**State and Life Cycle in Class Component**

Class component is statefull. When we pass parameter to this component we can receive these parameter into this.

To work modular way we will write all the components into separate file. component name should be in capital letter.

**Props** will be changed outside of the component where we mainly pass data as parameter from outside.

But **state** will be changed inside component. Mainly component data will be changed and changing functionality will be remained inside component.

Component is like database. It is functional and it is state full. State is component data. To changed component data we should write code react way.

React know only state data means component data. But props is mainly from outside data. When we pass data from outside react only one showed this data. But when we use state data then react know this state data and changed the UI mainly data of the component when the value of the state will be changed.

**React.Component** has **state** property. State is simple object. It hold the data.

To initialize something in class we need to use constructor.

When state is change then react will reacted and call the render method again in class component

import React from "react";

class Clock extends React.Component {

  // constructor

  constructor(props) {

    super(props);

    this.state = {

      date: new Date(),

    };

  }

  render() {

    return (

      <h1 className="heading">

        <span className="text">

          {this.state.date.toLocaleTimeString(this.props.locale)}

        </span>

      </h1>

    );

  }

}

export default Clock;

we have a method **set state** in class component to change the value of the state. When state value is changed by the setState method the UI data will be changed and re render.

**setState** is another method of **React.Component**

when UI is showed into browser then **componentDidMount()** method will be called. Mainly when react compare the virtual dom and show the updated data into screen then **componentDidMount()** will be called

  // constructor

  constructor(props) {

    super(props);

    this.state = {

      date: new Date(),

    };

  }

  // called this method when UI is showed

  componentDidMount() {

    setInterval(() => {

      this.setState({ date: new Date() });

    }, 1000);

  }

When component will be unmounted then this **componentWillUnmount()** method will be called or fired. Mainly when we go to another route then this timer page need to clear otherwise this will run always.

  // clear the mount of the component

  componentWillUnmount() {

    clearInterval(this.clockTimer);

  }

**We can write state without constructor:**

  // constructor

  constructor(props) {

    super(props);

    this.state = {

      date: new Date(),

    };

  }

Instead we can write directly when we don’t have any props to this constructor then we can write

state = { date: new Date() };

**asynchronous setState: callback function inside setState() method**

when we need previous state inside **setState() ,**we should use callback function and parameter as previous state below is example:

  tick(){

    this.setState((state, props) => state.date = new Date());

  }

**Merge state:**

When we set a state, mainly this state merge with the previous whole state, react only compare the new and previous state and merge the changes.

**Note:** don’t perform any operation with the state directly because when any single change of the state will be counted then component will be render. Unnecessary render will be occurred. So always copy the state into new variable and perform operation with this copied variable. After performing operation then change the state and render when work is done.

**The data flows down or one way data flow into react parent to child component :**

When we pass data to the any component, component only know the data but it is unknown that from where data is coming. Mainly when we pass data to any component mainly we pass only value not reference that’s why we can not change the parent data whose data we passed to the component. That’s way react only allowed one way data pass.

**Pass by value or unidirectional**

When parent mainly source is changed then only inside child will be changed. But when child will be changed then there is not effect on the parent value