props vs state

props	state
props get passed to the component	state is managed within the component
Function parameters	Variables declared in the function body
props are immutable	state can be changed
props – Functional Components this.props – Class Components	useState Hook – Functional Components this.state – Class Components

setState

Always make use of setState and never modify the state directly.

Code has to be executed after the state has been updated ? Place that code in the call back function which is the second argument to the setState method.

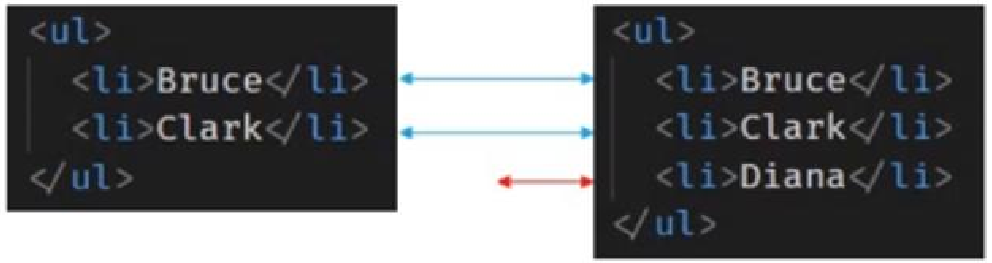
When you have to update state based on the previous state value, pass in a function as an argument instead of the regular object.

Conditional Rendering

1. if/else
2. Element variables
3. Ternary conditional operator
4. Short circuit operator

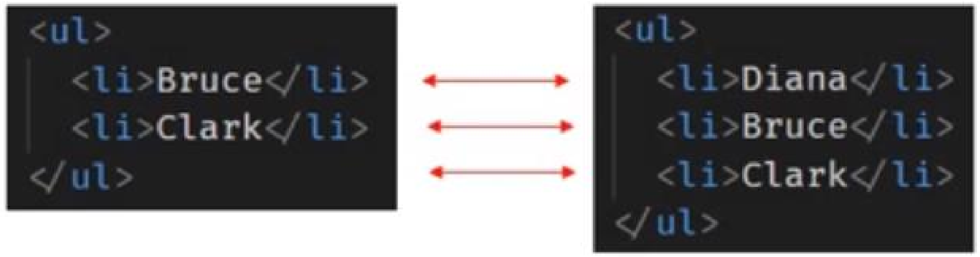
Lists and Keys

List without key attribute



```
<ul>
  <li>Bruce</li>
  <li>Clark</li>
</ul>
```

```
<ul>
  <li>Bruce</li>
  <li>Clark</li>
  <li>Diana</li>
</ul>
```

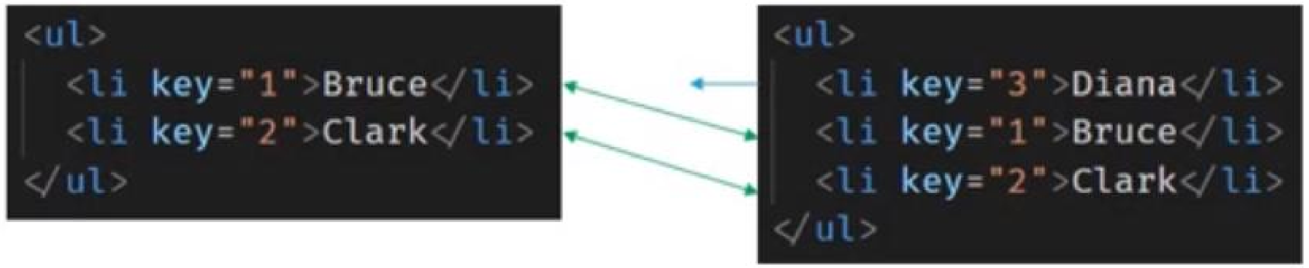


```
<ul>
  <li>Bruce</li>
  <li>Clark</li>
</ul>
```

```
<ul>
  <li>Diana</li>
  <li>Bruce</li>
  <li>Clark</li>
</ul>
```

Lists and Keys

List with key attribute



```
<ul>
  <li key="1">Bruce</li>
  <li key="2">Clark</li>
</ul>
```

```
<ul>
  <li key="3">Diana</li>
  <li key="1">Bruce</li>
  <li key="2">Clark</li>
</ul>
```

Lists and Keys

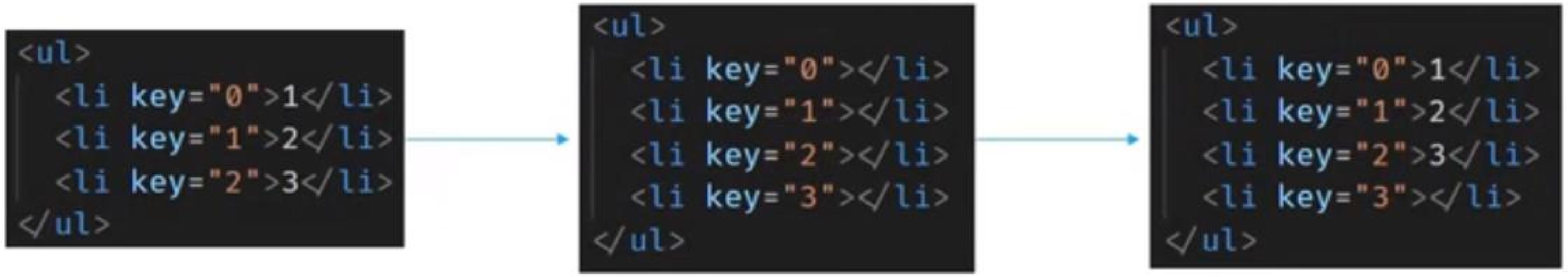
A “key” is a special string attribute you need to include when creating lists of elements.

Keys give the elements a stable identity.

Keys help React identify which items have changed, are added, or are removed.

Help in efficient update of the user interface.

Index as key anti-pattern



```
<ul>
  <li key="0">1</li>
  <li key="1">2</li>
  <li key="2">3</li>
</ul>
```

```
<ul>
  <li key="0"></li>
  <li key="1"></li>
  <li key="2"></li>
  <li key="3"></li>
</ul>
```

```
<ul>
  <li key="0">1</li>
  <li key="1">2</li>
  <li key="2">3</li>
  <li key="3"></li>
</ul>
```

Index as key

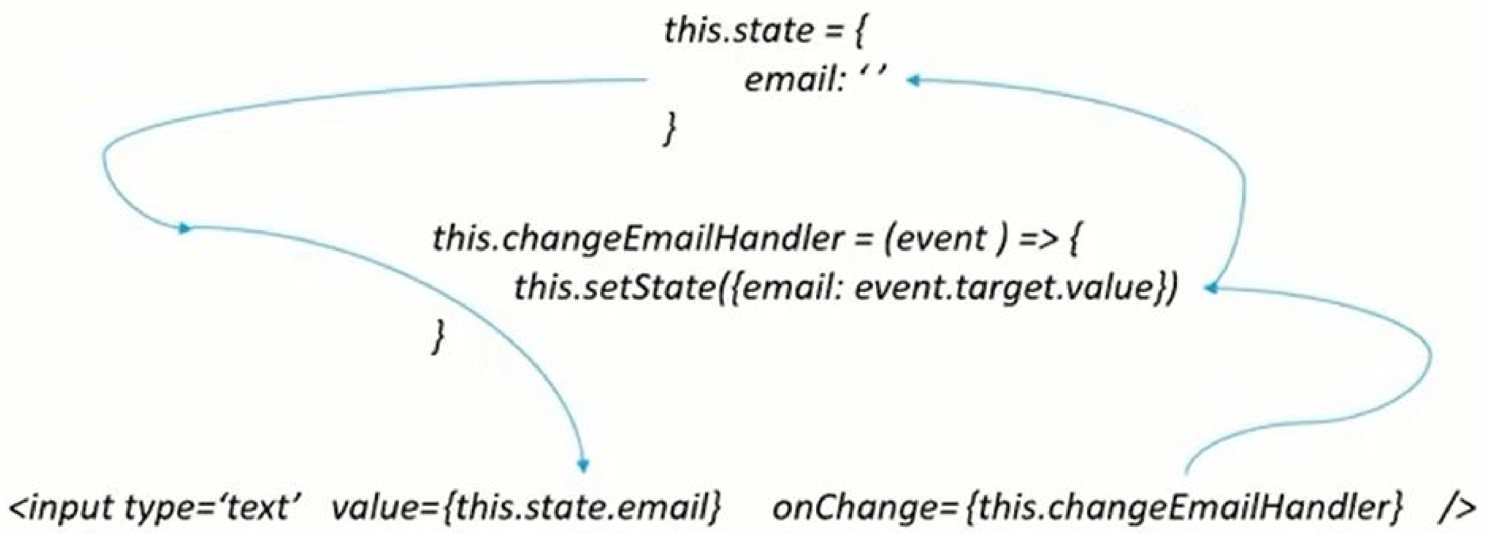
When to use index as a key?

1. The items in your list do not have a unique id.
2. The list is a static list and will not change.
3. The list will never be reordered or filtered.

Styling React Components

1. CSS stylesheets
2. Inline styling
3. CSS Modules
4. CSS in JS Libaries

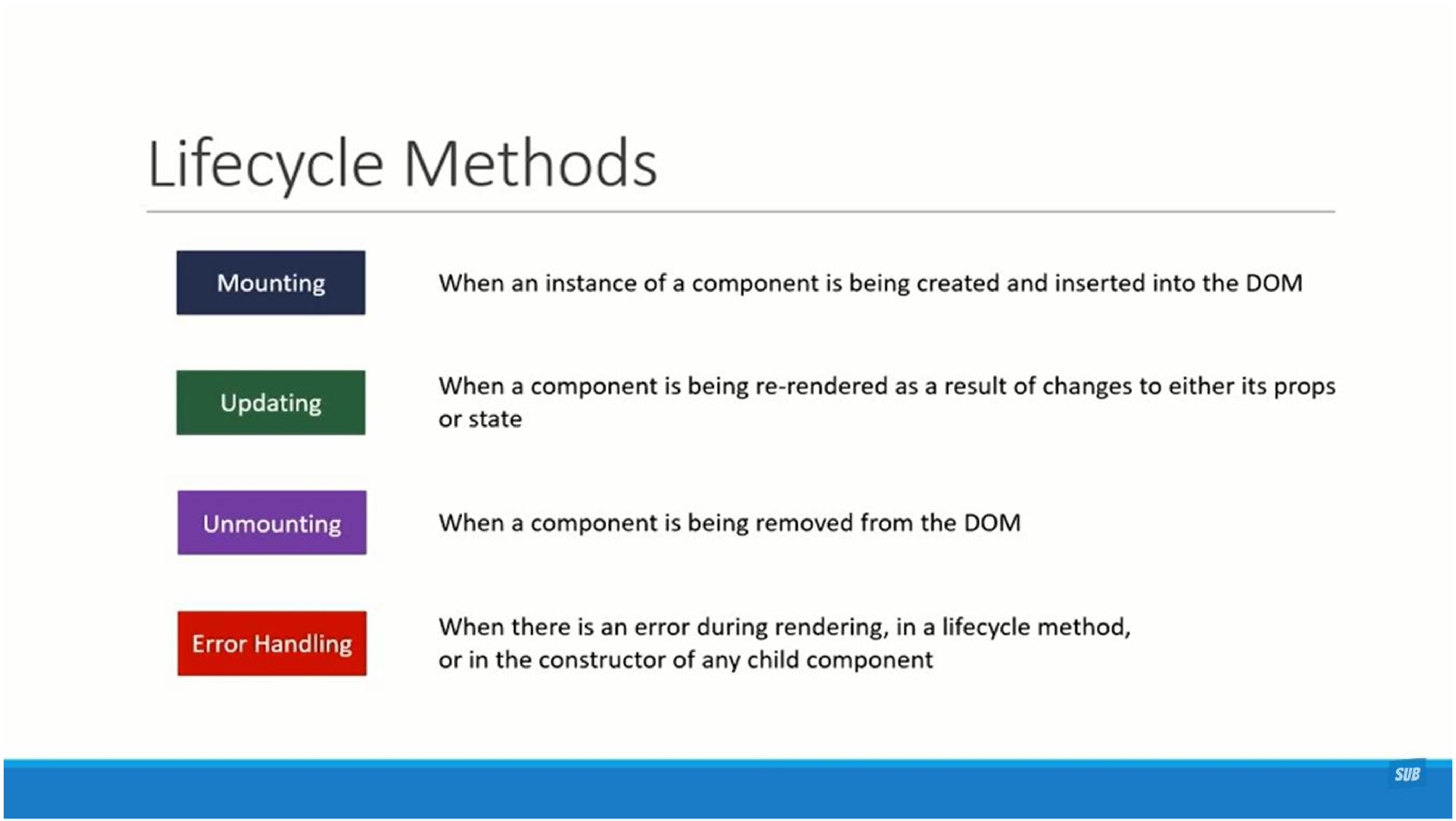
Controlled components



```
this.state = {
      email: ‘ ’
}

this.changeEmailHandler = (event ) => {
      this.setState({email: event.target.value})
}

<input type=‘text’   value={this.state.email}    onChange={this.changeEmailHandler}   />
```



Lifecycle Methods

Mounting	When an instance of a component is being created and inserted into the DOM
Updating	When a component is being re-rendered as a result of changes to either its props or state
Unmounting	When a component is being removed from the DOM
Error Handling	When there is an error during rendering, in a lifecycle method, or in the constructor of any child component

Lifecycle Methods

Mounting	*constructor, static getDerivedStateFromProps, render* and *componentDidMount*
Updating	*static getDerivedStateFromProps, shouldComponentUpdate, render, getSnapshotBeforeUpdate* and *componentDidUpdate*
Unmounting	*componentWillUnmount*
Error Handling	*static getDerivedStateFromError* and *componentDidCatch*

Mounting Lifecycle Methods

constructor(props)

A special function that will get called whenever a new component is created.

Initializing state
Binding the event handlers

Do not cause side effects. Ex: HTTP requests

super(props)
Directly overwrite this.state

SUB

Mounting Lifecycle Methods

constructor(props)

↓

static getDerivedStateFromProps(props, state)

When the state of the component depends on changes in props over time.

Set the state

Do not cause side effects. Ex: HTTP requests

Mounting Lifecycle Methods

constructor(props)

↓

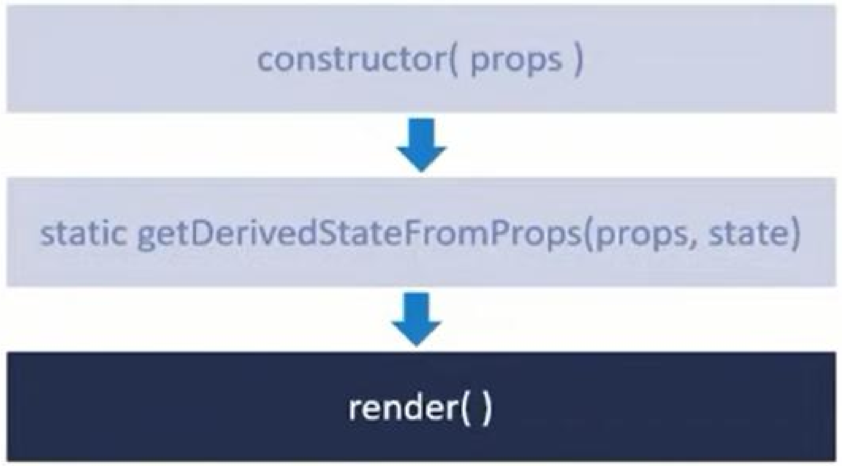
static getDerivedStateFromProps(props, state)

When the state of the component depends on changes in props over time.

Set the state

Do not cause side effects. Ex: HTTP requests

Mounting Lifecycle Methods



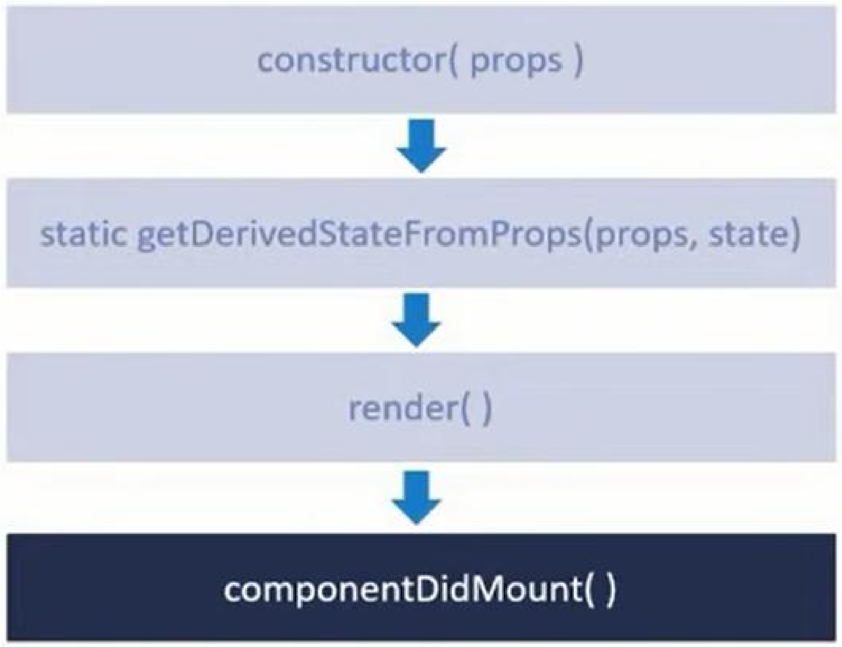
Only required method

Read props & state and return JSX

Do not change state or interact with DOM or make ajax calls.

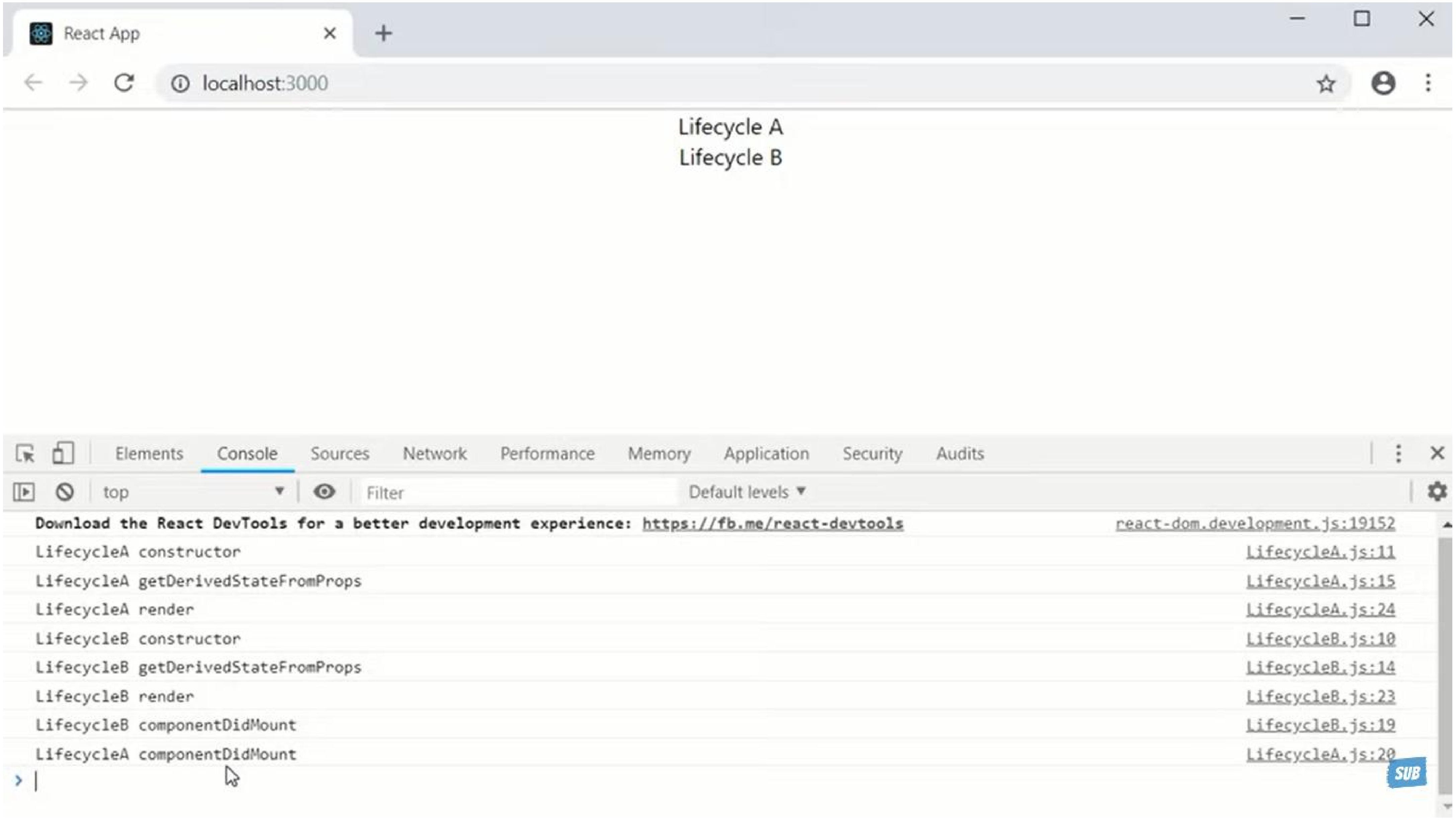
Children components lifecycle methods are also executed.

Mounting Lifecycle Methods



Invoked immediately after a component and all its children components have been rendered to the DOM.

Cause side effects. Ex: Interact with the DOM or perform any ajax calls to load data.



Lifecycle A
Lifecycle B

```
Download the React DevTools for a better development experience: https://fb.me/react-devtools   react-dom.development.js:19152
LifecycleA constructor                                                                                    LifecycleA.js:11
LifecycleA getDerivedStateFromProps                                                                       LifecycleA.js:15
LifecycleA render                                                                                         LifecycleA.js:24
LifecycleB constructor                                                                                    LifecycleB.js:10
LifecycleB getDerivedStateFromProps                                                                       LifecycleB.js:14
LifecycleB render                                                                                         LifecycleB.js:23
LifecycleB componentDidMount                                                                              LifecycleB.js:19
LifecycleA componentDidMount                                                                              LifecycleA.js:20
>
```

Updating Lifecycle Methods

static getDerivedStateFromProps(props, state)

Method is called every time a component is re-rendered

Set the state

Do not cause side effects. Ex: HTTP requests

Updating Lifecycle Methods

static getDerivedStateFromProps(props, state)

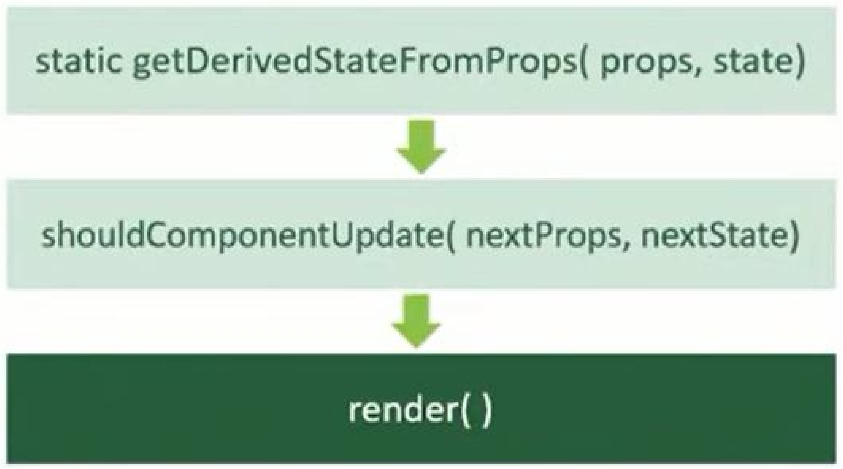
shouldComponentUpdate(nextProps, nextState)

Dictates if the component should re-render or not

Performance optimization

Do not cause side effects. Ex: HTTP requests
Calling the setState method

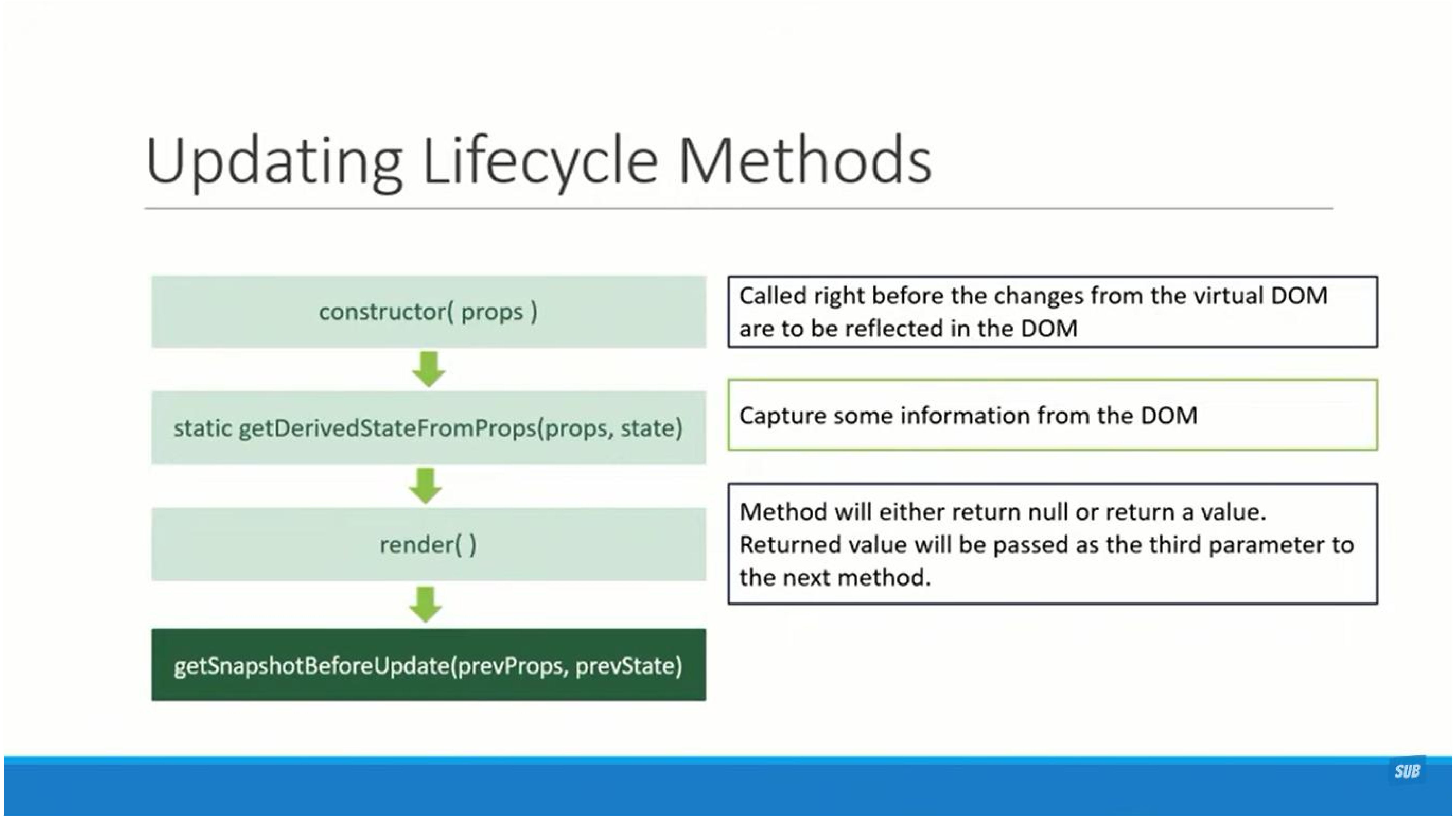
Updating Lifecycle Methods


static getDerivedStateFromProps(props, state)
shouldComponentUpdate(nextProps, nextState)
render( )

Only required method

Read props & state and return JSX

Do not change state or interact with DOM or make ajax calls.



Updating Lifecycle Methods

constructor(props)

↓

static getDerivedStateFromProps(props, state)

↓

render( )

↓

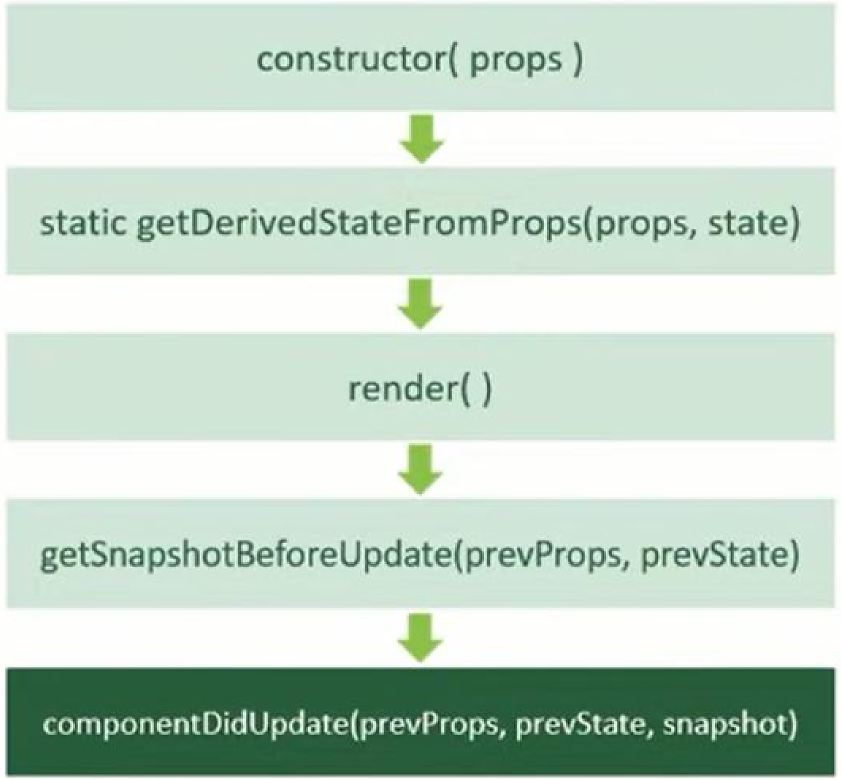
getSnapshotBeforeUpdate(prevProps, prevState)

Called right before the changes from the virtual DOM are to be reflected in the DOM

Capture some information from the DOM

Method will either return null or return a value. Returned value will be passed as the third parameter to the next method.

Updating Lifecycle Methods



constructor(props)
static getDerivedStateFromProps(props, state)
render( )
getSnapshotBeforeUpdate(prevProps, prevState)
componentDidUpdate(prevProps, prevState, snapshot)

Called after the render is finished in the re-render cycles

Cause side effects

Updating Lifecycle Methods

static getDerivedStateFromProps(props, state)

Method is called every time a component is re-rendered

Set the state

Do not cause side effects. Ex: HTTP requests

Updating Lifecycle Methods

static getDerivedStateFromProps(props, state)

↓

shouldComponentUpdate(nextProps, nextState)

Dictates if the component should re-render or not

Performance optimization

Do not cause side effects. Ex: HTTP requests
Calling the setState method

Updating Lifecycle Methods

static getDerivedStateFromProps(props, state)

↓

shouldComponentUpdate(nextProps, nextState)

↓

render( )

Only required method

Read props & state and return JSX

Do not change state or interact with DOM or make ajax calls.

Updating Lifecycle Methods

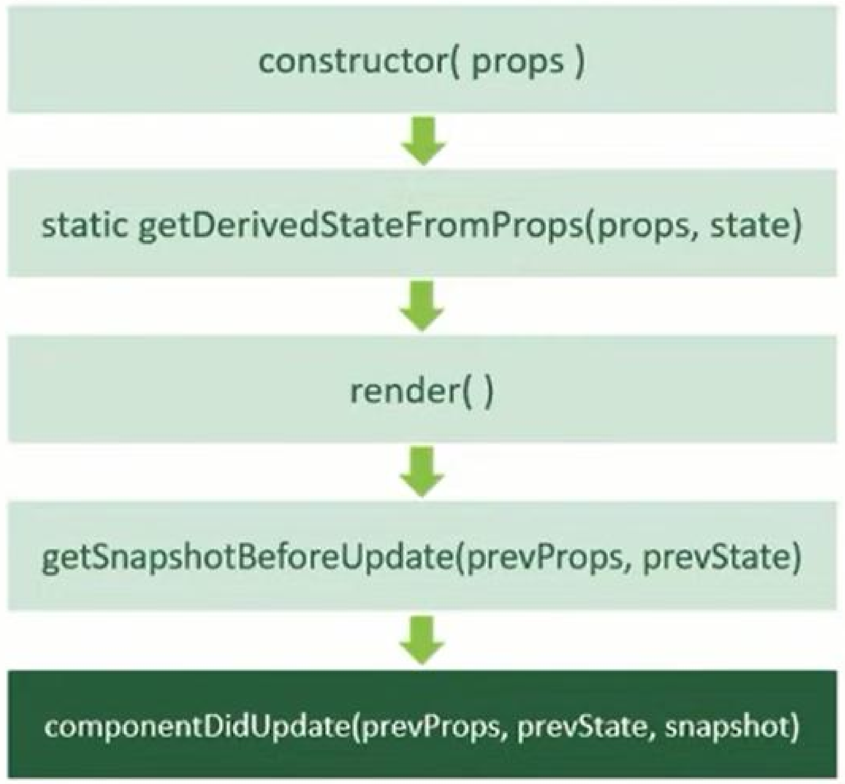
$$\begin{tikzcd}
\text{constructor(props)} \arrow[d] \\
\text{static getDerivedStateFromProps(props, state)} \arrow[d] \\
\text{render( )} \arrow[d] \\
\text{getSnapshotBeforeUpdate(prevProps, prevState)}
\end{tikzcd}$$

Called right before the changes from the virtual DOM are to be reflected in the DOM

Capture some information from the DOM

Method will either return null or return a value. Returned value will be passed as the third parameter to the next method.

Updating Lifecycle Methods


constructor(props)
static getDerivedStateFromProps(props, state)
render( )
getSnapshotBeforeUpdate(prevProps, prevState)
componentDidUpdate(prevProps, prevState, snapshot)

Called after the render is finished in the re-render cycles

Cause side effects

Unmounting Phase Method

componentWillUnmount( )

Method is invoked immediately before a component is unmounted and destroyed.

Cancelling any network requests, removing event handlers, cancelling any subscriptions and also invalidating timers.

Do not call the setState method.

Error Handling Phase Methods

static getDerivedStateFromError(error)

componentDidCatch(error, info)

When there is an error either during rendering, in a lifecycle method, or in the constructor of any child component.