Lifecycle Hooks

Mounting –

1. constructor()
2. getDerivedStateFromProps()
3. render()
4. componentDidMount()
5. The constructor() method is called before anything else, when the component is initiated, and it is the natural place to set up the initial state and other initial values.

The constructor() method is called with the props, as arguments, and you should always start by calling the super(props) before anything else, this will initiate the parent's constructor method and allows the component to inherit methods from its parent (React.Component).

1. getDerivedStateFromProps

The getDerivedStateFromProps() method is called right before rendering the element(s) in the DOM.

This is the natural place to set the state object based on the initial props.

It takes state as an argument, and returns an object with changes to the state.

### render

The render() method is required, and is the method that actually outputs the HTML to the DOM.

1. **componentDidMount**

The componentDidMount() method is called after the component is rendered.

This is where you run statements that requires that the component is already placed in the DOM.

## **Updating**

The next phase in the lifecycle is when a component is updated.

A component is updated whenever there is a change in the component's state or props.

React has five built-in methods that gets called, in this order, when a component is updated:

1. getDerivedStateFromProps()
2. shouldComponentUpdate()
3. render()
4. getSnapshotBeforeUpdate()
5. componentDidUpdate()
   1. getDerivedStateFromProps

Also at updates the getDerivedStateFromProps method is called. This is the first method that is called when a component gets updated.

This is still the natural place to set the state object based on the initial props.

* 1. shouldComponentUpdate

In the shouldComponentUpdate() method you can return a Boolean value that specifies whether React should continue with the rendering or not.

The default value is true.

* 1. render

The render() method is of course called when a component gets updated, it has to re-render the HTML to the DOM, with the new changes.

* 1. getSnapshotBeforeUpdate

In the getSnapshotBeforeUpdate() method you have access to the props and state before the update, meaning that even after the update, you can check what the values were before the update.

If the getSnapshotBeforeUpdate() method is present, you should also include the componentDidUpdate() method, otherwise you will get an error.

* 1. **componentDidUpdate**

The componentDidUpdate method is called after the component is updated in the DOM.

## **Unmounting**

The next phase in the lifecycle is when a component is removed from the DOM, or unmounting as React likes to call it.

React has only one built-in method that gets called when a component is unmounted:

* componentWillUnmount()

### componentWillUnmount

The componentWillUnmount method is called when the component is about to be removed from the DOM.

Flux :

MVC – model view controller

Two way data binding it supports source to target and target to source data flow.

Limitations –i. keep track all the bindings from source to target and target to source

ii. when application length increases and you may have to add new modules then it creates problem, it becomes complex

MVP – model view presenter – tight coupling between view and presenter

MVPM – Model view presenter model – works exactly same as MVC

MVVM – Model view view Model – it is designed by Microsoft by adopting MVPM

Flux is an architectural pattern that is introduced by facebook and it supports one way data flow.

MVC implementations

Java – spring

.net – ASP .net

Node – Express

Flux implementation – facebook flux, redux, reflux,mcfly,fluxory

It supports one way data flow i.e when the source changes automatically changes will be there in target. If new functionality will be added only new view need to be created model can remain same.

MVC – browser->view->controller->model->DB

Controller is responsible to control the flow between view to model and model to view

Flux architecture – Action – Dispatcher – store – View ->Action

| | -------------------------------------

Action – it is a helper method that facilitates passing data to dispatcher

Dispatcher – receives action and broadcast data / payload to store via registered callback

Store – container for application’s state and logic, it has a callback registered with dispatcher

Controller view – react comp that grabs the state changes from store and pass it down to descendent child via props

Presentational view – it is a react comp that will receive props from controller view and it will be used to present that on a view.

Changes in state in the store if you want automatically reflect in controller view then store has to extend EventEmitter and need to publish change event and that event need to be subscribed by controller view.