React JS PropTypes

* **In React JS, props, which stand for “properties”, pass data from one component to another.**
* **In React JS, If a component receives the wrong type of props, it can lead to bugs and unexpected errors in your app.**
* **As we all know that, JavaScript does not have a built-in type-checking solution, many developers use extensions such as TypeScript and Flow.**
* **However, React has an internal mechanism for props validation called prop types (type checking).**
* **In React JS, We pass different types of information, such as integers, strings, arrays, etc., as props to the components.**
* **In React JS, we can either create default props or pass the props directly as attributes of the components.**
* **In React JS, we passed props from outside a component and used them inside that component.**
* **But did we have to check what kind of values we are getting inside our component via props.**
* **Before React 15.5.0, proptypes are available in the React package, but in later versions of React, you need to add a dependency to your project.**

Wide Range Of Validators

* **PropTypes.array,**
* **PropTypes.bool,**
* **PropTypes.func,**
* **PropTypes.number,**
* **PropTypes.object,**
* **PropTypes.string,**
* **PropTypes.symbol,**

React Arrow Function

* **It gets shorter! If the function has only one statement, and the statement returns a value, you can remove the brackets and the return keyword.**
* **We can also create arrow functions with parameters.**
* **Arrow functions were introduced in ES6.**

Class Component In React JS

* **In React JS, these components are simple classes (made up of multiple functions that add functionality to the application).**
* **In React JS, all class based components are child classes for the Component class of ReactJS (React.Component).**
* **In React JS, the class must implement a render() member function which returns a React component to be rendered, similar to a return value of a functional component.**
* **In React JS, component name always starts with Capital Letter e-g: <App/> not <app/>**
* **In React JS, If you write tag in lowercase like <div/> then react treats this as a DOM tags but If you write tag in First letter uppercase like <App/> then it represents a react component.**

Short-Cuts Of Creating Functional Component In VS Code

* **rfc – react functional component**
* **rfce – react functional component with export**
* **rfcp – react functional component with prop types**
* **rafc – react functional component with arrow function**
* **rafce – react functional component with arrow function and export**
* **rafcp – react functional component with arrow function and prop types**

CONSTRUCTOR IN REACT JS

* **In React JS, the constructor is a method used to initialize an object’s state in a class.**
* **In React JS, constructor automatically called during the creation of an object in a class.**
* **In React JS, constructor’s concept of a constructor is the same in React.**
* **In React JS, If you don’t initialize state and you don’t bind methods, you don’t need to implement a constructor for your React component.**
* **In React JS, when you implement the constructor for a React component, you need to call super(props) method before any other statement.**
* **In React JS, If you do not call super(props) method, this.props will be undefined in the constructor and can lead to bugs.**

he setState() Method

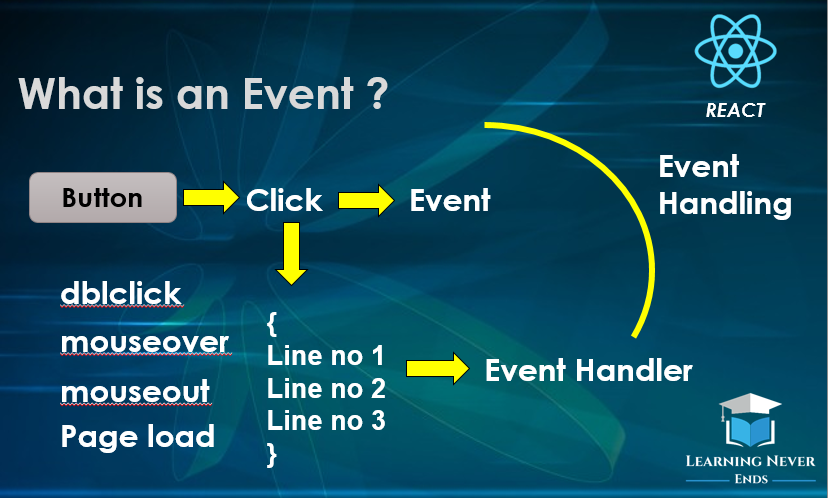
* **In React JS, state can be updated in response to event handlers, server responses or props changes, this is done using the setState() method.**
* **In React JS, the setState() method enqueues all of the updates made to the component state and instructs React to re-render the component and its children with the updated state.**
* **In React JS, always use the setState() method to change the state object, since it will ensure that the component knows it’s been updated and calls the render() method.**

## ifference B/W Props And State

| **PROPS** | **STATE** |
| --- | --- |
| **In React JS, Props get passed to the component.** | **In React JS, State is created and managed within the component.** |
| **Function Parameters.** | **Variables.** |
| **Props are immutable / un-changeable** | **State is mutable / changeable.** |
| **props -> Functional Components** | **useState Hook –> Functional Components.** |
| **this.props -> Class Components** | **this.state -> Class Component** |

What is Destructuring In React JS ?

* **Destructuring is a characteristic of JavaScript, It is used to take out sections of data from an array or objects, We can assign them to new own variables created by the developer.**
* **In destructuring, It does not change an array or any object, it makes a copy of the desired object or array element by assigning them in its own new variables, later we can use this new variable in React (class or functional) components.**



* **In React JS, Events are actions or occurrences that happen in the system you are programming, which the system tells you about so your code can react to them.**
* **Most often user of a website generates events.**

mportant Note

* **These 3 phases or life cycle only applied to class components not on functional component.**
* **There is another way of applying life cycle on functional components are called hooks.**

Mounting

* **In React JS, Mounting means putting elements into the DOM.**
* **In this phase, an instance of a component is being created and inserted into the DOM.**

Updating

* **In React JS, 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.**

Unmounting

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

The Component Lifecycle

* **In React JS, each component has several “lifecycle methods” that you can override to run code at particular times in the process.**

Methods In Mounting

**React has four built-in methods that gets called, in this order, when mounting a component:**

1. **constructor()**
2. **getDerivedStateFromProps()**
3. **render()**
4. **componentDidMount()**

**In React JS, the render() method is required and will always be called, the others are optional and will be called if you define them.**

Methods of Updating

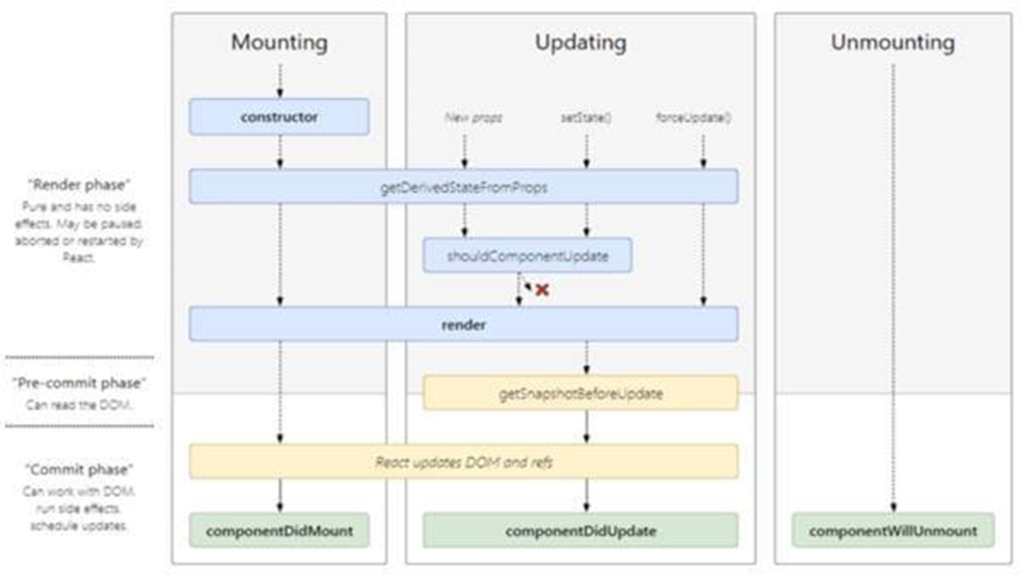
**In React JS, React has five built-in methods that gets called, in this order, when a component is updated:**

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

**In This phase, the render() method is also required and will always be called, the others are optional and will be called if you define them.**

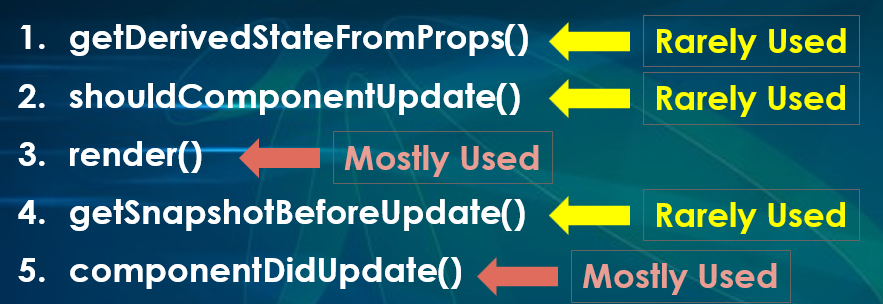
Method Of Unmounting

* **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()**

****

## Updating Phase

* **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()

* **It is used when the state of the component depends on changes in props over time.**
* **Lets say, you have a component but the initial state of the component depends on the props being passed to the component, in this scenario you can used this method to set the state.**
* **This is very rarely used method.**
* **It is invoked right before calling the render function, both on the initial mount and on subsequent updates.**
* **We cannot use this keyword inside this function, it means we cannot update the state by using this.setState method.**
* **It should return an object to update the state, or null to update nothing.**
* **This method is called every time a component is re-rendered.**

### **Syntax getDerivedStateFromProps()**

**static getDerivedStateFromProps(props, state){**

**}**

## 2. 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 not ?**
* **shouldComponentUpdate() is called before render method.**
* **This method return true by default.**
* **Render() method will not be called if shouldComponentUpdate() returns false.**
* **We cannot use HTTP requests (API calls) in this method and also we cannot call the setState() Method.**
* **This is Rarely used method according to React documentation.**

## Unmounting Phase In React Life Cycle

* **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 is the only method that executes in unmount phase.**
* **Just before the component gets removed from actual DOM, this method gets called.**
* **Along with removal of this component from DOM tree, all children of this component also gets removed automatically.**
* **The componentWillUnmount() method allows us to execute the React code when the component gets destroyed or unmounted from the DOM (Document Object Model)**
* **This method is called during the Unmounting phase of the React Life-cycle i.e before the component gets unmounted.**
* **All the cleanups such as invalidating timers, canceling network requests, or cleaning up any subscriptions that were created in componentDidMount() should be coded in the componentWillUnmount() method block.**
* **Cleanup activities helps in improving performances, memory leakages and maintain security.**
* **You should not call setState() in componentWillUnmount() because the component will never be re-rendered.**