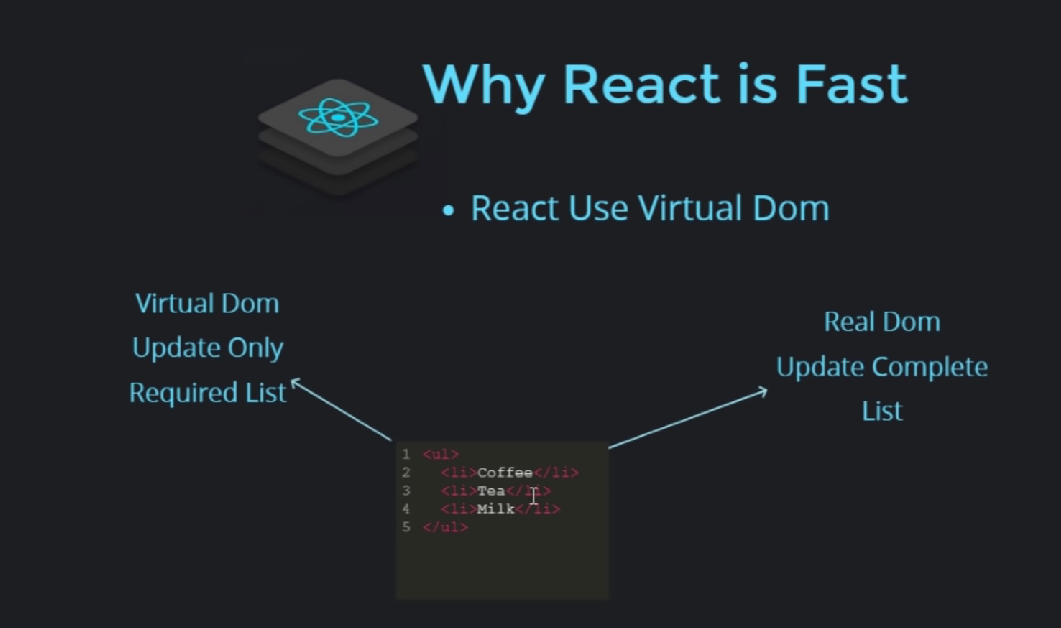
**REACT JS**



**JSX :**

JSX **stands for JavaScript XML**. JSX allows us to write HTML in React. JSX makes it easier to write and add HTML in React.



// The create-react-app is an excellent tool for beginners, which allows you to create and run React project very quickly.

*npx create-react-app blog*

// Command to start react app

npm start

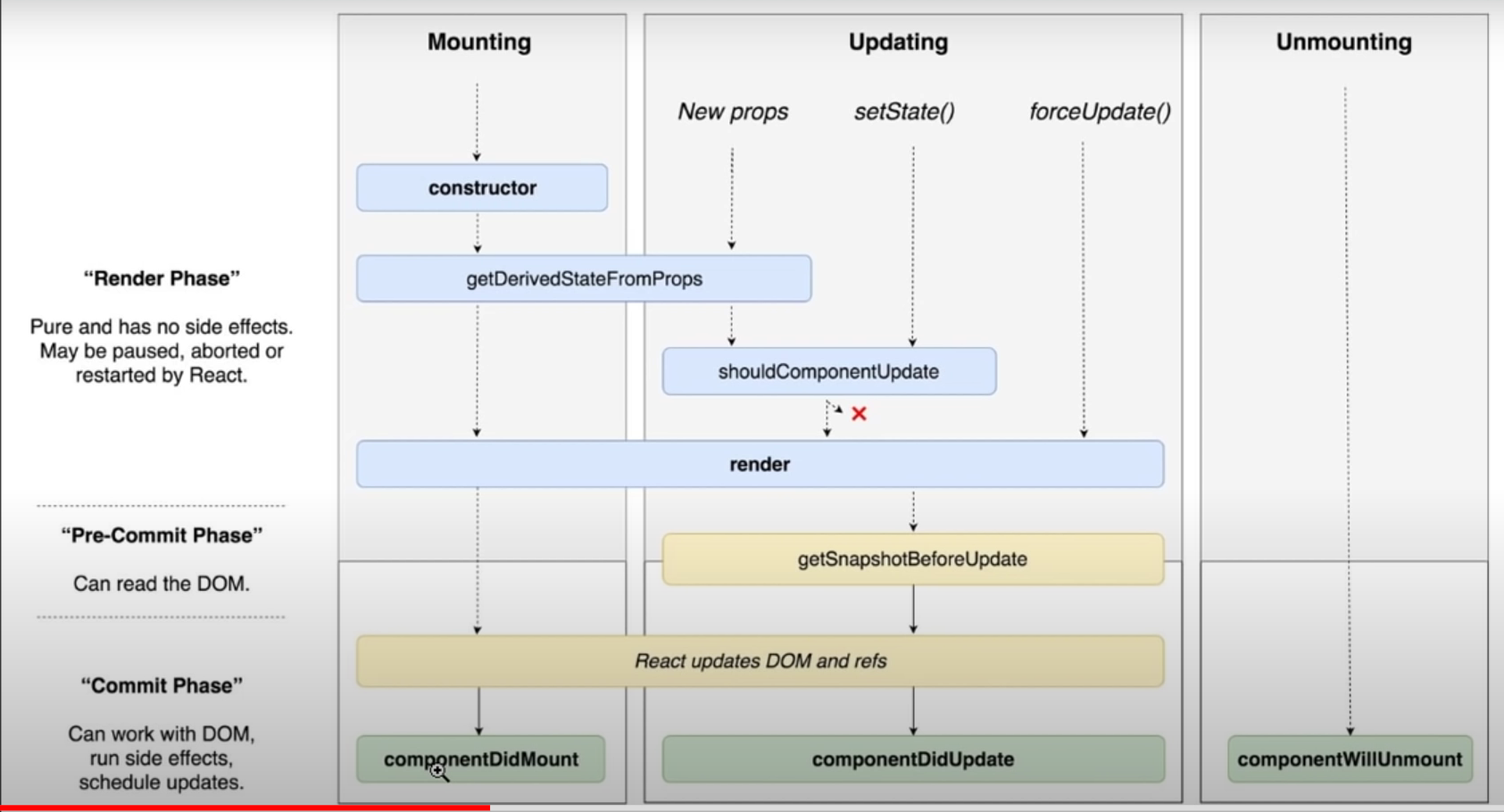
Life Cycle:

In react class components called step by step

1. *constructor*
2. *render*
3. *componentDidMount*

when the state of any variable or object is updated or when new props are added.

Then the *render* page is updated (re-rendered) and *componentDidUpdate* is called.



*render*():

**Rendering an Element in React:**In order to render any element into the Browser DOM, we need to have a container or root DOM element.

Render page re-render when the state of any variable or object is updated or when new props are added.

When the state of object is changed it is necessary to re-render the page (*the changes must be updated on the page*).

Render return the html page.

render returns twice in react js?

The reason why this happens is an intentional feature of the React StrictMode. It only happens in development mode and should help to find accidental side effects in the render phase.

*componentDidMount():*

API call is recommended in componentDidMount () to avoid unnecessary re-rendering and code complexity, **it's better to call API after render**()

(As componentDidMount run after render).

*componentDidUpdate():*

  The **componentDidUpdate**() method allows us to execute the **React** code when the component is updated.

when we click on button the state is updated hence the render page is updated (re-rendered) and

componentDidUpdate is called.

**shouldComponentUpdate():**

shouldComponentUpdate if it returns false the page not render as well as componentDidUpdate not called

    shouldComponentUpdate if it returns true the page render as well as componentDidUpdate called

*It decides the page will render and update or not if false then no if true then yes*

*It can stop rendering.*

**Hooks:**

*By using hook we can use class components features in functional component such as state, life cycle, pure component, etc.*

Hook start have use.

**useEffect:**

The useEffect Hook allows you to perform side effects in your components.

useEffect re-render when the state of any variable or object is updated or when new props are added.

**There are three ways to include CSS in React js:**

1] <p className="primary">

                Style in react js 1

            </p>

2] <p style={{fontSize:"35px" , color:"orangered"}}>

                Style in react js 2

            </p>

3] // First create this file in directory

import Style from "./custom.module.css";

<p className={Style.success}>

                Style in react js 3

            </p>

***Install Bootstrap in React js:***

npm install react-bootstrap bootstrap



***Why we use map instead of loop?***

Because return statement supports map but does not support loop.

**Fragments**:

**Fragments** let you group a list of children without adding extra nodes to the DOM.

There are two ways to do it:

1] <>

            <h1> Sumit </h1>

            <h2> OP </h2>

        </>

By making it blank

2] import { Fragment } from "react";

<Fragment>

            <h1> Sumit </h1>

            <h2> OP </h2>

        </Fragment>

# Components

Pure Component:

Let suppose we give state component name “sumit” and on button click we check the state of name and re-rendering it again and again.

Pure component matches the before value of state and re-entered value of state name if it is same then it does not re-render.

It is waste of time and performance if the values are same.

Same for props if state of prop changed it re-render else does not re-render.

useMemo:

The useMemo Hook only runs when one of its dependencies update.

This can improve performance.

Ref:

Use it as minimum as possible.

If we want to modify directly DOM then we use use ref.

For class we use { *createRef* }.

this.input = createRef();

For function we use {useRef}.

let input = useRef();

forwardRef:

forwardRef is a function used to pass the ref to a child component.

What if input box and button both in different components then we use forwardRef.

In React, controlled components refer to components that have their state and behavior controlled by the parent component. These components rely on props passed down from the parent component to update their state and behavior. Uncontrolled components refer to components that manage their own state internally.

*The preventDefault() method cancels the event if it is cancelable, meaning that the default action that belongs to the event will not occur.*

Controlled Components:

Controlled Components are managed by the state Components.

Uncontrolled Components:

These Components are managed by the ref.

# Routing

*For components we make different pages. And apply link to these pages so every functionality has its own page.*

// *https://v5.reactrouter.com/web/guides/quick-start*

All component children of <Routes> must be a <Route> or <React.Fragment>

Dynamic Routing:

# REDUX

When state is used in the multiple components then we have to use redux.

We also use probe drilling but it make code very complex and for large projects it is not possible.

// This command is for babel error coming in console

npm install --save-dev @babel/plugin-proposal-private-property-in-object