Mounting & Unmounting → States / Lifecycle Methods –

Example of clock (tick)

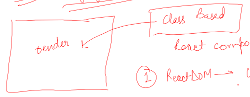
→ Hello World
12 : 48 : 53 → update every second
↳ ReactDOM . render ( ) × n → calling multiple times
↳ multiple react elements instantiated

① clock → reusable → functional Component
→ Class Based Component.

② clock. → State add → Lifecycle Methods

State and Lifecycle Methods

Ⓘ



React DOM

clock ( )

React component

① ReactDOM → clock ( ) gets called.

② clock ( ) constructor (props) → base constructor (props) gets called + this.state values get initialized

③ clock ( ) → render ( ) gets called which returns react component with initial state.

④ clock ( ) → returns this react component and gets rendered in ReactDOM.

⑤ componentDidMount ( ) ↳ gets called as soon as the the clock ( ) gets rendered at React DOM → starts the timer and records the timer id.

⑥ every second the tick ( ) method is called → where this.setState ( ) is called → which notifies a change in the initial state → date → a new date.

⑦ Then the render ( ) will be called again to rerender the changed UI.

⑧ whenever the DOM destroys this React component → componentWill-Unmount ( ) gets called which stops the timer.

★ this.state and this.setState ( ) are used to record initial state of React component and notify and change the initial state of the component and rerender ( ) the UI.

★ Lifecycle Methods → ① componentDidMount ( )
↳ gets called when the component gets rendered on the React DOM for the first time

② componentWillUnmount ( )
↳ when the react component gets destroyed from DOM.

→ Lifecycle methods are special methods available in React component which determines the life of the component. i.e, used to mount and unmount the component.

→ idea is to delete resources when components are destroyed so that they can be reused.

★ Three things to Remember while using React : –
① Do not change the State directly.
② The state changes may be asynchronous.
↳
③ React shallow merges the changes with multiple SetState calls for multiple State variables.

☆ Three things to Remember while using React :-

1. Do not change the State directly.
2. The state changes may be asynchronous.
   ↳
3. React shallow merges the changes with multiple setState calls for multiple State variables.

☆ Data Flows down ~~the~~ parent to ~~the~~ children →
either parent or children does not know whether its parent or its children is Stateful or Stateless

☆ Each component is the owner of its own state & props but can transfer its state / props data downwards to ~~the~~ children,