipulation here

- **`shouldComponentUpdate(nextProps, nextState)`**
  - not called before initial render, but before others
  - use to **optimize performance** by avoiding unnecessary virtual DOM creation and diffing
  - return `true` to proceed with render; `false` otherwise
  - subsequent lifecycle methods will not be called if this returns `false`
  - more on this later

can efficiently compare old and new state and prop values if they are held in immutable objects

… Lifecycle Methods

- **`componentWillUpdate(nextProps, nextState)`** deprecated!
  - not called before initial render, but before others
  - cannot call `setState` here
  - for performing "preparation" before render

- **`componentDidUpdate(prevProps, prevState)`** not typically used
  - called after updates are flushed to DOM, but not after initial render
  - can perform **DOM manipulation** on what was rendered `componentDidMount` is a related method

… Lifecycle Methods

- **`componentWillUnmount()`** not typically used
  - called immediately before a component is removed from DOM
  - good place to perform teardown such as unsubscribing from data sources or store changes or call **`clearInterval`** if **`setInterval`** was used

shouldComponentUpdate

- Common way to optimize component performance
  by avoiding unnecessary virtual DOM building and diffing
- "React will invoke this function pretty often,
  so the implementation has to be fast"
- Hard to implement in components that use **`props.children`**
  - because that can include other components ("transclusion")
    and code will have to determine whether they should update
- Typical implementation

```
class MyComponent extends React.Component {
  ...
  shouldComponentUpdate(nextProps, nextState) {
    // Check all props and state the component uses.
    // This example assumes only one prop is used and
    // it has a primitive or immutable object value.
    return this.props.someProp !== nextProps.someProp;
  }
  ...
}
```

Modifying DOM

- Can modify DOM and CSS after React renders a component
  - for initial render, do in `componentDidMount` method
  - for other renders, do in `componentDidUpdate` method

```
// Inside some component ...

  componentDidMount() {
    // Called the first time this component is rendered.
    this.modifyDom();
  }

  componentDidUpdate() {
    // Called after each time this component is re-rendered.
    this.modifyDom();
  }

  modifyDom() {
    const node = ReactDOM.findDOMNode(this);
    // Can modify this node or any other node here,
    // but changes will be removed by next render.
  }
```