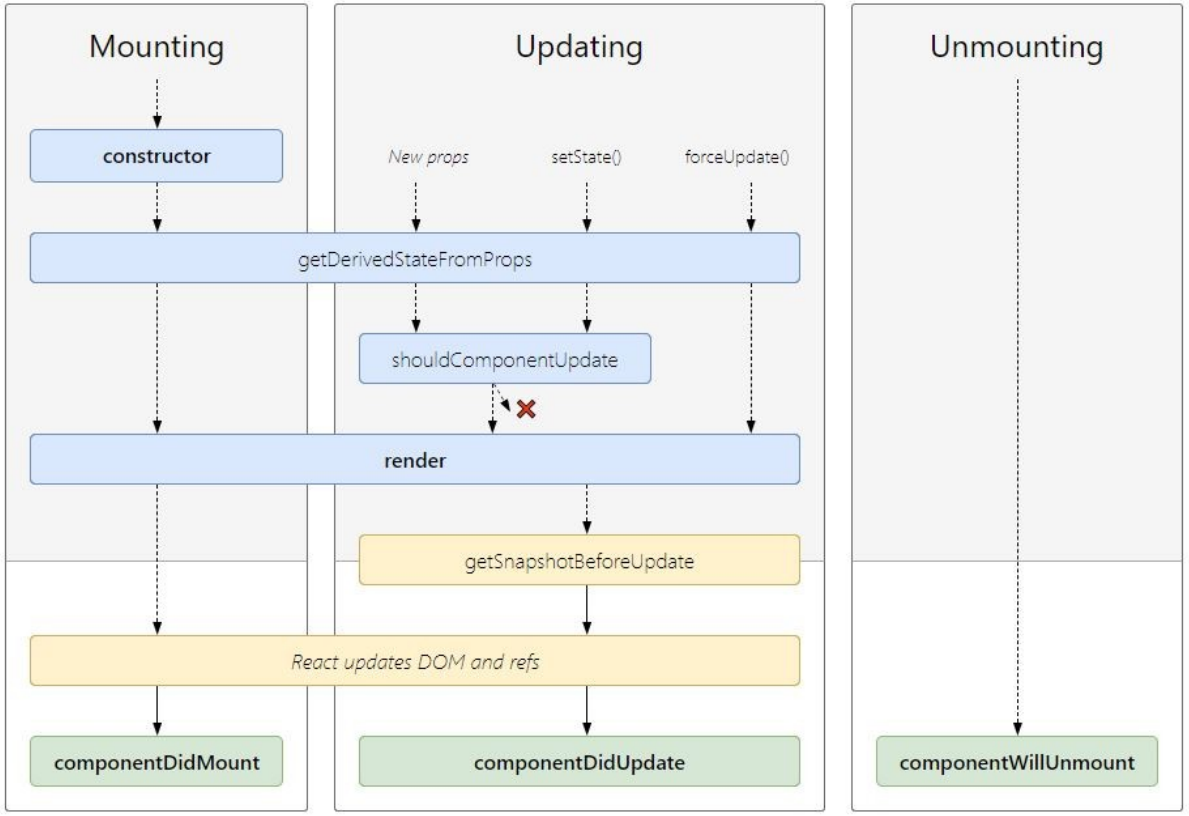
Lifecyle Methods

Each component has several “lifecycle methods” that you can override to run code at particular times in the process.

- Mounting
- Updating
- Error Handling
- Unmounting

Lifecyle Methods



Mounting

Mounting – Mounting is the process of creating an element and inserting it in a DOM tree.

Following methods are called in the following order when an instance of a component is being created and inserted into the DOM:-

- constructor()
- static getDerivedStateFromProps()
- render()
- componentDidMount()

constructor ( )

The constructor for a React component is called before it is mounted.

When implementing the constructor for a React.Component subclass, you should call super(props) before any other statement. Otherwise, this.props will be undefined in the constructor, which can lead to bugs.

React constructors are only used for two purposes:

- Initializing local state by assigning an object to this.state.

  Ex:- this. State = {name: “Rahul”}

- Binding event handler methods to an instance.

  Ex:- this.handleClick = this.handleClick.bind(this);

constructor ( )

```
constructor(props) {
    super(props);
    this.state = {
      name: "Rahul",
      roll: this.props.roll
    };
    this.handleClick = this.handleClick.bind(this);
  }
```

If you don’t initialize state and you don’t bind methods, you don’t need to implement a constructor for your React component.

static getDerivedStateFromProps( )

getDerivedStateFromProps is invoked right before calling the render method, both on the initial mount and on subsequent updates. It should return an object to update the state, or null to update nothing. This method exists for rare use cases where the state depends on changes in props over time. This method doesn't have access to the component instance. As its static method so *this* is not available inside this method.

Syntax:-

```
static getDerivedStateFromProps(props, state) {


}
```

render( )

The render() method is the only required method in a class component. It examines this.props and this.state .

It returns one of the following types:

React elements – These are created via JSX(Not required).

For example, <div /> and <App /> are React elements that instruct React to render a DOM node, or another user-defined component, respectively.

Arrays and fragments - It is used to return multiple elements from render.

Portals – It is used to render children into a different DOM subtree.

String and numbers - These are rendered as text nodes in the DOM.

Booleans or null - It renders nothing. (Mostly exists to support return test && <Child /> pattern, where test is boolean.)

Note - The render() function should be pure, meaning that it does not modify component state, it returns the same result each time it’s invoked, and it does not directly interact with the browser.

componentDidMount( )

componentDidMount() is invoked immediately after a component is mounted (inserted into the tree), after the render() method has taken place.

This method is executed once in a lifecycle of a component and after the first render.

Initialization that requires DOM nodes should go here.

This is where AJAX requests and DOM or state updates should occur. This method is also used for integration with other JavaScript frameworks and any functions with delayed execution such as setTimeout or setInterval.

The API calls should be made in componentDidMount method always.

Syntax:-

```
componentDidMount() {


}
```

Updating

Updating – Updating is the process of changing state or porps of component and update changes to nodes already in the DOM.

An update can be caused by changes to props or state. These methods are called in the following order when a component is being re-rendered:

- static getDerivedStateFromProps()
- shouldComponentUpdate()
- render()
- getSnapshotBeforeUpdate()
- componentDidUpdate()

static getDerivedStateFromProps( )

getDerivedStateFromProps is invoked right before calling the render method, both on the initial mount and on subsequent updates. It should return an object to update the state, or null to update nothing. This method exists for rare use cases where the state depends on changes in props over time. This method doesn’t have access to the component instance. As its static method so *this* is not available inside this method

Syntax:-

```
static getDerivedStateFromProps(props, state) {


}
```

shouldComponentUpdate()

Use shouldComponentUpdate() to let React know if a component's output is not affected by the current change in state or props, It means should React re-render or it can skip rendering? shouldComponentUpdate() is invoked before rendering when new props or state are being received. This method return true by default.

render() will not be invoked if shouldComponentUpdate() returns false.

Syntax: -

```
shouldComponentUpdate(nextProps, nextState){


}
```

render()

The render() method is the only required method in a class component. It examines this.props and this.state .

It returns one of the following types:

React elements – These are created via JSX(Not required).

For example, <div /> and <App /> are React elements that instruct React to render a DOM node, or another user-defined component, respectively.

Arrays and fragments - It is used to return multiple elements from render.

Portals – It is used to render children into a different DOM subtree.

String and numbers - These are rendered as text nodes in the DOM.

Booleans or null - It renders nothing. (Mostly exists to support return test && <Child /> pattern, where test is boolean.)

Note - The render() function should be pure, meaning that it does not modify component state, it returns the same result each time it's invoked, and it does not directly interact with the browser.

<u>getSnapshotBeforeUpdate()</u>

This method is called right before the virtual DOM is about to make change to the DOM (before DOM is updated), which allows our components to capture the current values or some information from the DOM(eg. Scroll Position) before it is potentially changed. Any value returned by this lifecycle will be passed as third parameter to componentDidUpdate() .

Syntax: -

```
getSnapshotBeforeUpdate(prevProps, prevState){


}
```

componentDidUpdate()

componentDidUpdate() is invoked immediately after updating occurs. This method is not called for the initial render.

This method is used to re trigger the third party libraries used to make sure these libraries also update and reload themselves.

componentDidUpdate() will not be invoked if shouldComponentUpdate() returns false.

Syntax:-

```
componentDidUpdate(prevProps, prevState, snapshot) {


}
```

If your component implements the getSnapshotBeforeUpdate() lifecycle (which is rare), the value it returns will be passed as a third “snapshot” parameter to componentDidUpdate(). Otherwise this parameter will be undefined.

Error Handling

These methods are called when there is an error during rendering, in a lifecycle method, or in the constructor of any child component.

- static getDerivedStateFromError()
- componentDidCatch()

static getDerivedStateFromError()

This lifecycle method is invoked after an error has been thrown by a descendant component. It receives the error that was thrown as a parameter and should return a value to update state.

Syntax:-

```
static getDerivedStateFromError(error){

}
```

componentDidCatch()

This lifecycle method is invoked after an error has been thrown by a descendant component.

Syntax:-

```
componentDidCatch(error, info) {

}
```

Where,

error - The error that was thrown.

info - An object with a componentStack key containing information about which component threw the error.

Unmounting

Unmounting – Unmounting is the process of removing components from the DOM.

This method is called when a component is being removed from the DOM:

- componentWillUnmount()

componentWillUnmount()

componentWillUnmount() is invoked immediately before a component is unmounted and destroyed.

Perform any necessary cleanup in this method, such as invalidating timers, canceling network requests, or cleaning up any subscriptions that were created in.

This is executed just before the component gets removed from the DOM.

Syntax:-

```
componentWillUnmount(){

}
```