Lifecycle Methods:

Now that we've seen a bit of how state works at at a high level, let's dig into the component's lifecycle methods. Each component goes through a series of stages going from instantiation to being rendered.

So far, as we've seen, when we create a component, we invoked the constructor, and then invoked the render. That was it. There are, however, more methods that we can overload and put to use if we so please!

Phases:

Mounting

As in the name, these below methods are run once when we mount the component onto the DOM. They are run in the order they appear below:

- constructor():
  - The constructor of a react component calls super(props), where `this.props` gets set.
  - It is called before the component is mounted.
  - Local state is set with usage of this.state (**not set state**). Initial state is assigned with `this.state = { someStateVar: 0 }`
  - Binding of event handlers to the instance also happen in the constructor.
- static getDerivedStateFromProps():
  - Invoked immediately before the render method on both the first mounting and every following update.
  - Should return an object to update state or null to update nothing.
- render():
  - REQUIRED! The only component that must be there.
  - Returns:
    - React Elements: JSX Elements
    - Arrays && Fragments: Allows for multiple elements to be returned
    - Portals: Lets children be rendered to a different DOM subtree.
    - Strings and numbers: Rendered as text nodes.
    - Booleans/Null: for when your component may or may not return `return someBoolean && <ChildElement />`
- componentDidMount():

    - Invoked immediately after the component is mounted into the DOM.
    - If a setState is called in here, the component will be rendered twice, but generally quickly enough so that the user does not see the rerender.

Note: There does exist a *componentWillMount()*, however, that method has been deprecated and should be avoided!

Updating

These methods get run when the state or props change. When the component is re-rendered, the following methods are called in the following order:

- static getDerivedStateFromProps():
    - See Above
- shouldComponentUpdate(nextProps, nextState):
    - A function that returns a boolean. Defaults to True.
    - Allows ability to compare `this.props` and `this.state` with `nextProps` and `nextState` to determine if rendering is necessary.
- render()
    - See above
- getSnapshotBeforeUpdate(prevProps, prevState):
    - Allows for the component to capture the previous state before it is updated.
    - If anything is returned from this function, that is passed to `componentDidUpdate()` to the snapshot parameter
- componentDidUpdate(prevProps, prevState, snapshot)
    - Invoked immediately after DOM updating occurs (or immediately after getSnapshotBeforeUpdate, if called).
    - A good place to do quick checks on the new DOM's state against the previous state.
    - **DO NOT SET STATE UNLESS IN A CONDITIONAL**: You will trigger an infinite loop!

Note: there are two updating methods that are considered depracated: *componentWillUpdate()* and *componentWillReceiveProps()*.

Unmounting:

When a component is no longer being used, it unmounts. When that occurs, the following method is called:

- componentWillUnmount():
    - Invoked immediately before a component is removed from the DOM.
    - A good place to do any cleanup.

Error Handling:

When dealing with errors (and there are always errors) during rendering, lifecycle methods, or in any child constructors, the following methods are called:

- static getDerivedStateFromError(error)
    - Invoked after an error is thrown by a descendant component
    - Good for failing gracefully and returning an "error" component
- componentDidCatch
    - Invoked after an error has been thrown, and contains extra information about where the error came from.
    - Good for logging errors!