


Chapter 05- Lets get hooked

WHILE IMPORTING NAMED EXPORTS WE NEED TO ADD A {}
WHILE IMPORTING IT

```
45  
46  
47 Two types of Export/Import  
48  
49   
50 - Default Export/Import  
51  
52 export default Component;  
53 import Component from "path";  
54  
55  
56 - Named Export/Import  
57  
58 export const Component;  
59 import {Component} from "path";
```

Filter

Out top rated resto who have rating more than 4

useState

For super powerful variables we use states using useState hook
React is a utility javascript function.

Usestate is a javascript function that is written inside react-dom file

Whenever we import useState in a component, we always try to import it like this

```
import { useState } from "react";
```

Because useState is exported as a named function in react library.

If you will check <https://unpkg.com/react@18/umd/react.development.js>

React cdn link, this is exported as a named function.

Why we are using state and why we are not using normal javascript variable?

If we are using a normal variable, even if you change it, it will get updated but it will not be updated on the ui layer.

So we need a state that has a setmethod who will be called on an event listener and update that state and that state change will get reflected on the ui layer also.

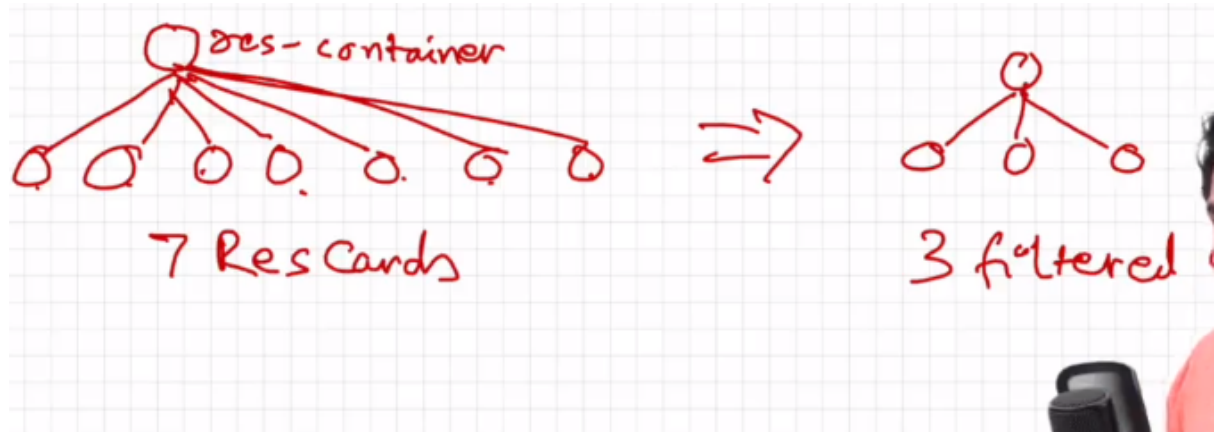
React will make sure that our ui layer is in sync with data layer. As soon as data layer updates, ui layer will also update.

Whenever a state variable changes, react will re render the component.

Why React works so fast?

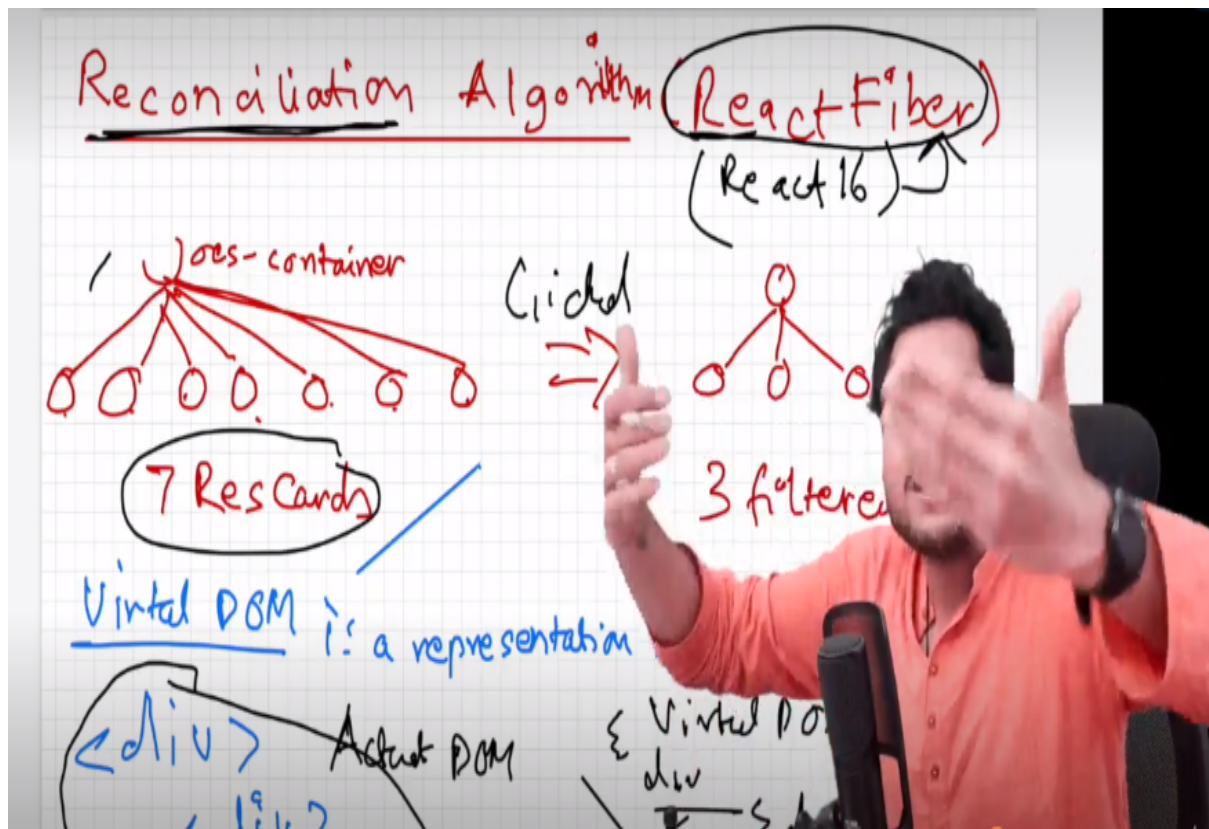
React uses Reconciliation algorithm.

Suppose we have a dom



React creates a virtual Dom (fake dom) which is representation of actual DOM

Diff algorithm finds the difference between old virtual dom and updated virtual dom, finds the difference and made those changes to real dom



Whenever there is change in any state variable , it will find the difference between those virtual dom and put those changes in real dom by re rendering the component

Reconciliation is the process by which React updates the UI to reflect changes in the component state. The reconciliation algorithm is the set of rules that [React](#) uses to determine how to update the UI in the most efficient way possible.

[React](#) uses a virtual DOM (Document Object Model) to update the UI. The virtual DOM is a lightweight in-memory representation of the real DOM, which allows [React](#) to make changes to the UI without manipulating the actual DOM. This makes updates faster, as changing the virtual DOM is less expensive than changing the real DOM.

The reconciliation algorithm works by comparing the current virtual DOM tree to the updated virtual DOM tree, and making the minimum number of changes necessary to bring the virtual DOM in

line with the updated state.

Tree diffing: [React](#) compares the current virtual DOM tree with the updated virtual DOM tree, and identifies the minimum number of changes necessary to bring the virtual DOM in line with the updated state.

Batching: React batches multiple changes into a single update, reducing the number of updates to the virtual DOM and, in turn, the real DOM.

The reconciliation algorithm is a critical part of React's performance and helps make React one of the fastest and most efficient JavaScript libraries for building user interfaces.

After the reconciler compares the current and updated virtual DOM, it identifies the differences and makes the necessary changes to the virtual DOM to bring it in line with the updated state.

The reconciliation process ensures that the virtual DOM accurately reflects the current state of the components so that the UI remains up-to-date and in line with the state of the application.

In summary, the reconciler updates the virtual DOM after the reconciliation process to keep it accurate, optimise updates to the real DOM, and ensure that the UI remains in sync with the state of the components.