

EPISODE 5: LET'S GET HOOKED

- We can build everything we built in react using html also.
- But in react we can build it with less code.

PART 01:

In this part we will clean all the mess in code of episode 4.

lets separate each component into different file

Importing ways:

```
import Header from './components/Header'  
import Header from './components/Header.js'
```

importing as Header or Header.js both works same.

TYPES OF EXPORT:

Default Export:

- For every component there is only one default export. We export at end of the component.

```

1 import { LOGO } from "../utils/constants";
2
3 const Header = () =>{
4   return(
5     <div className="header">
6       <div className="logo">
7         <img id="logo" src={LOGO} alt="logo"/>
8       </div>
9       <div className="nav-items">
10        <ul>
11          <li>Home</li>
12          <li>About us</li>
13          <li>Contact Us</li>
14          <li>Cart</li>
15        </ul>
16      </div>
17    </div>
18  )
19 }
20 export default Header;

```

Naming Export:

- We use it when we need to export more than one variable or component from a file.

```

export const RECIPE_IMAGE = "https://media-assets.swiggy.com/swiggy/image/upload/fl_lossy,f_auto,q_auto,w_660/"
export const LOGO = "https://encrypted-tbn0.gstatic.com/images?q=tbn:ANd9GcQwv45iAbWwD5glterGTBI06Q6pnjukcFkGw&usqp=CAU"

```

We can even use naming export for Components but we prefer default export.

TYPES OF IMPORT:

Default Import:

Syntax:

import Component from "path"

```
2 import React from "react";
3 import ReactDOM from "react-dom/client";
4 import Header from "../components/Header";
5 import Body from "../components/Body";
6
```

Naming Import :

Syntax:

```
import { Variable } from "path"
```

```
2
3 import { data } from "../utils/data";
4
5
```

REACT HOOKS:

- React hooks are nothing but a normal javascript utility functions .

useState ():

- A normal javascript function which is given to us by react
- It is used to create superpowerful state variables in react

why it is called as state variable?

Because it maintains the state of your application

- The scope of local state variable is inside that component only.

- Syntax for importing useState is as follows
`import { useState } from "react";`
`const [resData,setResData]= useState(data)`
`<button className='top-res'`
`onClick={()=>setResData(data.filter((item)=>`
`item.card.info.avgRating>4`
`))}`
`>`
-

Summary :

General way of creating variables in react.

`Const list = []`

`list.push("abc")`

- or
- `let list2 = []`
- `list2=["xyz"]`
- Even if the variable list or list2 is modified it won't reflect on the UI , the variable will be modified but it won't reflect on UI.
- So when we use useState it keeps the sync with the UI layer and reflects on the UI and the UI will update.
- So as soon as state variable changes it will automatically refresh our component which is known as render.

-

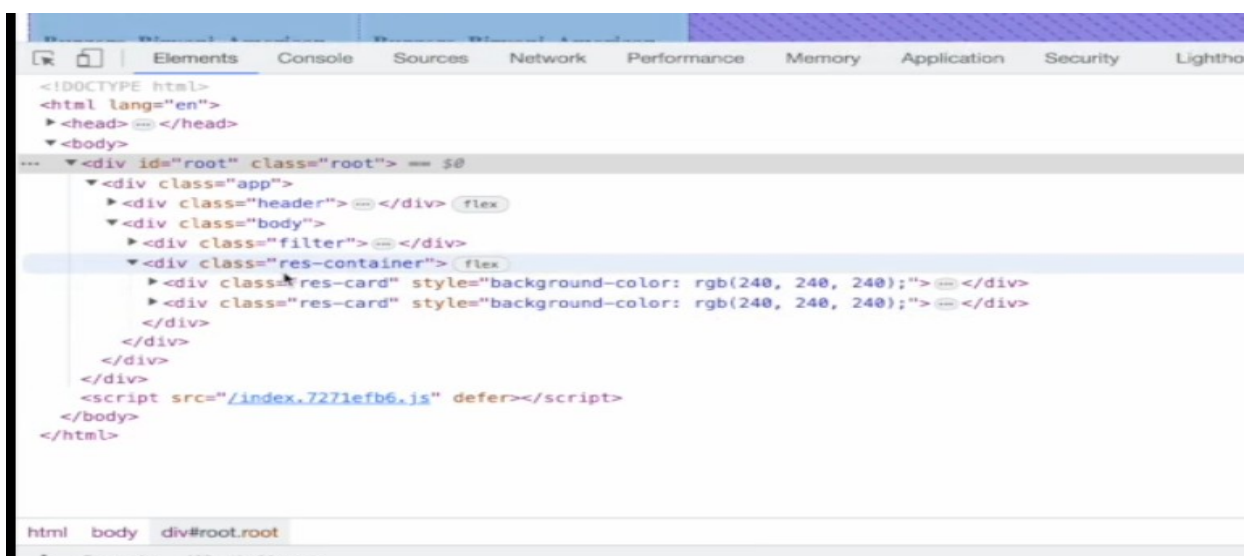
So here it comes whenever the state variable updates react rerenders the component.

Lets see this ,before updating the state variable.there are 3 rescards

It keeps the data layer in sync with the UI layer

```
<!DOCTYPE html>
<html lang="en">
  <head>
  </head>
  <body>
    <div id="root" class="root">
      <div class="app">
        <div class="header">
        </div>
        <div class="body">
          <div class="filter">
          </div>
          <div class="res-container">
            <div class="res-card" style="background-color: rgb(240, 240, 240);">
            </div>
            <div class="res-card" style="background-color: rgb(240, 240, 240);">
            </div>
            <div class="res-card" style="background-color: rgb(240, 240, 240);">
            </div>
          </div>
        </div>
      </div>
      <script src="/index.7271efb6.js" defer></script>
    </body>
  </html>
```

we can observe there are 3 rescards , after clicking the toprated restaurents button the state variable gets updated and rerenders the component like below



The screenshot shows the Chrome DevTools 'Elements' panel. The DOM tree is expanded to show the following structure:

```
<!DOCTYPE html>
<html lang="en">
  <head>
  </head>
  <body>
    <div id="root" class="root">
      <div class="app">
        <div class="header">
        </div>
        <div class="body">
          <div class="filter">
          </div>
          <div class="res-container">
            <div class="res-card" style="background-color: rgb(240, 240, 240);">
            </div>
            <div class="res-card" style="background-color: rgb(240, 240, 240);">
            </div>
          </div>
        </div>
      </div>
      <script src="/index.7271efb6.js" defer></script>
    </body>
  </html>
```

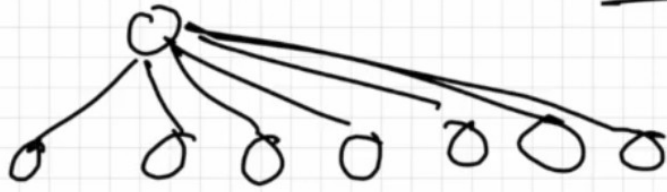
The breadcrumb at the bottom of the panel indicates the path: `html > body > div#root.root`.

here we are seeing only 2 res-cards only in the dom.

RECONCILATION :

- React uses an algorithm **React Fiber** came in REACT16 for reconciliation.
- In REACT16 a new algorithm to update the dom came out and it is known as React Fiber.
- The term for changing data on the screen refers to **Reconciliation**.
- This react fiber is the new way of finding the diff of virtual dom and updating the dom

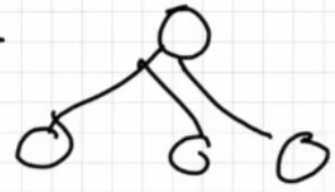
Diff. Algn



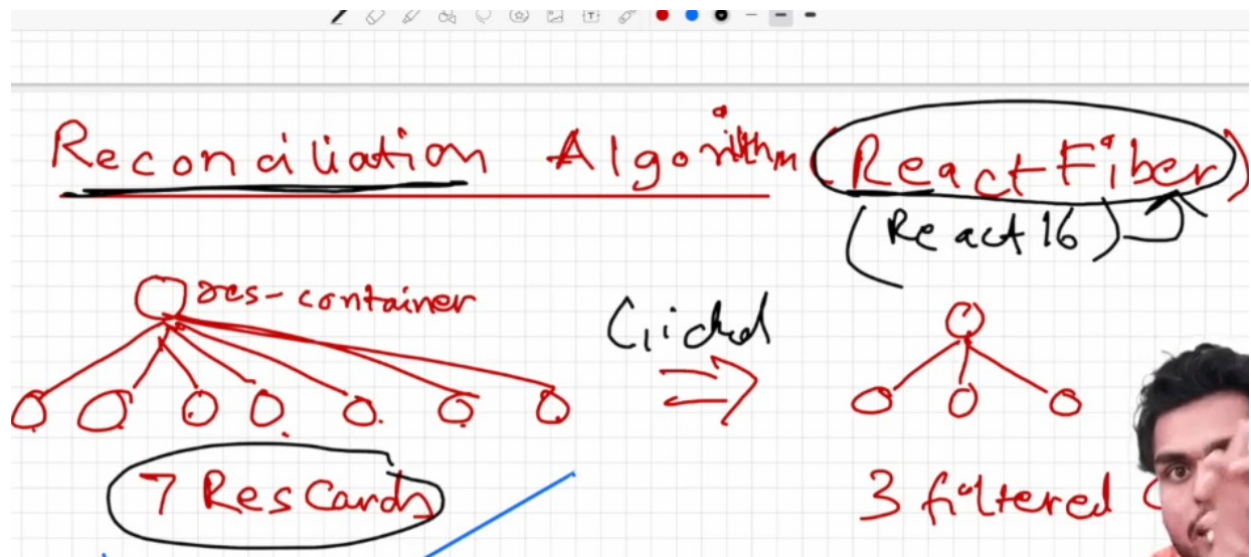
7 RC

OLD

Clicked



NEW



for example if we have 7 restaurants in total and when we click on button top restaurants then the ui will updated to 3.

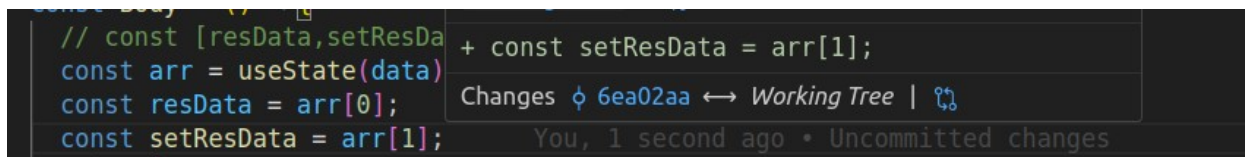
How will it happens?

Generally the react hold old virtual dom of 7 rescards. Whenever the state variable changes new virtual dom is created with 3 rescards .


Now it calculates and finds the difference between two doms and update the actual dom , the difference here is 4 nodes.

- Virtual dom is nothing but the object form (ex:the previous way we wrote react in object format like createElement)

- Actual dom will be a html structure.
- Virtual dom is in object form because it is easy to compare two objects rather than being in html format.
- React is faster because it do effiecient dom manipulation.
- We can also write in the below way it still works.



```
// const [resData, setResData] = useState(data);  
const arr = useState(data);  
const resData = arr[0];  
const setResData = arr[1];  
+ const setResData = arr[1];
```

Changes 6ea02aa ↔ Working Tree | 

You, 1 second ago • Uncommitted changes

- State variable is so powerful because whenever it keeps track on the variable whenever it changes it triggers diff algorithm checks the difference and updates the dom.