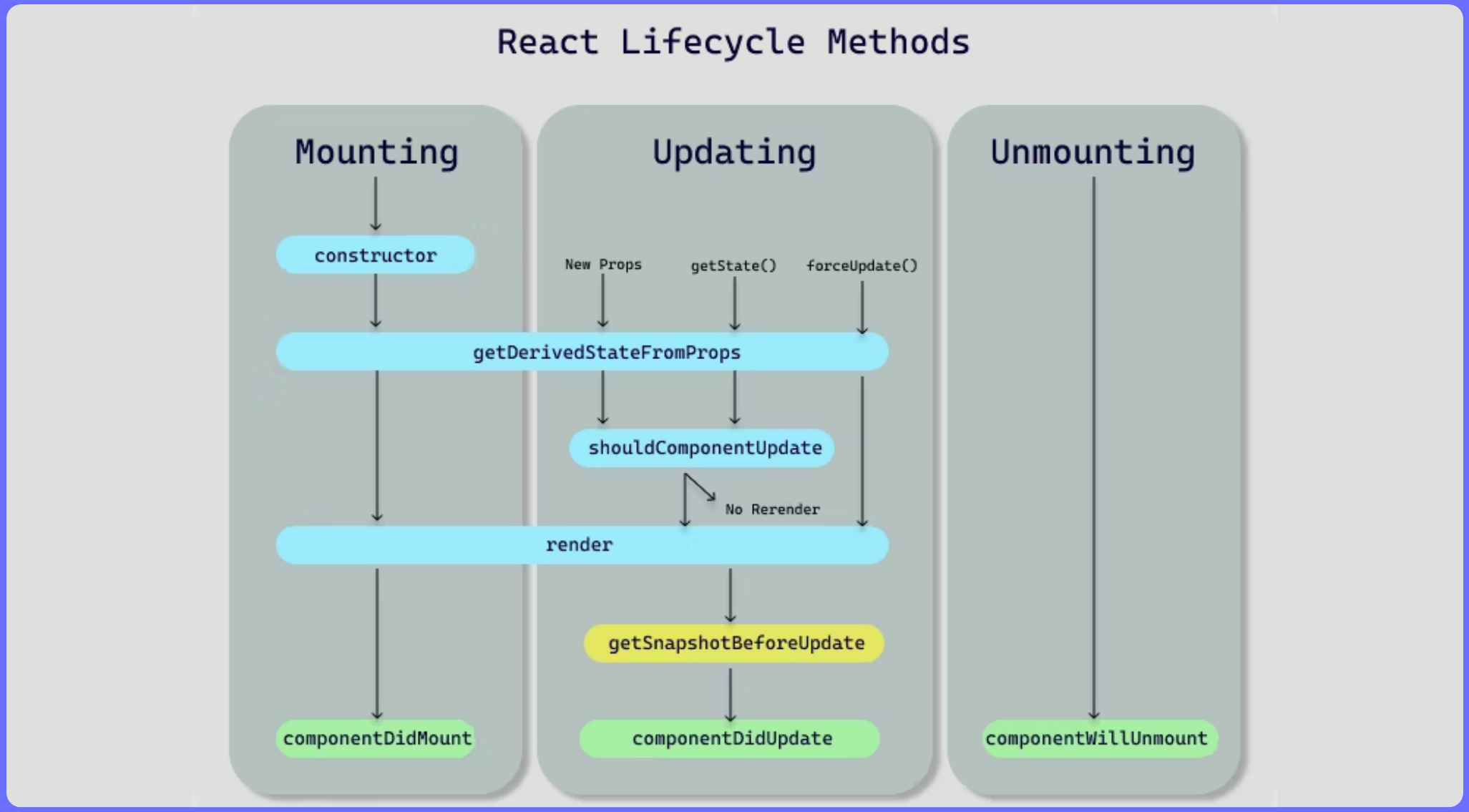
What are Component Lifecycle Methods?

Component Lifecycle Methods are a set of functions that enable developers to perform actions at specific points in a component's lifecycle.

Why Are They Important?

1. **Control and Flexibility:**
   - Component Lifecycle Methods give developers control over when and how certain tasks are executed during a component's lifetime.
   - This control allows for better management of state, data, and side effects.
2. **Optimization:**
   - Properly utilizing these methods can optimize the rendering and performance of your application.
   - For example, you can avoid unnecessary renders or handle resource cleanup when a component is unmounted.
3. **Side Effects and Data Fetching:**
   - Lifecycle methods are often used for handling side effects such as data fetching, DOM manipulation, or setting up subscriptions.
   - They ensure that these tasks are performed at the right time, preventing issues like data inconsistencies or memory leaks.
4. **Debugging and Logging:**
   - Lifecycle methods are excellent for debugging and logging purposes. They allow you to see how your component behaves at different stages.
5. **Understanding React Internals:**
   - Knowledge of component lifecycles is essential for understanding React's internal workings and making informed decisions when building applications.

Component Lifecycle, divided into three main phases: Mounting, Updating, and Unmounting.


React Lifecycle Methods
Mounting
constructor
componentDidMount
Updating
New Props
getState()
forceUpdate()
getDerivedStateFromProps
shouldComponentUpdate
No Rerender
render
getSnapshotBeforeUpdate
componentDidUpdate
Unmounting
componentWillUnmount

Mounting Phase:

- Where components are created and added to the DOM.
- the key methods associated with the Mounting phase are `constructor()`, `static getDerivedStateFromProps()`, `render()`, and `componentDidMount()`.

It involves the creation of a component and its insertion into the DOM.

Methods Associated with the Mounting Phase:

1. **constructor():**
   - `constructor()` method is called when an instance of the component is created.
   - initializing state, binding event handlers, and other setup tasks.
2. **static getDerivedStateFromProps():** `static getDerivedStateFromProps(props, state)`
   - This static method is called when a component receives new props.
   - it's used to update the state based on the new props and return an object to update the state.
   - It should return an object to update the state, or `null` to update nothing.
3. **render():**
   - The `render()` method, which returns the JSX to be rendered on the screen.
   - It should be a pure function with no side effects.
4. **componentDidMount():**
   - The `componentDidMount()` is called after the component is inserted into the DOM.
   - it's often used for tasks like data fetching, setting up subscriptions, or interacting with the DOM.
   - This is a good place to start asynchronous operations.

Execution Order of Mounting Phase Methods:

1. `constructor()`
2. `static getDerivedStateFromProps()`
3. `render()`
4. Component is inserted into the DOM.
5. `componentDidMount()`

Updating Phase:

This phase handles changes in a component's state or props, and how it responds to those changes.

Methods Associated with the Updating Phase:

1. **static getDerivedStateFromProps():** `static getDerivedStateFromProps(props, state)`
   - `static getDerivedStateFromProps()` is called whenever the component receives new props.
   - It should return an object to update the state, or `null` to update nothing.
   - its role in updating the component's state based on the new props.
2. **shouldComponentUpdate():** `shouldComponentUpdate(nextProps, nextState)`
   - the purpose of `shouldComponentUpdate()`, which determines whether a component should re-render.
   - its role in performance optimization by allowing you to prevent unnecessary renders.
   - It by default it returns true
3. **render():**
   - `render()` is called to re-render the component's user interface.
   - it should be a pure function with no side effects.
4. **getSnapshotBeforeUpdate():** `getSnapshotBeforeUpdate(prevProps, prevState)`
   - `getSnapshotBeforeUpdate()` is invoked just before the component's changes are reflected in the DOM.
   - its use for capturing information about the DOM before an update.
   - Any value returned by this lifecycle method will be passed as a parameter to `componentDidUpdate()`
5. **componentDidUpdate():** `componentDidUpdate(prevProps, prevState, snapshot)`
   - `componentDidUpdate()` is called after the component is updated in the DOM.
   - its common use cases, such as making network requests or interacting with the DOM after an update.

Execution Order of Updating Phase Methods:

1. `static getDerivedStateFromProps()`
2. `shouldComponentUpdate()`
3. `render()`
4. Component re-renders in the DOM.
5. `getSnapshotBeforeUpdate()`
6. `componentDidUpdate()`

Unmounting Phase:

This phase focuses on the cleanup and finalization of a component's behavior before it's removed from the DOM.

The purpose of `componentWillUnmount()` is for cleanup operations

Method Associated with the Unmounting Phase:

1. **componentWillUnmount():**
   - Explain that `componentWillUnmount()` is called just before the component is removed from the DOM.
   - Describe its purpose, which is to handle any necessary cleanup tasks such as canceling network requests, clearing timers, or unsubscribing from event listeners.