4---------------------------------------X---------------------------------------

**What is a difference between Java script and ES6?**

Answer -Basically there is no **difference between JavaScript and ES6**(ECMA Script 6).

ECMA Script is the official name of **JavaScript**. **ES 6** is the version of **JavaScript** 2015.

For making **JS** popular ECMA Script was named to **JavaScript** because of popularity of JAVA.

**----------------------------------------------X-----------------------------------------**

Is typescript same as javascript?

Ans-**TypeScript** is an open source syntactic superset of **JavaScript** that compiles to **JavaScript** (EcmaScript 3+). **TypeScript** offers type annotations which provide optional, static type checking at compile time. Since it is a superset of **JavaScript**, all **JavaScript** is syntactically valid **TypeScrip.**

**--------------------------------------------------------X---------------------------------**

**What is this with respect to java script?**

JavaScript’s this keyword is the source of a lot of confusion for many developers every single day. Unlike a language with a rigid class model, it’s not always clear what this is going to refer to in your code, especially when dealing with call-back functions, whose call sites you have no control over.

this.setState({ loading: true });

fetch('/').then(function loaded() {

this.setState({ loading: false });

});

This code results in a TypeError because this.setState is not a function. This is because when the callback to the promise is called, the internal context of the function is changed and this references the wrong object. Let’s take a look at the ways in which we can prevent this from happening.

So solution to above problem is as follows-2 solution

var component = this;

component.setState({ loading: true });

fetch('/').then(function loaded() {

component.setState({ loading: false });

});

Or

this.setState({ loading: true });

fetch('/').then(function loaded() {

this.setState({ loading: false });

}.bind(this));

All functions in JavaScript have a [bind method](https://developer.mozilla.org/en/docs/Web/JavaScript/Reference/Global_objects/Function/bind), which allow you to specify the value for this. Once a function has been “bound” the context can’t be overridden, meaning that we have a guarantee that this will refer to the correct thing.

Or

### React Component Methods

React allows you to define arbitrary methods on your component classes and these methods are automatically bound with the correct context for this when you create your components with React.createClass. This allows you move your callback code out onto your component.

React.createClass({

componentWillMount: function() {

this.setState({ loading: true });

fetch('/').then(this.loaded);

},

loaded: function loaded() {

this.setState({ loading: false });

}

});

**-------------------------------------X---------------------------------**

**What is binding in JavaScript?**

Ans-In **JavaScript** function **binding** is happens using **Bind**() method. by this method, we can **bind** an object to a common function, so that the function gives different result when its need. ... In other words, **bind**() method allows us to easily set which object will be bound by the this keyword when a function or method is invoked

**-----------------------------------------X-------------------------------------------------------------**

What is the difference between call,bind and appy in javascript?

Ans-The **call**, **bind** and **apply** methods can be used to set the this keyword independent of how a function is called. The **bind** method creates a copy of the function and sets the this keyword, while the **call and apply** methods sets the this keyword and **calls** the function

const Employee= {

age: 42,

getAge: function() {

return this.age;

}

};

const unboundGetAge = Employee.getAge;

console.log(unboundGetAge()); // The function gets invoked at the global scope

// expected output: undefined

const boundGetAge = unboundGetAge.bind(Employee);

console.log(boundGetAge());

// expected output: 42

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*/

/\* let user = {

firstName: "John",

sayHi() {

console.log(`Hello, ${this.firstName}!`);

}

};\*/

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*solution is wrapping\*\*\*\*\*\*\* \*/

let user = {

firstName: "John",

sayHi() {

console.log(`Hello, ${this.firstName}!`);

}

};

setTimeout(function() {

user.sayHi(); // Hello, John!

}, 1000);

setTimeout(user.sayHi, 1000); // Hello, undefined!

console.log("\*\*\*\*\*\*\*\*\*\*\*\*\*\*other solution\* using bind\*\*\*\*\*\*\*\*\* ");

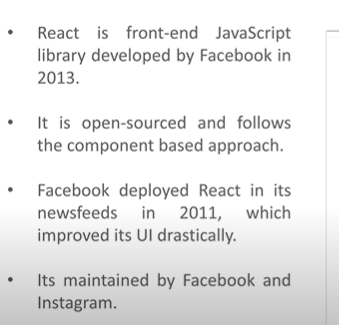
var callHi=user.sayHi;

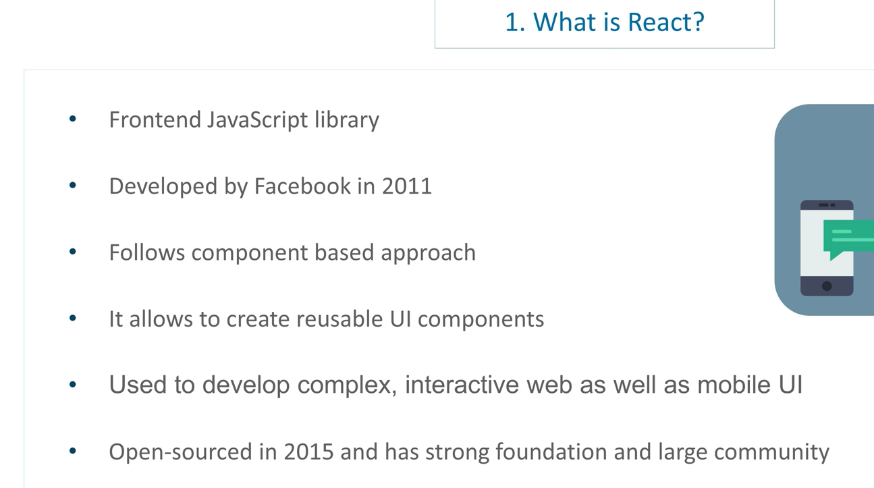
//var bindCallHi= callHi.bind(user);

setTimeout(callHi.bind(user), 2000);

**--------------------------------------X-----------------------------------**

**What si React?**



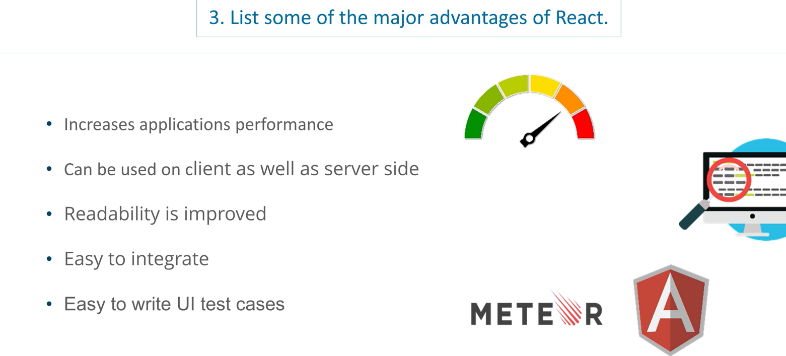


**--------------------------------------------X-----------------------------------------**

**What are features of react?**



* Use Virtual Dom
* React does server side rendering[**Server**-**side rendering** (SSR) is a popular technique for **rendering** a **client**-**side** single page application (SPA) on the **server** and then sending a fully **rendered** page to the **client**.
* Follow uni directional data flow



**------------------------------------X----------------------------------------**

**What are limitations of react?**

* It is not full scale framework for MVS. It is just a view.
* Library is very Large
* For new programmer it is difficult to understand
* It uses inline templet and JSX

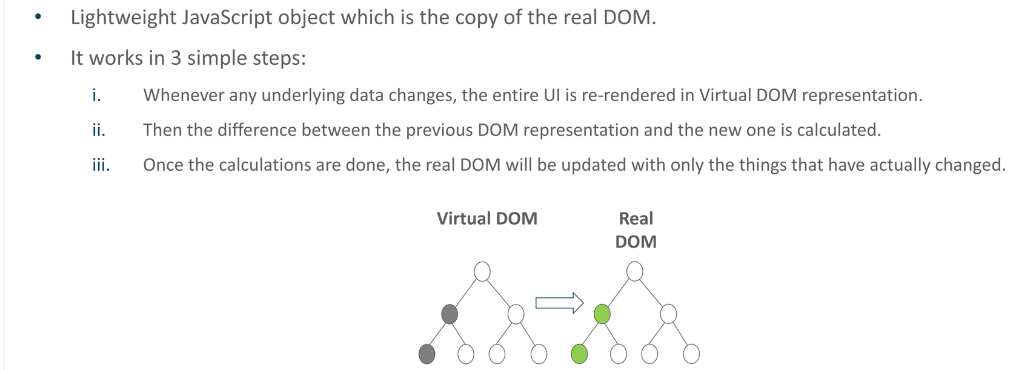
**----------------------------------X---------------------------------------------**

**What is JSX?**

It is java script xml. It uses java script expression with html templet syntax. It makes html easy to understand. It is robust and boot the performance. JSX expression must have only one outermost element.

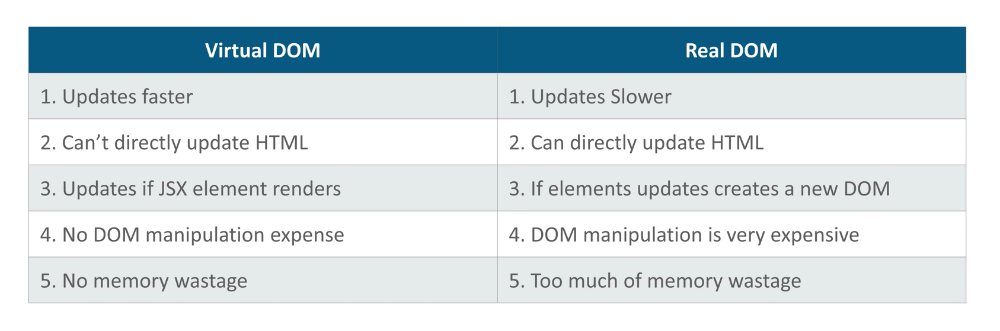
**------------------------------------------X-------------------------------------**

**What is Virtual DOM?**



-------------------------X-----------------------------------------------

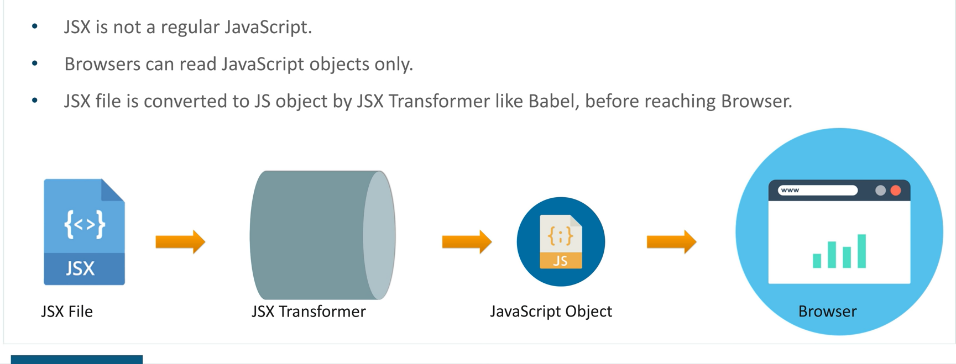
What is difference between real DOM and virtual DOM?



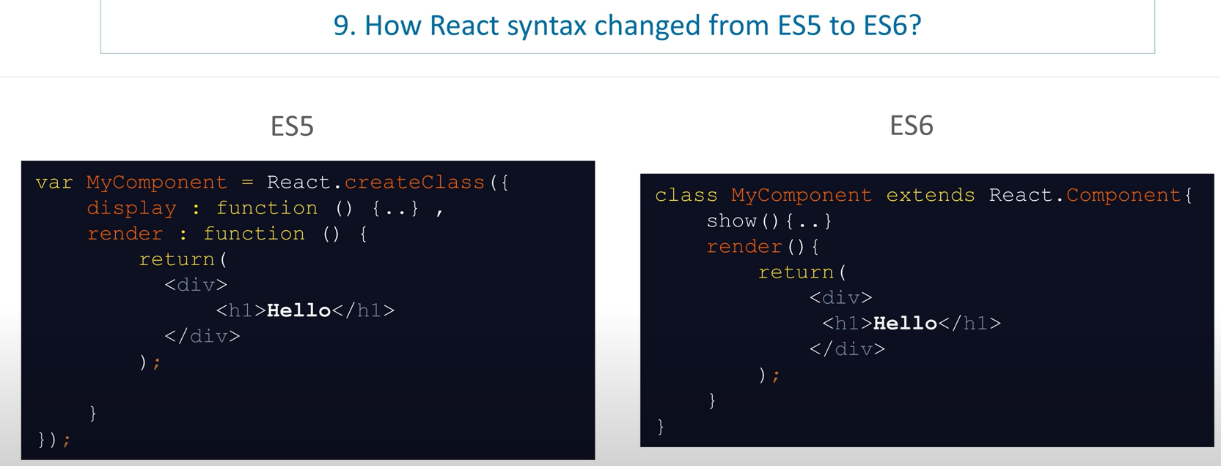
**---------------------------------------------X-------------------------------------------**

**Why browser can’t read JSX?**

Ans-

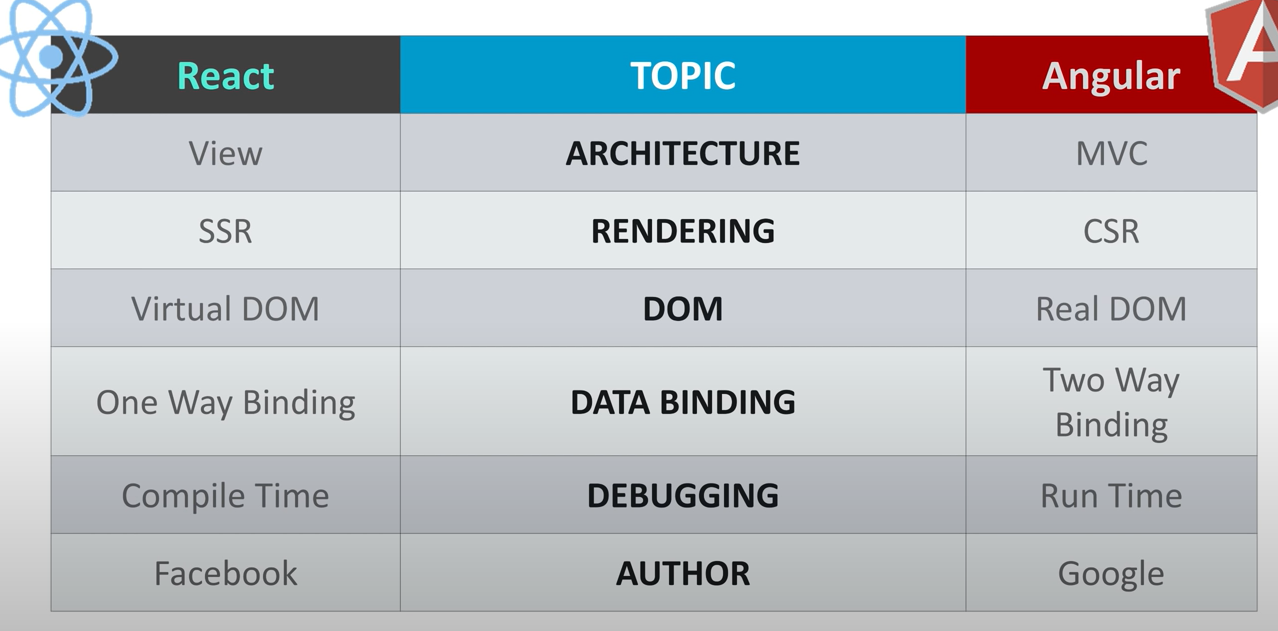


**-----------------------------------------\*------------------------------------------------------**



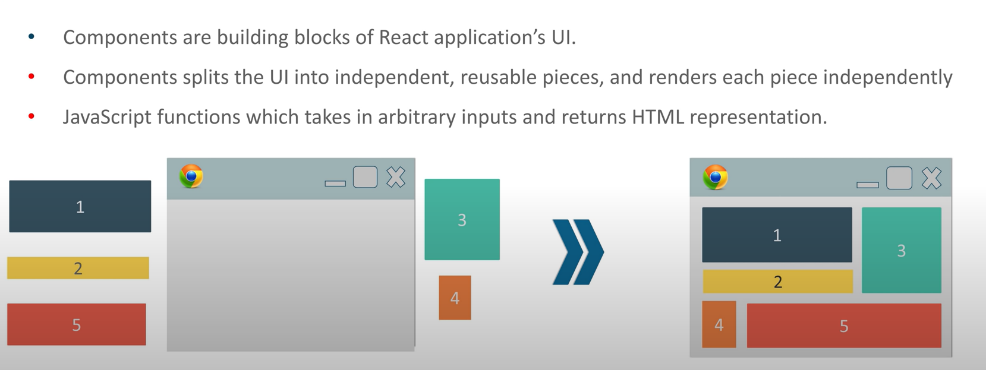
**--------------------------------------X----------------------------------------------------**

What is difference between React and angular? [SSR-CSR-server side and client side rendering]



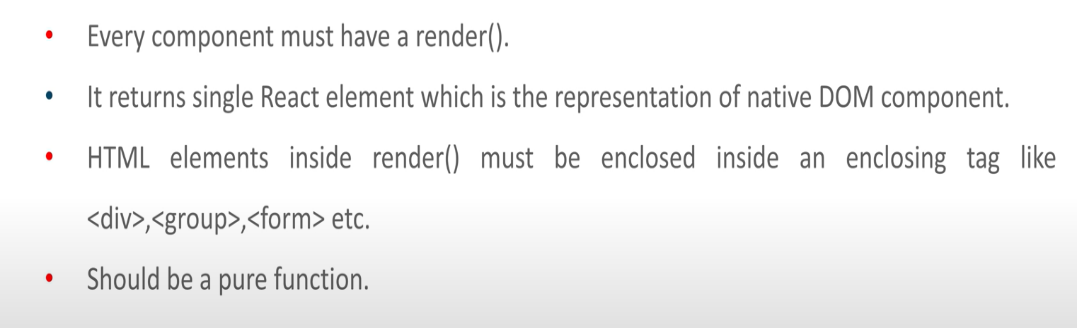
\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*X\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Why do we say that in react everything is a component?



**------------------------------------------------------X-----------------------------------------------**

**What is render function in react?**



Render function is pure which do not change the state or logic /data.

**-----------------------------------X--------------------------------------------------------**

**What are pure functions?**

Ans-**Pure functions** take an input value (a parameter or argument) and depending on that input, produce an output value, that's all. They do one thing only, but they do it well. It should be that whenever you give a **pure function** the same input it will return the same output every single time.

**---------------------------------------X----------------------------------------------------------------**

What is props in react?



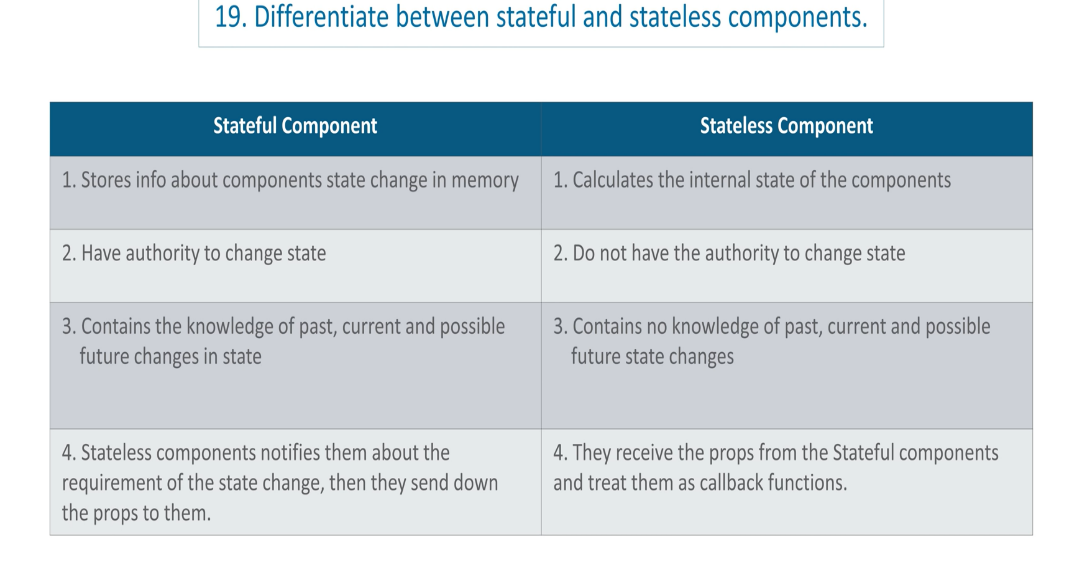
**-------------------------------------------------------------X------------------------------------------------**

**What is state in react?**



**-----------------------------------X--------------------------------------------------------------------------------**

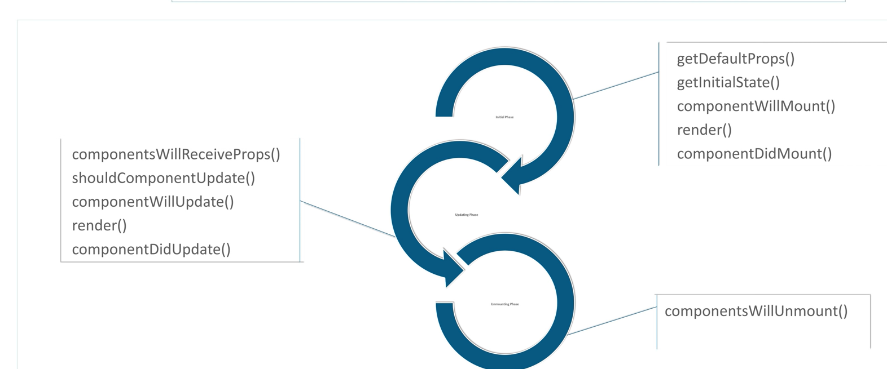
**What is the difference between Statefull and stateless component?**

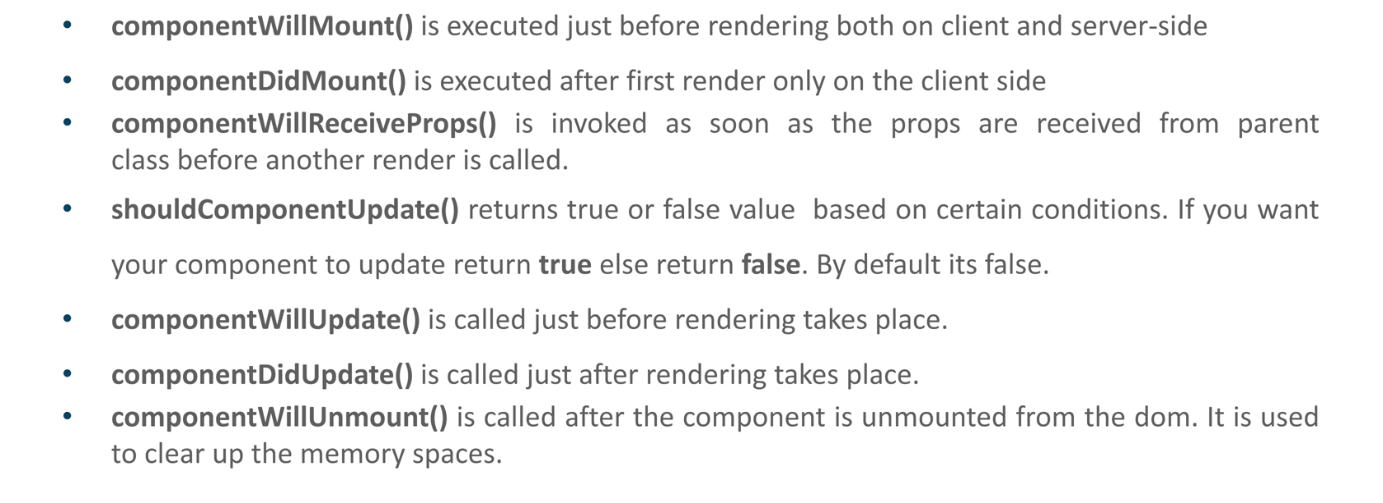


**---------------------------------X----------------------------------------------------------**

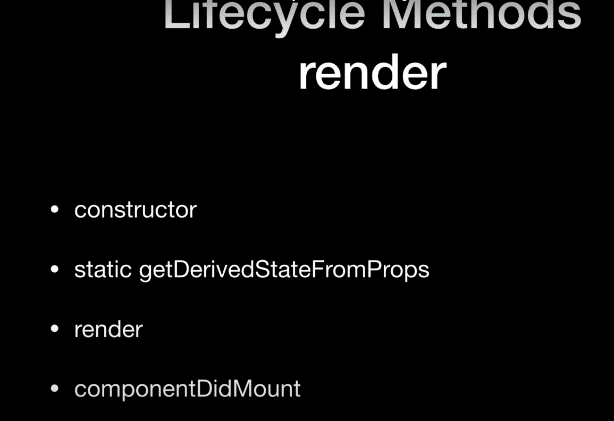
Following are React 16.x life cycle method-------------------------------------------

Phase- 4 phases- initial phase/update/Mounting and /Unmounting



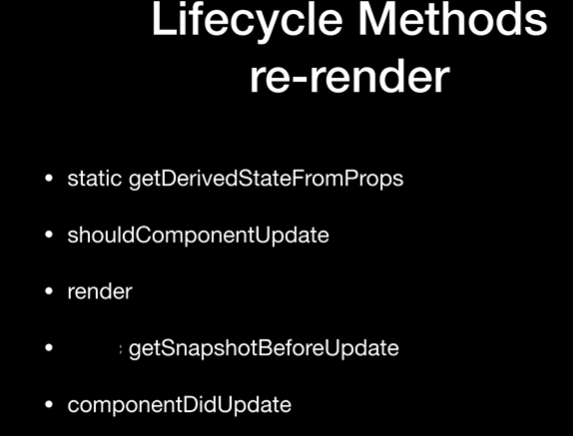


React Lifecycle –Initial Render has following method

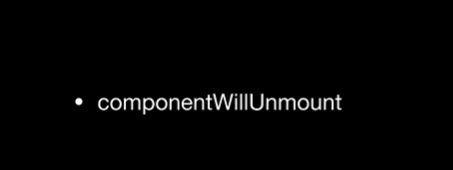


* Constructor set the initial state.
* getDerivedstateProps is replacement of old [**componenetWillReceivedProps** ] it will get state if props changes so it will set state accordingly and it is static method.so u cannot use this here
* render-[Manadatory method where u caanot set state.
* componentDidMount- u can use third party call here. Which u want to get ready.ex third party chat componenet u can use here.

Rerender------------This phase will run many time whenever there is change in the ui



* shouldComponentUpdate-decide if u really want this component to be updated or not. If somebody set State if previous state and new state is same no need to rerender because render is same as previous. So if u return true render. If u say return false it will not get render.
* getSnapshotBeforeUpdate- This is pre commit phase. Mounting happens after this method. React recently introduced lazy loading. If delay between if u render component and next phase and if user scrool. So u can do something here.[This is replacement of componentWillUpdate)

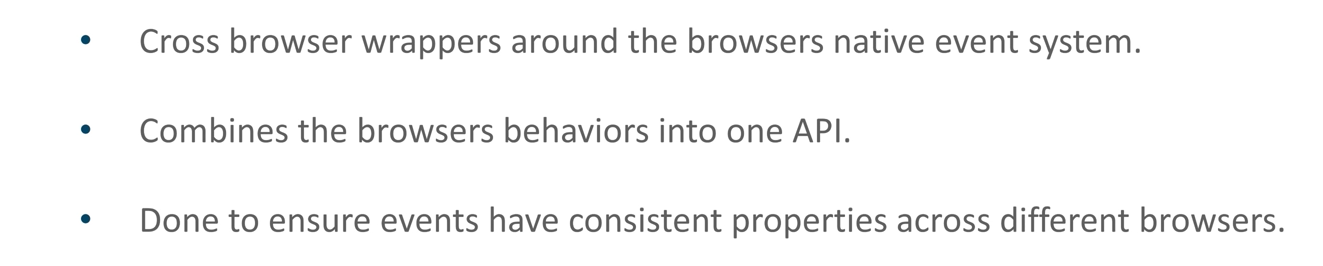


\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*X\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

In latest version of react what are life cycle method in react hoos?

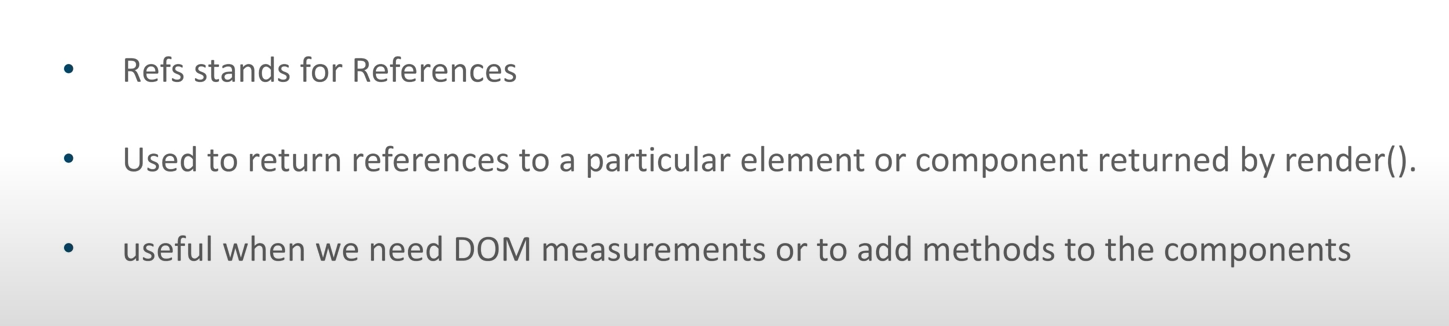
**--------------------------------------X--------------------------------**

**What is synthetic events in React?**



**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*X\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**What is refs in React?**



Using ref we can access dom node directly in react

**----------------------------------------------X------------------------------------------**

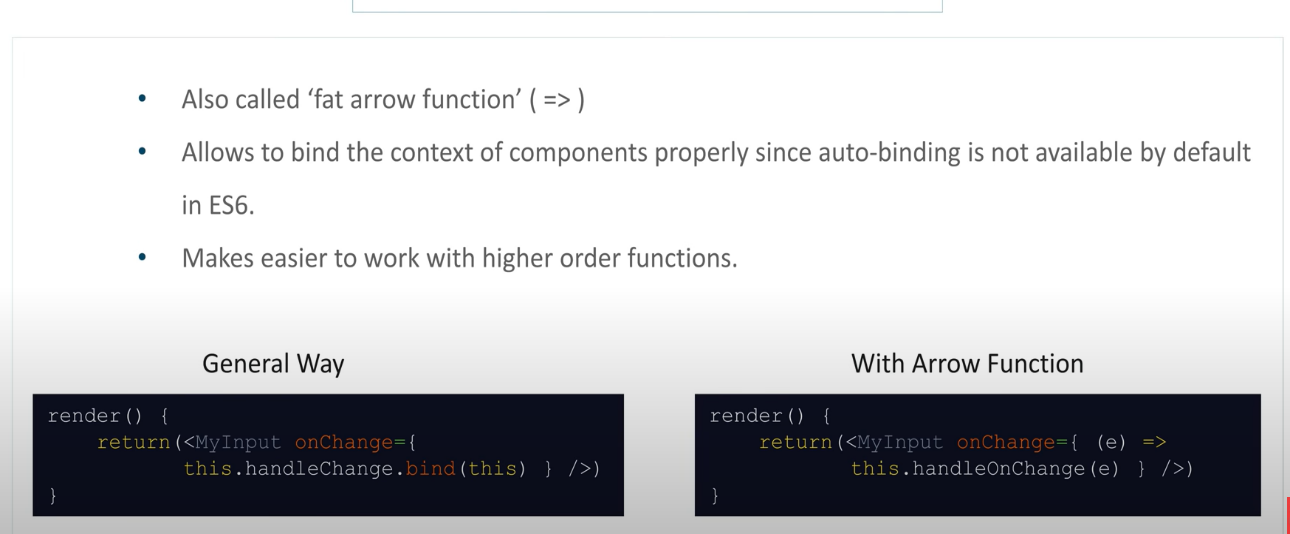
**When do we use Refs?**

Ans:-



-------------------------------------X----------------------------------------------------

**What is arrow function? And Why do we use arrow function?**





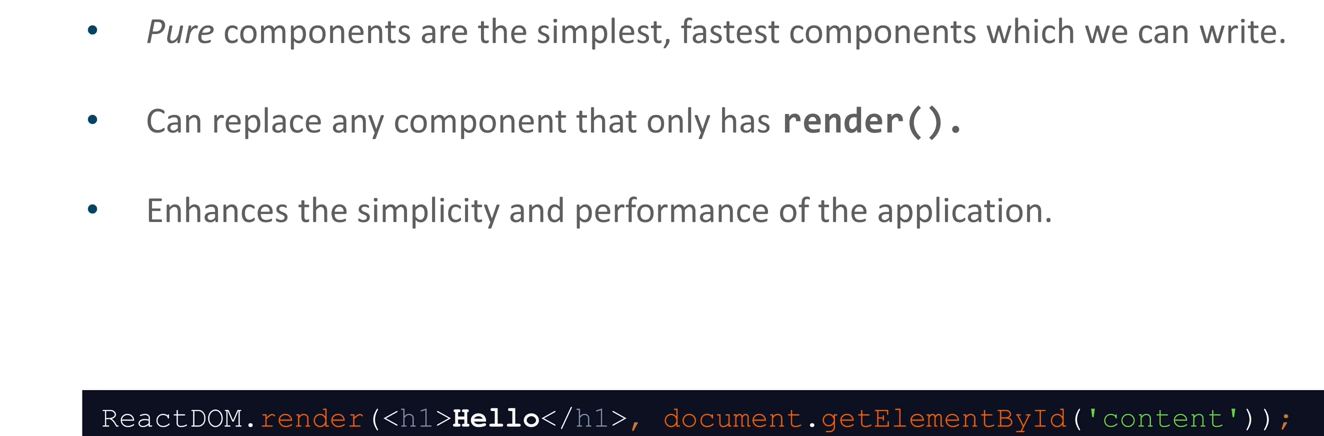
* In above code login Handler is a property of App Class .it is not function
* Generally, in react component class state belonged to class as property. Which we define

In constructor using this. State or state.

* If you make above login hander as function and if you have function with function where u set state or use this it will not belong to class.It will belong to function.
* So almost all clieckevent handler are class properties but not class methods.
* Rro function do not have its own this. It automaticcaly take this from lexicle scope that is immegiate parent
* Another way is without arrow function using bind method in class with the constructor.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*X\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**What is Pure Component?**



import React from ‘react’;

export default class Test extends React.PureComponent{

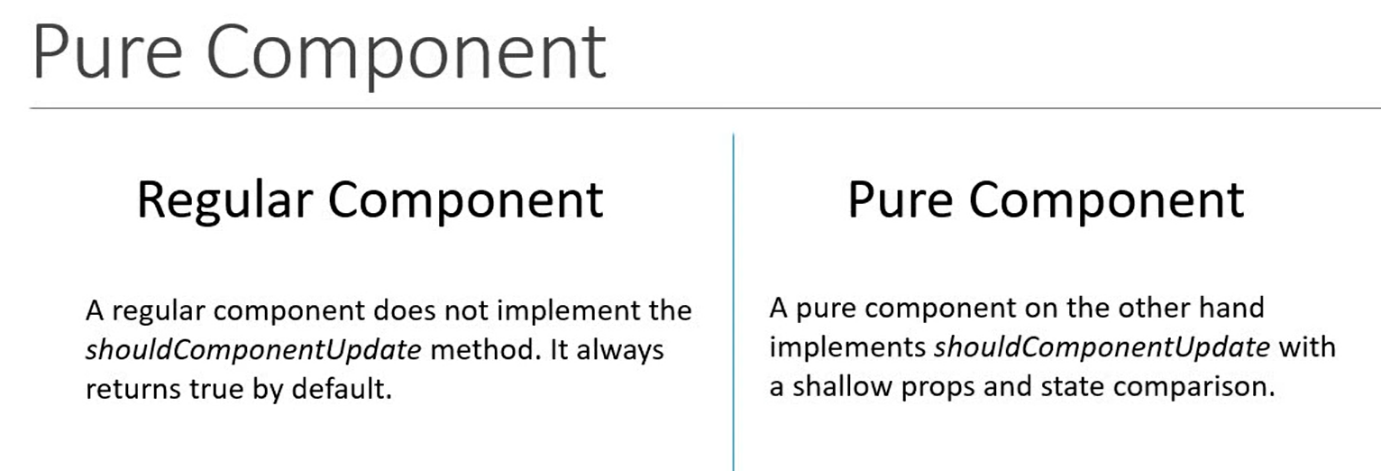
   render(){

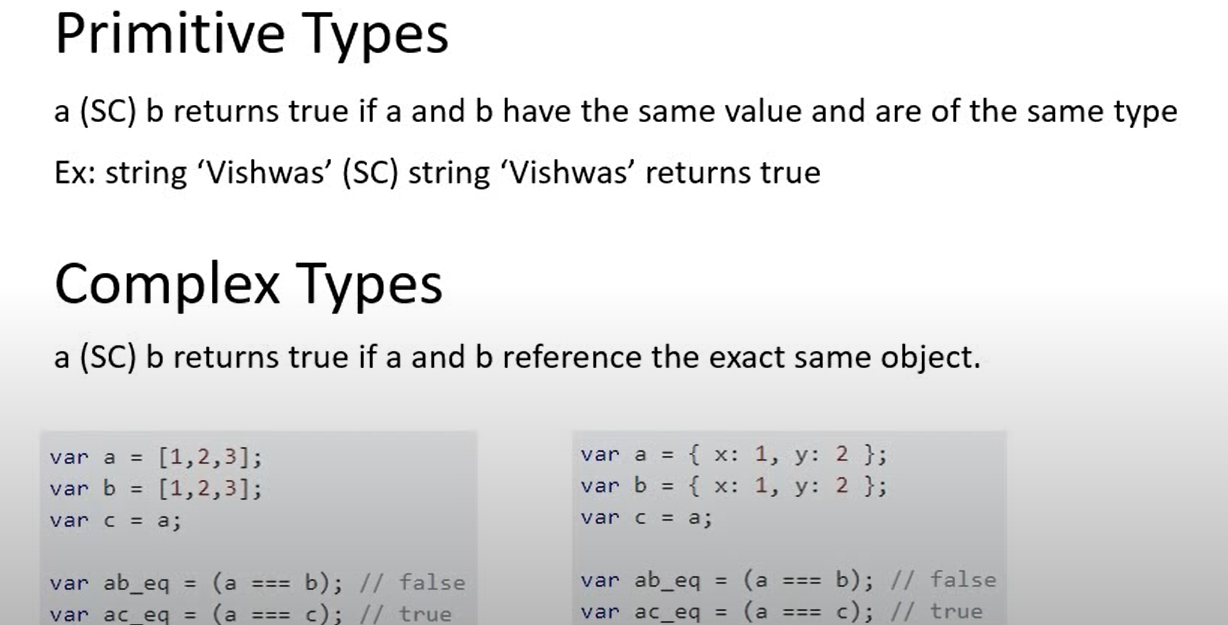
      return <h1>Welcome to GeeksforGeeks</h1>;

   }

}

* **ReactJS** has provided us a **Pure Component**. If we extend a class with **Pure Component**, there is no need for**shouldComponentUpdate()**Lifecycle Method.
* **ReactJS Pure Component** Class compares current state and props with new props and states to decide whether the React component should re-render itself or Not.
* In simple words, If the previous value of state or props and the new value of state or props is the same, the component will not re-render itself.
* Since **Pure Components** restricts the re-rendering when there is no use of re-rendering of the component.
* Pure Components are Class Components which extends **React.PureComponent**.
* Extending React Class Components with **Pure Components** ensures the higher performance of the Component and ultimately makes your application faster, While in the case of Regular Component, it will always re-render either value of State and Props changes or not.
* While using Pure Components, Things to be noted are that, In these components, the Value of State and Props are **Shallow Compared** (Shallow Comparison) and It also takes care of “**shouldComponentUpdate**” Lifecycle method **implicitly**.
* So there is a possibility that if these State and Props Objects contain nested data structure then**Pure Component’**s implemented **shouldComponentUpdate** will return false and will not update the wholesubtree of**Children** of this**Class Component.**
* So in **Pure Component**,the nested data structure doesn’t work properly.

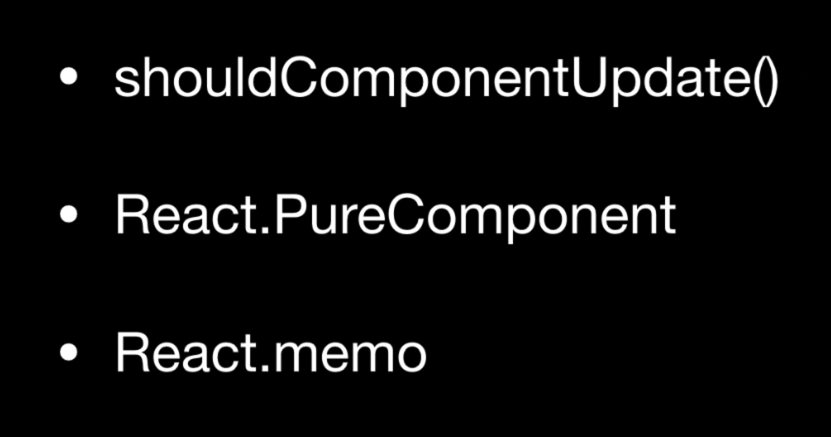


****

**----------------------------------------------------X-------------------------------------------------**

**How to prevent the component from retendering?**

Ans:-Using 3 different was as follows.

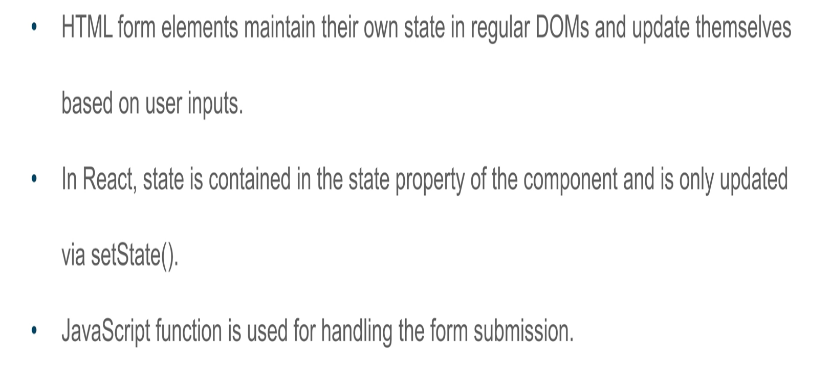


In Purecomponent we don’t have to use shouldComponentUpdate() .

React Memo should be used with fuctional component.

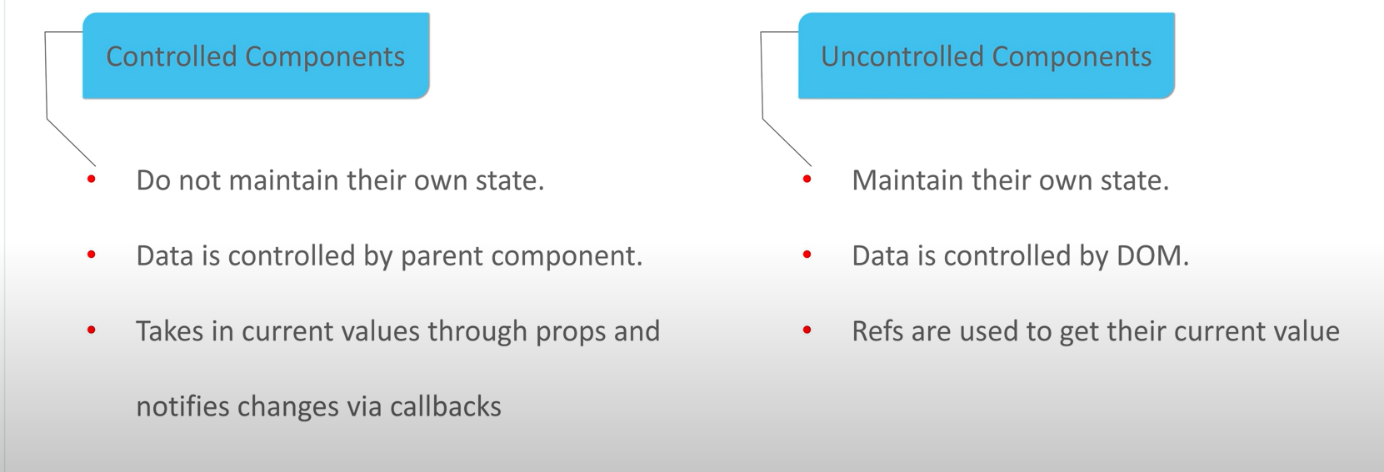
**------------------------X---------------------------------------------------------**

**How Forms are created in react?**



**-----------------------------------------------X--------------------------------**

**What is the difference between controlled and uncontrolled component?**



In conventional Html form element we need to write java script code to get form data.

## Controlled Components

In HTML, form elements such as <input>, <textarea>, and <select> typically maintain their own state and update it based on user input. In React, mutable state is typically kept in the state property of components, and only updated with [setState()](https://reactjs.org/docs/react-component.html" \l "setstate).

We can combine the two by making the React state be the “single source of truth”. Then the React component that renders a form also controls what happens in that form on subsequent user input. An input form element whose value is controlled by React in this way is called a “controlled component”.

With a controlled component, the input’s value is always driven by the React state. While this means you have to type a bit more code, you can now pass the value to other UI elements too, or reset it from other event handlers. class NameForm extends React.Component {

constructor(props) {

super(props);

this.state = {value: ''};

this.handleChange = this.handleChange.bind(this);

this.handleSubmit = this.handleSubmit.bind(this);

}

handleChange(event) {

this.setState({value: event.target.value});

}

handleSubmit(event) {

alert('A name was submitted: ' + this.state.value);

event.preventDefault();

}

render() {

return (

<form onSubmit={this.handleSubmit}>

<label>

Name:

<input type="text" value={this.state.value} onChange={this.handleChange} />

</label>

<input type="submit" value="Submit" />

</form>

);

}

}

**------------------UnControlled Component------------------**

UncrolledComponent Demo

In most cases, we recommend using [controlled components](https://reactjs.org/docs/forms.html#controlled-components) to implement forms. In a controlled component, form data is handled by a React component. The alternative is uncontrolled components, where form data is handled by the DOM itself.

To write an uncontrolled component, instead of writing an event handler for every state update, you can [use a ref](https://reactjs.org/docs/refs-and-the-dom.html) to get form values from the DOM.

class NameForm extends React.Component {

constructor(props) {

super(props);

this.handleSubmit = this.handleSubmit.bind(this);

this.input = React.createRef();

}

handleSubmit(event) {

alert('A name was submitted: ' + this.input.current.value);

event.preventDefault();

}

render() {

return (

<form onSubmit={this.handleSubmit}>

<label>

Name:

<input type="text" ref={this.input} />

</label>

<input type="submit" value="Submit" />

</form>

);

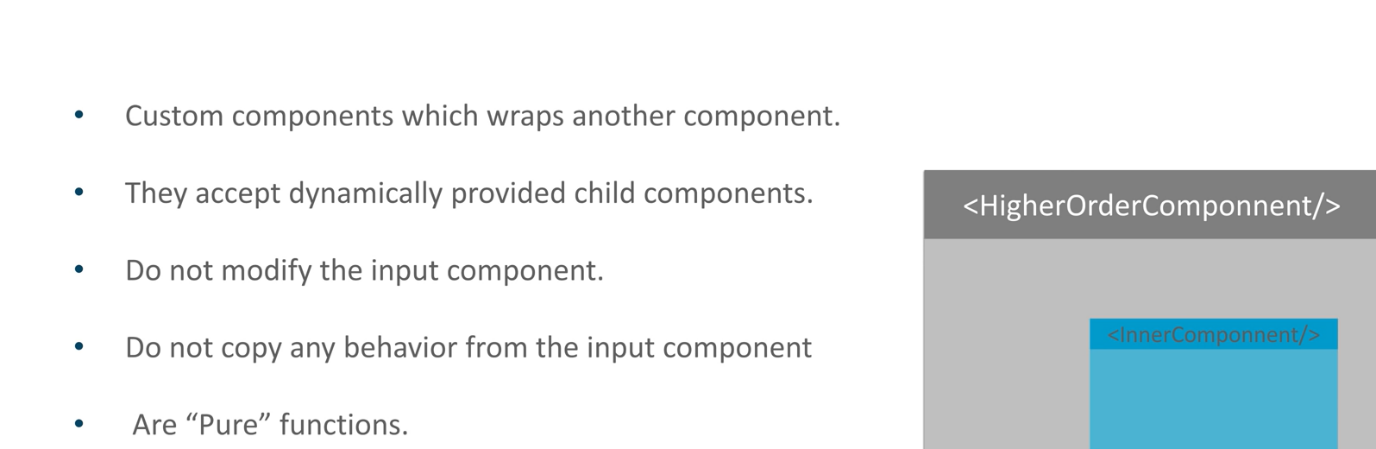
}

}

**------------------------X-------------------------------------**

**What is HOC?**

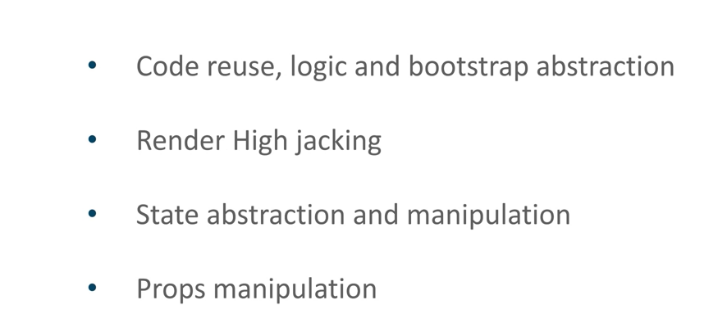
Ans:-



-----------------------------------------X--------------------------------------

**What can we do with HOC?**

Ans:-



**-----------------------------X-----------------------------------------------**

**3)what is the difference between npm install –save and npm install –save-dev?**

Ans-NPM (Node Project Manager) is a package manager used by JavaScript runtime environment Node.js. It has two very frequently used commands to downloaded different dependencies, npm install --save [package-name] and npm install --save-dev [package-name]. Both commands will lead to download and installation of packages from NPM servers but they have a bit different ways.

**npm install [package-name] –save**: When –save is used without -dev, it signifies that the package is core dependency. A core dependency is any package without which the application cannot perform its intended work. In package.json file under the dependencies section contains the list of core dependencies. The npm install will also lead to a similar result. When someone installs your package they will also install all the packages listed in the dependencies section of package.json. Example: express, body-parser.

**npm install [package-name] –save-dev**: When –save-dev is used with npm install, it signifies that the package is a development dependency. A development dependency is any package that absence will not affect the work of the application. In package.json file under the devDependencies section contains the list of all development dependencies. When someone installs your package they will not install any development dependencies but if they clone the repository, then they will install all the development dependencies too. Example: nodemon

**---------------------------------------------------X--------------------------------------**

**What is browserlist in package.js?**

### Ans-browserslist

Is used to tell which browsers (and their versions) you want to support. It's referenced by Babel, Autoprefixer, and other tools, to only add the polyfills and fallbacks needed to the browsers you target.

Example:

"browserslist": [

"> 1%",

"last 2 versions",

"not ie <= 8"

]

This configuration means you want to support the last 2 major versions of all browsers with at least 1% of usage (from the [CanIUse.com](https://caniuse.com/) stats), except IE8 and lowe

What is eslistconfig in package.json?

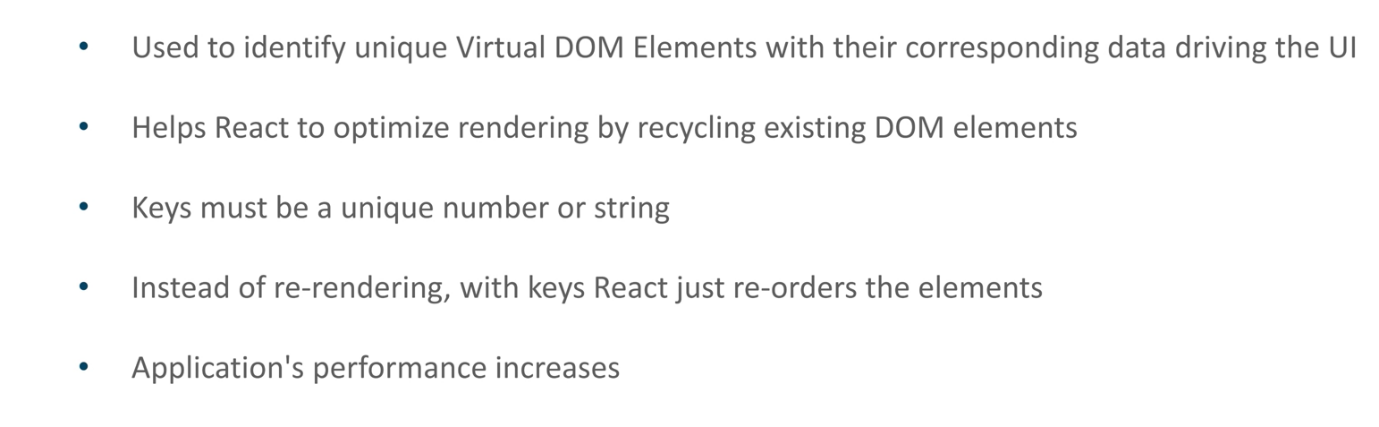
### Command-specific properties

The package.json file can also host command-specific configuration, for example for Babel, ESLint, and more.

Each has a specific property, like eslintConfig, babel and others. Those are command-specific, and you can find how to use those in the respective command/project documentation

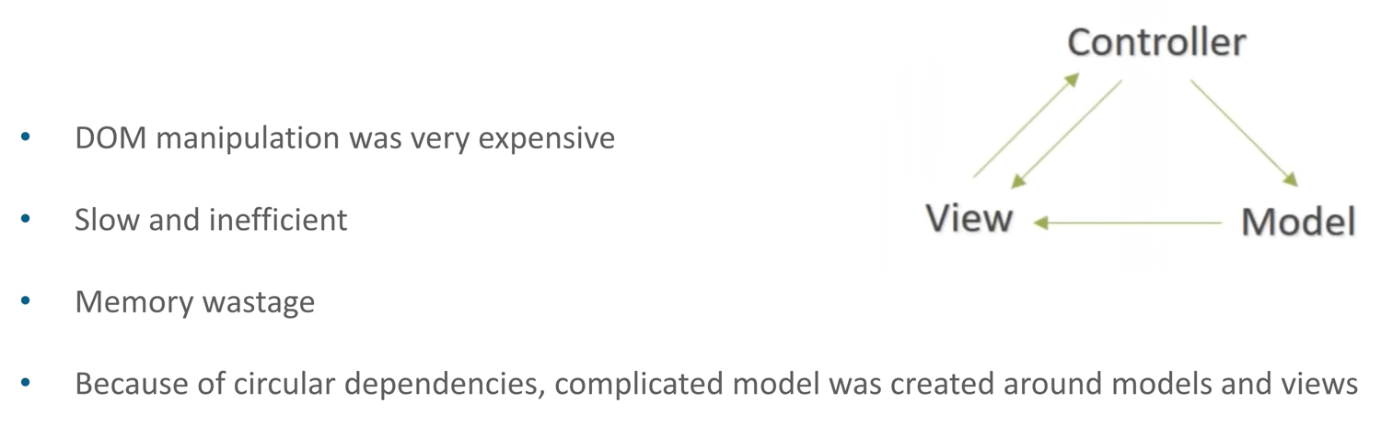
**-------------------------X-----------------------------**

**Why do we use keys in react ?**



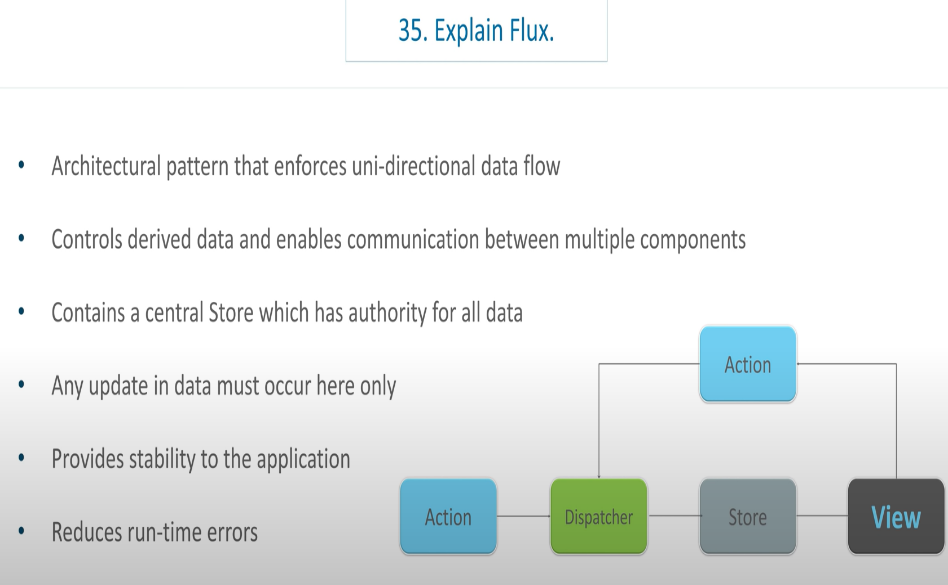
**-------------------------------------------------------X-------------------------------------------**

**What is draw back of MVC?**



**------------------------------------------------------X---------------------------------------------------**

**What is Flux?**



**----------------------------------------X-------------------------------------------------------**

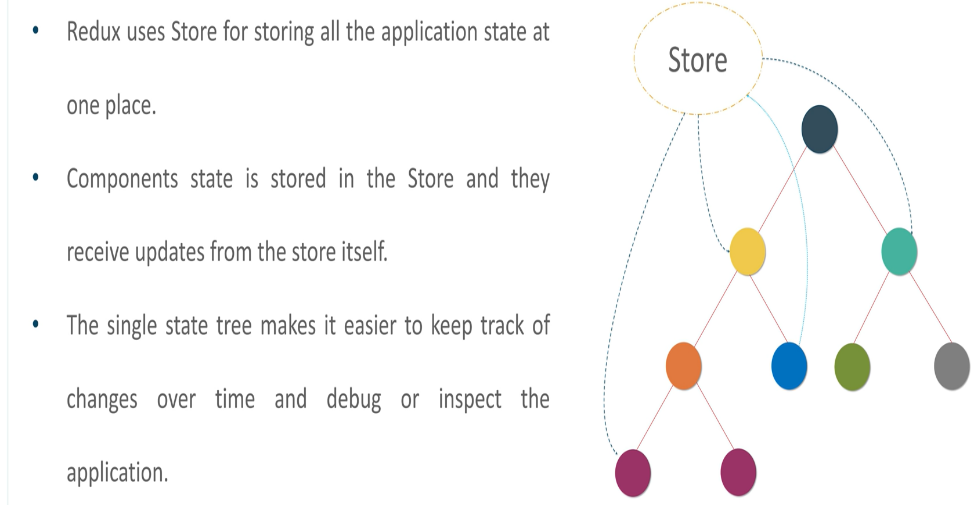
**What is Redux?**

Ans-Redux is hottest third party library used for state management. Application developed with redux are easy to test and can be executed in all environment.

**---------------------------------------X-----------------------------------------**

**What is meant by single source of truth in redux?**

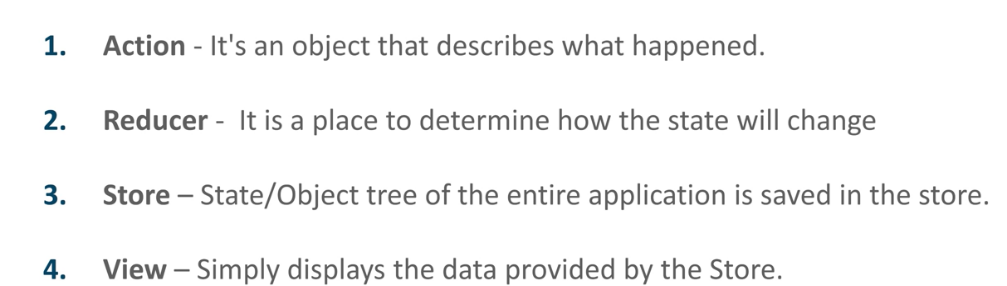
Ans-



**--------------------------------------X--------------------------------------**

**What are 4 different components of Redux?**

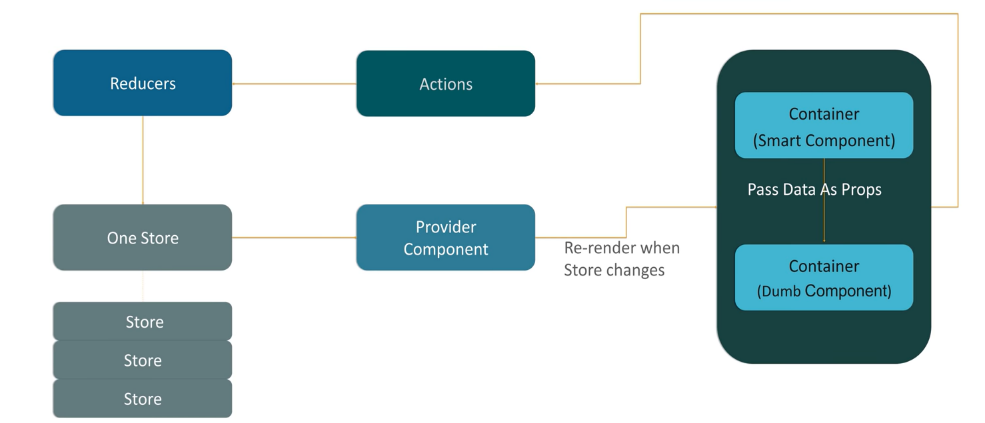
Ans:-



**-------------------------------X----------------------------------------**

**How Data flows in the redux?**

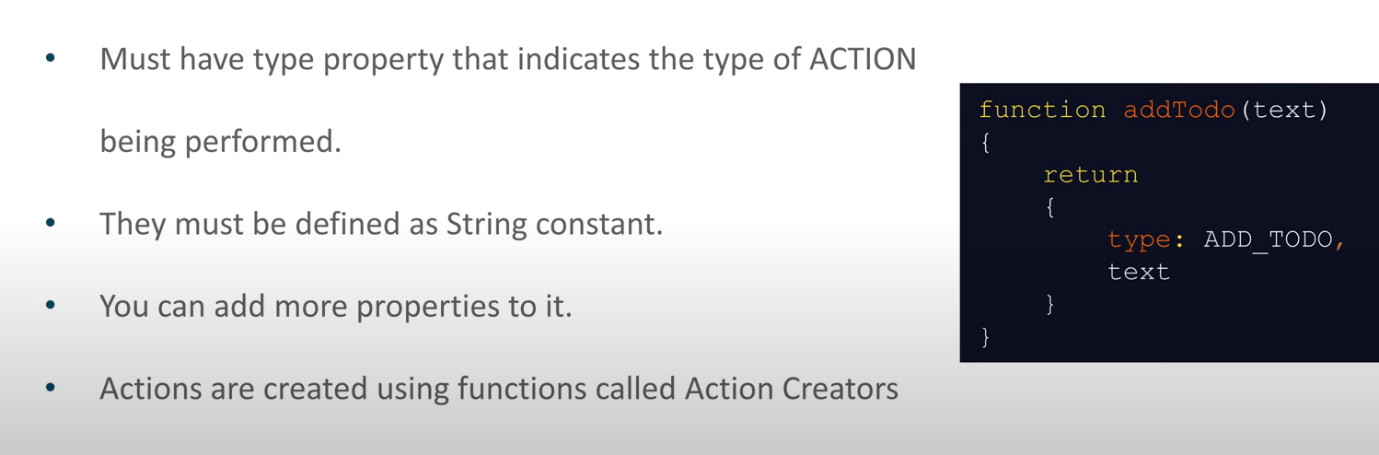
**Ans-**

-

**-----------------------------------------X--------------------------------------------**

**What is action in redux?**

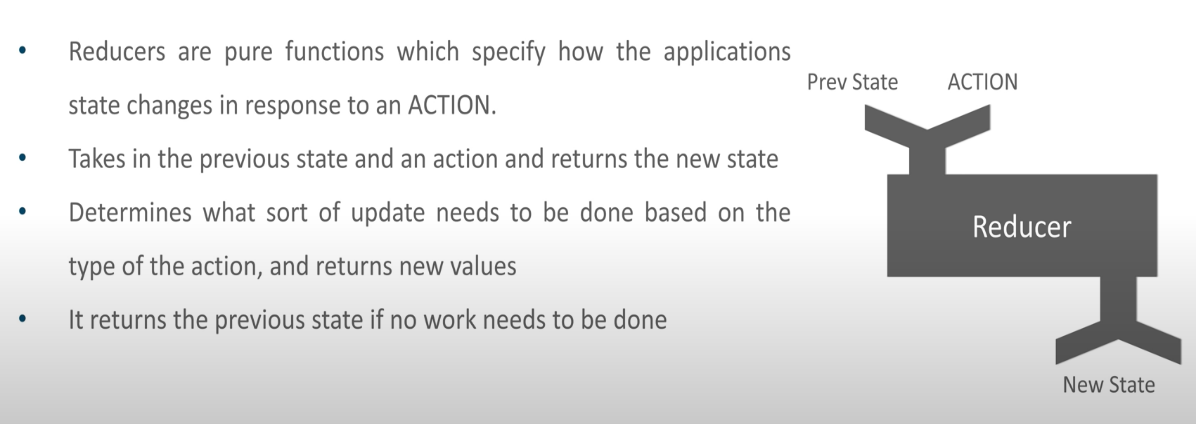
Ans-



**-------------------------------X-----------------------------**

**What is reducer?**

Ans-



**------------------------------------X----------------------------------------------------**

**What is store in redux?**

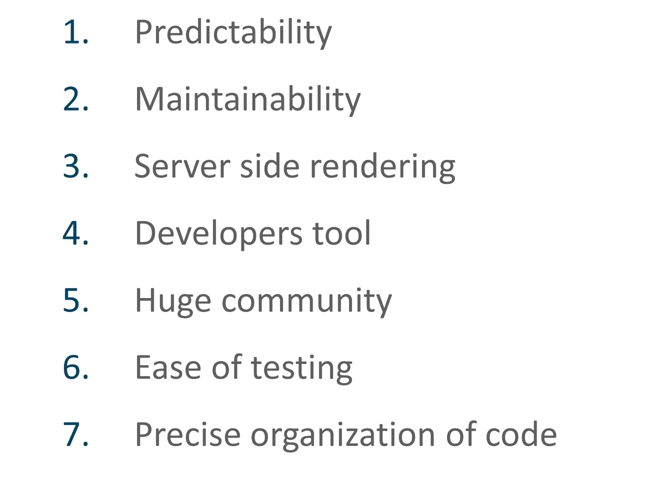
Ans-

-

**------------------------------------X--------------------------------------------**

**What are advantages of redux?**

Ans-



--------------------------------X-------------------------------------

Why switch keyword is used in react routining V4 ?

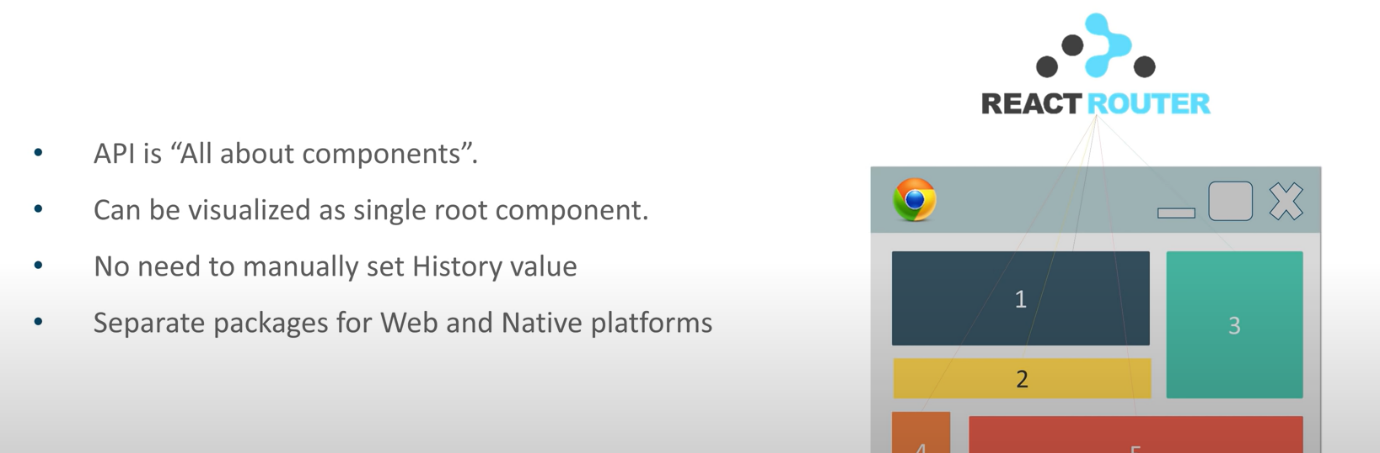
Ans-



**----------------------------------X----------------------------------------**

**What are advantages of Router?**

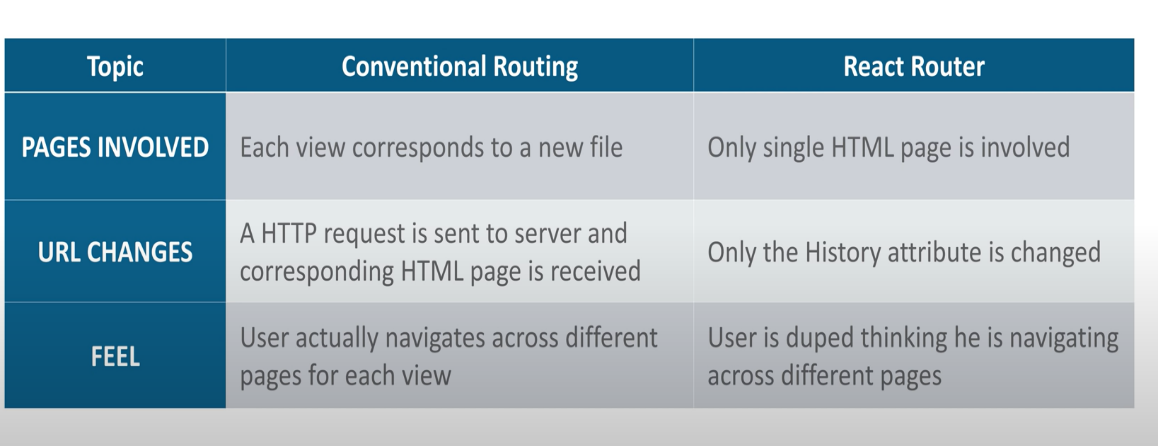
Ans-



-----------------------------------X----------------------------------------

**How react routing is different than conventional routing?**

Ans-



**-------------------------------------------X-----------------------------**

**What is difference between ES6 and Typescript?**

Ans:- **TypeScript** is a free and open-source pure object-oriented programming language. It is developed and maintained by Microsoft.

**ES6** is a version of ECMAScript (ES), which is a scripting language specification standardized by ECMA international. **Typescript** is to eradicate the development errors.

**-----------------------------------X--------------------------------------------**

**What is asynch and await keywords in ES6?**

**Async**/**Await** is the extension of promises which we get as a support in the language.

**Async**: ... It makes sure that a promise is returned and if it is not returned then **javascript** automatically wraps it in a promise which is resolved with its value

he **await** expression causes **async** function execution to pause until a Promise **is** settled (that **is**, fulfilled or rejected), and to resume execution of the **async** function after fulfillment. ... If the Promise **is** rejected, the **await** expression throws the rejected value.

The advantage of an async function only becomes apparent when you combine it with the [await](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Operators/await) keyword. await only works inside async functions within regular JavaScript code, however it can be used on its own with [JavaScript modules.](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Guide/Modules)

await can be put in front of any async promise-based function to pause your code on that line until the promise fulfills, then return the resulting value.

**-------------------------Async-await demo--------------------------------------**

const fetch = require("node-fetch");

function hello1() { return "Hello1" };

console.log(hello1());//prints Hello

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*/

async function hello2() { return "Hello2" };

console.log(hello2());// This prints Promise { 'Hello2' }

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*/

/\* Invoking the function now returns a promise.

This is one of the traits of async functions —

their return values are guaranteed to be converted to promises\*/

let hello3 = async function() { return "Hello3" };

console.log(hello3());

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

let hello4 = async () => { return {id:"1",name:"aaa"} };

console.log(hello4());

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

/\*To actually consume the value returned when the

promise fulfills, since it is returning a promise,

we could use a .then() block: \*/

hello3().then((value) => console.log(value))

hello4().then((response) => console.log(response.name))

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*await\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*/

/\* The advantage of an async function only becomes

apparent when you combine it with the await keyword.

await only works inside async functions within regular

JavaScript code, however it can be used on its own with JavaScript modules.

await can be put in front of any async promise-based function

to pause your code on that line until the promise fulfills,

then return the resulting value.

You can use await when calling any function that

returns a Promise, including web API functions. \*/

/\* async function hello5() {

return greeting = await Promise.reject("Hello5");

};

hello5().

then(console.log("heelo5 fullfilled"))

.catch(console.log("heelo5 rejected"));\*/

/\*\*\*\*\*\*\*\*\*npm install node-fetch --save\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*/

//fetch('ftp://C:/VaishaliBatchDemo/AccountTrainingES6/ES6Demo/Ball.GIF')

/\*fetch("http://localhost:3000/emps")

.then(response => {

if (!response.ok) {

throw new Error(`HTTP error! status: ${response.status}`);

}

//return "response.blob()";

return response;

})

.then(myBlob => {

//let objectURL = URL.createObjectURL(myBlob);

// let image = document.createElement('img');

// image.src = objectURL;

//document.body.appendChild(image);

console.log("Inage shown"+myBlob)

})

.catch(e => {

console.log('There has been a problem with your fetch operation: ' + e.message);

});

\*/

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*/

async function myFetch()

{

let response = await fetch('http://localhost:3000/emps');

if (!response.ok) {

throw new Error(`HTTP error! status: ${response.status}`);

}

let myBlob = await response;

console.log("vaish..."+myBlob);

}

myFetch()

.catch(e => {

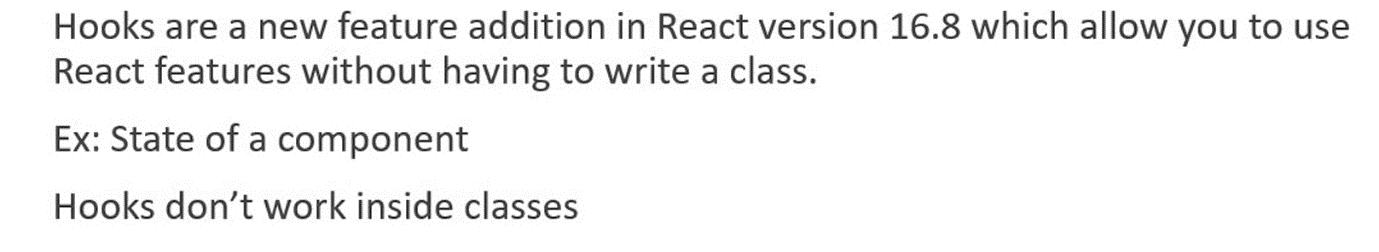
console.log('There has been a problem with your fetch operation: ' + e.message);

});

---------------------------------------\*---------------------------------------------

What is React hooks?

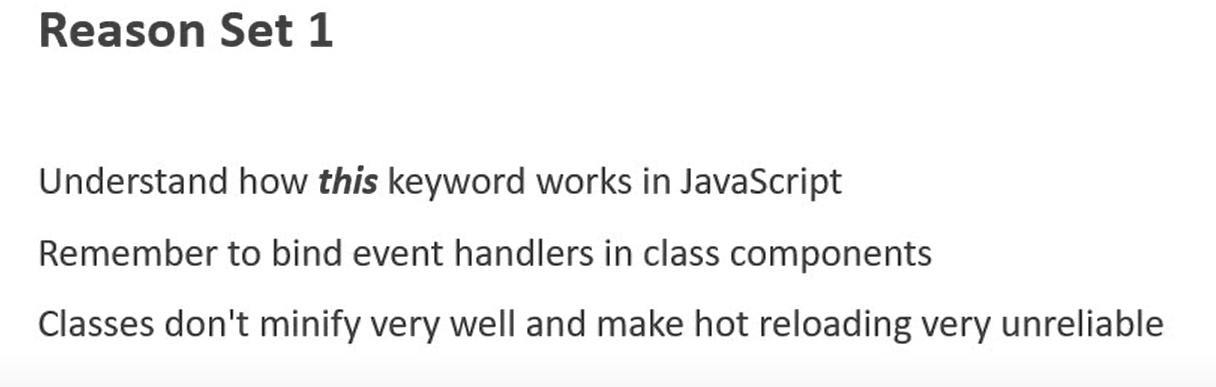
Ans-

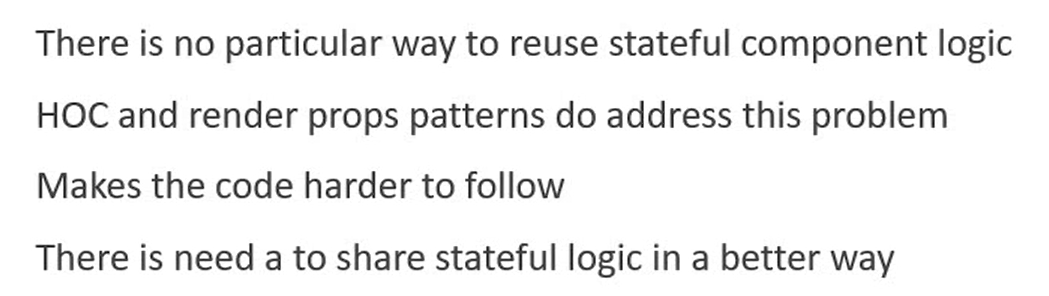


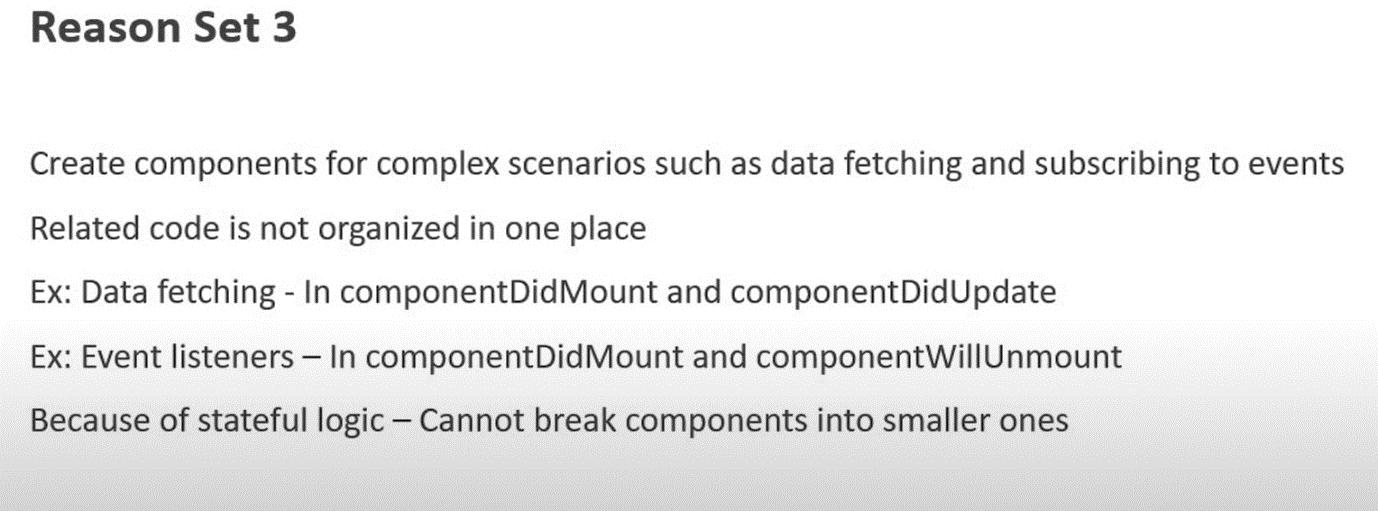
**-----------------------------------------X--------------------------------------------**

Why do we use hooks?

Ans-







--------------------------------------------------X-------------------------------------------------------------------------

**What are different Hooks in React**?

Ans- useState/useEffect

------------------------------------X-----------------------------------------

**What is difference between setState() in the class componentand useState**

**From functional component?**

Ans- setState() will merge the state but useState will not automatically meagre the state.U have to do it mannualy using spread operator

**---------------------------------useState Hook Demo--------------------------------**

import React, {useState} from 'react'

function HookCounter() {

//---------------------------------------------

/\*const [count, setCount] = useState(0)

return (

<div>

<button onClick={() =>

setCount(count + 1)}>Count {count}</button>

</div>

)\*/

//------------------2---------------------------

/\* const initialCount = 0

const [count, setCount] = useState(initialCount)

const incrementFive = () =>

{

for (let i = 0; i < 5; i++) {

setCount(prevCount => prevCount + 1)

}

}

return (

<>

Count: {count}

<button onClick={() => setCount(initialCount)}>Reset</button>

<button onClick={() => setCount(prevCount => prevCount + 1)}>

Increment

</button>

<button onClick={() => setCount(prevCount => prevCount - 1)}>

Decrement

</button>

<button onClick={incrementFive}>Increment 5</button>

</>

)\*/

//----------------------------3-------------------

/\*

const [name, setName] = useState({ firstName: '', lastName: '' })

return (

<form>

<input

type="text"

value={name.firstName}

onChange={e => setName({ ...name, firstName: e.target.value })}

/>

<input

type="text"

value={name.lastName}

onChange={e => setName({ ...name, lastName: e.target.value })}

/>

<h2>Your first name is - {name.firstName}</h2>

<h2>Your last name is - {name.lastName}</h2>

<h2>{JSON.stringify(name)}</h2>

</form>

)

\*/

//------------------------4----------------

const [items, setItems] = useState([])

const addItem = () => {

setItems([

...items,

{

id: items.length,

value: Math.floor(Math.random() \* 10) + 1

}

])

}

return (

<div>

<button onClick={addItem}>Add a number</button>

<ul>

{items.map(item => (

<li key={item.id}>{item.value}</li>

))}

</ul>

</div>

)

}

export default HookCounter

--------------------------------------------X--------------------------------------

**What is useEffect () and when it is executed?**

**Ans-**

* It is executed after every render of the component. And after each update.
* But in some cases it might create a performance problem.
* We can pass second parameter to the use Effect Function, which specify that first function is only executed when there is change in the props or state.
* So second parameter is either props or state which has to be watch for change.
* We can even run useEffect once. On initial render by passing second parameter as [].
* Whatever you return in useEffect that is consider as clean up like componentDidUnmount.

**-----------------------------------useEffectDemo.js------------------------**

/\*function UseEffectDemo() {

const [count, setCount] = useState(0)

const [name, setName] = useState('')

useEffect(() => {

console.log('useEffect - Updating document title ')

document.title = `You clicked ${count} times`

}, [count])

return (

<div>

<input type="text" value={name} onChange={e => setName(e.target.value)} />

<button onClick={() => setCount(count + 1)}>

useEffect - Click {count} times

</button>

</div>

)

}

export default UseEffectDemo\*/

/\*function HookMouse() {

const [x, setX] = useState(0)

const [y, setY] = useState(0)

const logMousePosition = e => {

console.log('Mouse event')

setX(e.clientX)

setY(e.clientY)

}

useEffect(() => {

console.log('useFffect called')

window.addEventListener('mousemove', logMousePosition)

return () => {

console.log('Component unmounting code')

window.removeEventListener('mousemove', logMousePosition)

}

}, [])

return (

<div>

Hooks - X - {x} Y - {y}

</div>

)

}

export default HookMouse\*/

/\*

import React, { Component } from 'react'

class IntervalClassCounter extends Component {

constructor(props) {

super(props)

this.state = {

count: 0

}

}

componentDidMount() {

this.interval = setInterval(this.tick, 1000)

}

componentWillUnmount() {

clearInterval(this.interval)

}

tick = () => {

this.setState({

count: this.state.count + 1

})

}

render() {

return <h1>{this.state.count}</h1>

}

}

export default IntervalClassCounter

\*/

//-----------------------------------------------

/\*

import React, {useState, useEffect} from 'react'

function IntervalHookCounter() {

const [count, setCount] = useState(0)

const tick = () => {

setCount(count + 1)

}

useEffect(() => {

const interval = setInterval(tick, 1000)

return () => {

clearInterval(interval)

}

}, [count])

return (

<div>

{count}

</div>

)

}

export default IntervalHookCounter

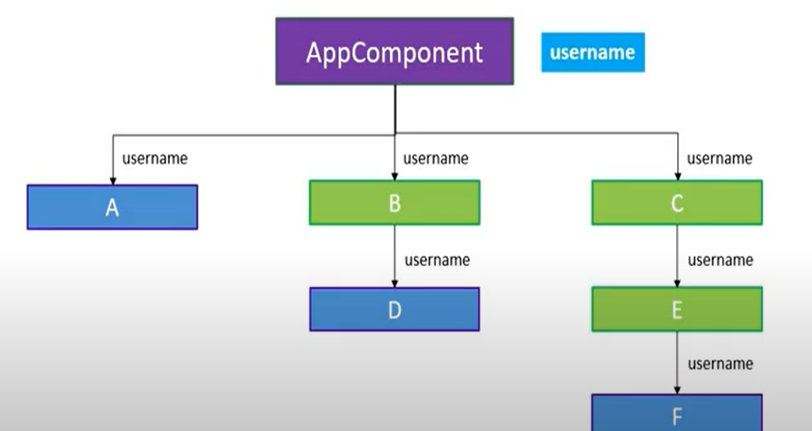
\*/

**-------------------------useContext-------------------------------------**

What is use Context Hook?

Ans-

* Why context compoenent?



---------------------------------X---------------------------------------

------------------------------useContextDemo----------------------------------

/\*import React from 'react'

import ComponentE from './ComponentE'

function ComponentC() {

return <ComponentE />

}

export default ComponentC

\*/

/\*

import React, { useContext } from 'react'

import ComponentF from './ComponentF'

import { UserContext, ChannelContext } from '../App'

function ComponentE() {

const user = useContext(UserContext)

const channel = useContext(ChannelContext)

return <div> User is {user} and channel is {channel}</div>

}

export default ComponentE

\*/

/\*

import React from 'react'

import { UserContext, ChannelContext } from '../App'

function ComponentF() {

return (

<div>

<UserContext.Consumer>

{user => {

return (

<ChannelContext.Consumer>

{channel => {

return <div>User context value {user}, channel context value {channel}</div>

}}

</ChannelContext.Consumer>

)

}}

</UserContext.Consumer>

</div>

)

}

export default ComponentF

\*/

/\*//------------------app.js-------------

import React from 'react'

import './App.css'

import ComponentC from './components/ComponentC'

export const UserContext = React.createContext()

export const ChannelContext = React.createContext()

function App() {

return (

<div className="App">

<UserContext.Provider value={'Vishwas'}>

<ChannelContext.Provider value={'Capgemini'}>

<ComponentC />

</ChannelContext.Provider>

</UserContext.Provider>

</div>

)

}

export default App

\*/

-----------------------------------X----------------------------------------

**Ref demo with class?**

**---------------------------------------X----------------------------------**

**Ref demo use useRef Hooks?**

**Ans-**

import React, { useRef, useEffect } from 'react'

function FocusInput() {

const inputRef = useRef(null)

useEffect(() => {

inputRef.current.focus()

}, [])

return (

<div>

<input ref={inputRef} type="text" />

</div>

)

}

export default FocusInput

------------------------------------X---------------------------------------

-------------------------------Pure Componenet Demo---------------

import React, { Component, Pure } from 'react'

import RegComp from './RegComp'

import PureComp from './PureComp'

import MemoComp from './MemoComp'

class ParentComp extends Component {

constructor(props) {

super(props)

this.state = {

name: 'Vishwas'

}

}

componentDidMount() {

setInterval(() => {

this.setState({

name: 'Vishwas'

})

}, 2000)

}

render() {

console.log('\*\*\*\*\*\*\*\*\*Parent Comp render\*\*\*\*\*\*\*\*\*\*\*\*')

return (

<div>

{/\* <RegComp name={this.state.name} />

<PureComp name={this.state.name} /> \*/}

<MemoComp name={this.state.name} />

</div>

)

}

}

export default ParentComp

---------------------------------X----------------------------------------

import React, { Component } from 'react'

class RegComp extends Component {

render() {

console.log('Regular Comp render')

return (

<div>

Regular Component {this.props.name}

</div>

)

}

}

export default RegComp

---------------------------------------X--------------------------------------

import React, { PureComponent } from 'react'

class PureComp extends PureComponent {

render() {

console.log('Pure Comp render')

return (

<div>

Pure Component {this.props.name}

</div>

)

}

}

export default PureComp

**----------------------------------X---------------------------------------**

**LifeCycleDemo-----------------------------------------------**

import React, { Component } from 'react'

class LifecycleB extends Component {

constructor(props) {

super(props)

this.state = {

name: Vaishali

}

console.log('LifecycleB constructor')

}

static getDerivedStateFromProps(props, state) {

console.log('LifecycleB getDerivedStateFromProps')

return null

}

componentDidMount() {

console.log('LifecycleB componentDidMount')

}

shouldComponentUpdate() {

console.log('LifecycleB shouldComponentUpdate')

return true

}

getSnapshotBeforeUpdate(prevProps, prevState) {

console.log('LifecycleB getSnapshotBeforeUpdate')

return null

}

componentDidUpdate(prevProps, prevState, snapshot) {

console.log('LifecycleB componentDidUpdate')

}

render() {

console.log('LifecycleB render')

return (

<div>

LifecycleB

</div>

)

}

}

export default LifecycleB

--------------------------------------------------------------------X-------------------

import React, { Component } from 'react'

import LifecycleB from './LifecycleB'

class LifecycleA extends Component {

constructor(props) {

super(props)

this.state = {

name: Vaishali

}

console.log('LifecycleA constructor')

}

static getDerivedStateFromProps(props, state) {

console.log('LifecycleA getDerivedStateFromProps')

return null

}

componentDidMount() {

console.log('LifecycleA componentDidMount')

}

shouldComponentUpdate() {

console.log('LifecycleA shouldComponentUpdate')

return true

}

getSnapshotBeforeUpdate(prevProps, prevState) {

console.log('LifecycleA getSnapshotBeforeUpdate')

return null

}

componentDidUpdate(prevProps, prevState, snapshot) {

console.log('LifecycleA componentDidUpdate')

}

changeState = () => {

this.setState({

name: 'Codevolution'

})

}

render() {

console.log('LifecycleA render')

return (

<div>

<button onClick={this.changeState}>Change state</button>

LifecycleA<LifecycleB />

</div>

)

}

}

export default LifecycleA

---------------------------------X----------------------------------

What is closure?

What is self invocation function?

What is difference between lexiclas scope and closure?

What is callback?

What is ReactDom.render?Where does it render ?In vrirtual Dom or Real dom?