React js
Lecture 1

Agenda

- Introduction about React
- Single page application
- Virtual Dom
- Environment setup
- React app structure
- What is JSX ?
- Building Reusable components
- Component lifecycle methods
- State and Props.
- Handling events

an open-source JavaScript library that is used for building user interfaces specifically for single-page applications.

React first deployed on Facebook's newsfeed in 2011 and on Instagram.com in 2012.

React allows developers to create large web applications that can change data, without reloading the page. The main purpose of React is to be fast, scalable, and simple

Last version :18.1.0

Why React?

1-High Performance
ReactJS main feature i.e Virtual DOM that results in high performance of the app

2-Easy to learn.

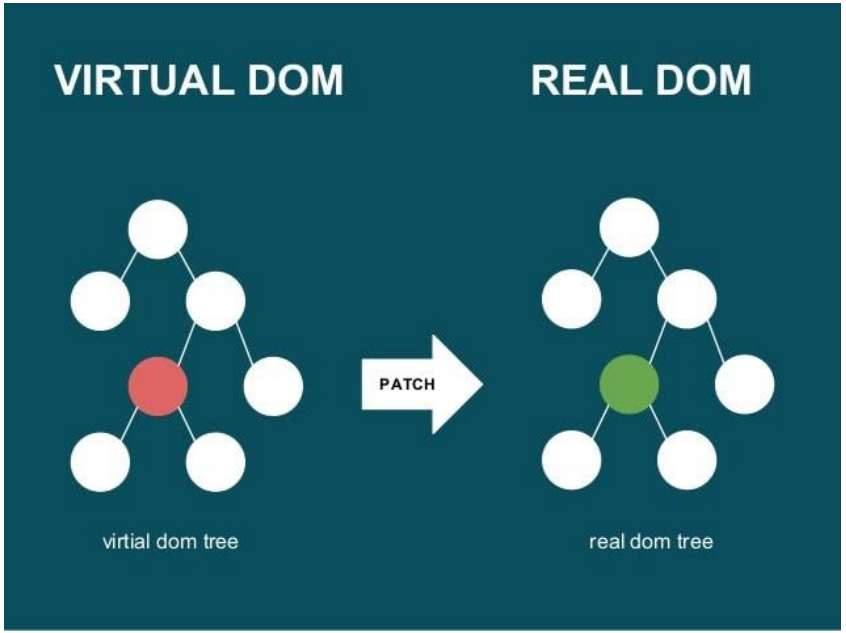
3-Easy Transition to react native.

4-Component Style Architecture.

5-Reusable Component.

Traditional

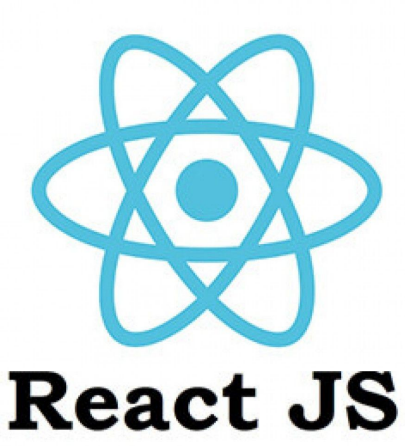
Single Page Application



Virtual Dom?

The virtual DOM (VDOM) is a programming concept where an ideal, or “virtual”, representation of a UI is kept in memory and synced with the “real” DOM by a library such as ReactDOM.whenever there is a change in state of any element, a new Virtual DOM tree is created. This new Virtual DOM tree is then compared with the previous Virtual DOM tree and make a note of the changes. After this, it finds the best possible ways to make these changes to the real DOM. Now only the updated elements will get rendered on the page again.

Getting Started



- Install node js.
- Create new react app : npx create-react-app appName
- Enter your folder and run your app : npm start

What is JSX?

Stands for javascript xml.JSX is an XML/HTML-like syntax used by React that extends ECMAScript so that XML/HTML-like text can co-exist with JavaScript/React code. The syntax is intended to be used by preprocessors (i.e., transpilers like Babel) to transform HTML-like text found in JavaScript files into standard JavaScript objects that a JavaScript engine will parse.

With JSX :about

Without JSX : React.createElement(“a”,{href:”#”},”about”)

Class component vs Functional component

A **functional**(a.k.a. **stateless**) component is just a plain javascript function which takes props as an argument and returns a react element.

A stateless component has no state(obvious, isn't it?), it means that you can't reach `this.state` inside it. It also has no lifecycle so you can't use componentDidMount and other hooks.

A **class** component should be used whenever you need to work with **state**, it might be relay or internal react state. Whenever you need to fetch data before you render your component you should call a fetchData function in componentDidMount.

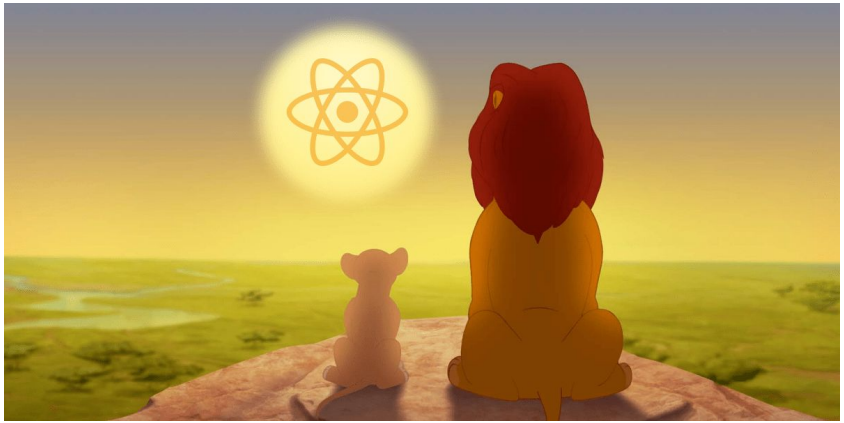
Class component vs Functional component

a functional component has no state, no lifecycle methods and it’s easy to write(plain function), a class component has state, lifecycle methods and React creates an instance of a class component every time React renders it. If you don’t need to use state or lifecycle I would recommend you to use a function component, but If there’s a chance that you need one of those things(state, lifecycle methods) I would suggest you to use class component

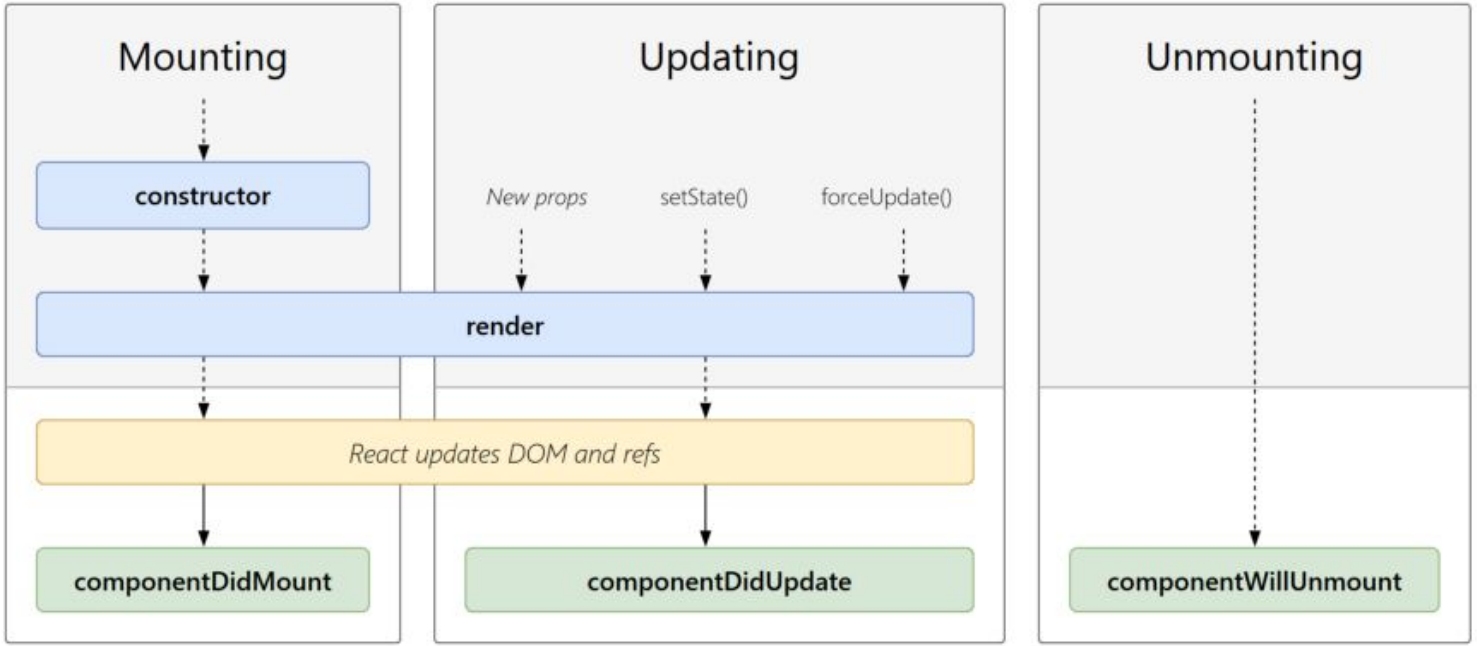
Component lifecycle

Each component goes through three phases:

1. **Mounting**
2. **updating**
3. **unmounting**



Component lifecycle



Props and state

PROPS	STATE
Props are used for passing data to child components	State is used for defining the shape of data both initially and upon user interaction.
Props are usually passed down from “above” parent components	State is created in the component, it gets its initial data in the `constructor()` method
Props are immutable to the component receiving them. You don't change props passed to a component from within the component	State is changeable, React uses the `setState()` method to update the object of a state. State can only be mutated by the component that contains the state. It is private in this sense.

Handling events

```
<button onClick={adduser}>Click
Me</button>
```

```
<button onClick={() =>
this.handleClick()}>Click Me</button>
```

Thank you