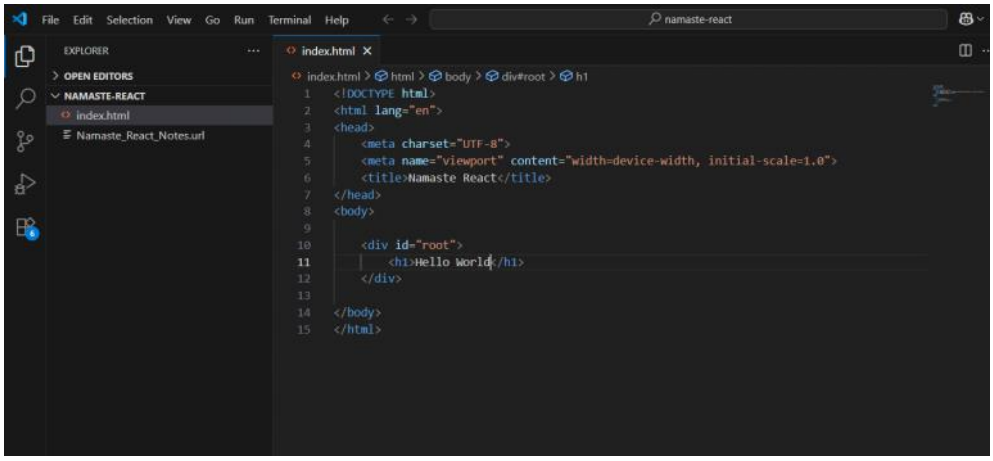
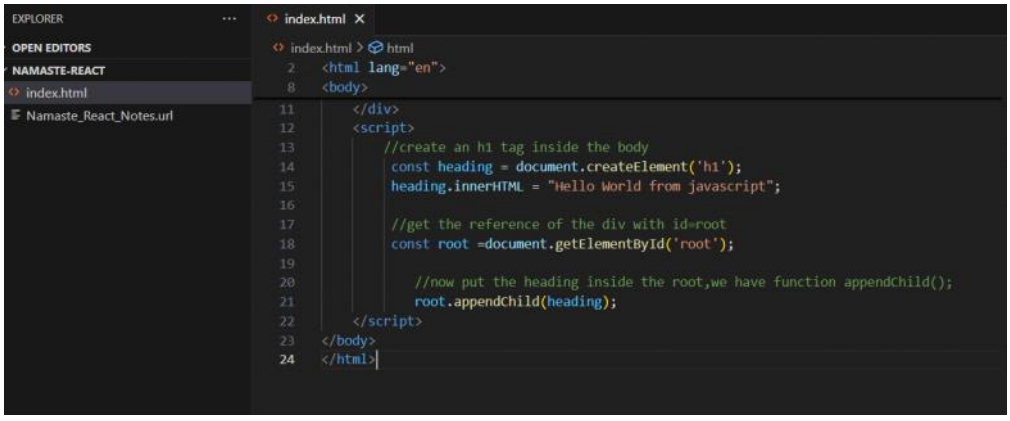
Namaste React

26 September 2025 22:42

Episode 01 Part 02
==========
->create one folder namaste-react. Open it in VsCode.
1)first make one html page & write hello world.



2)build hello world using javascript.
=>

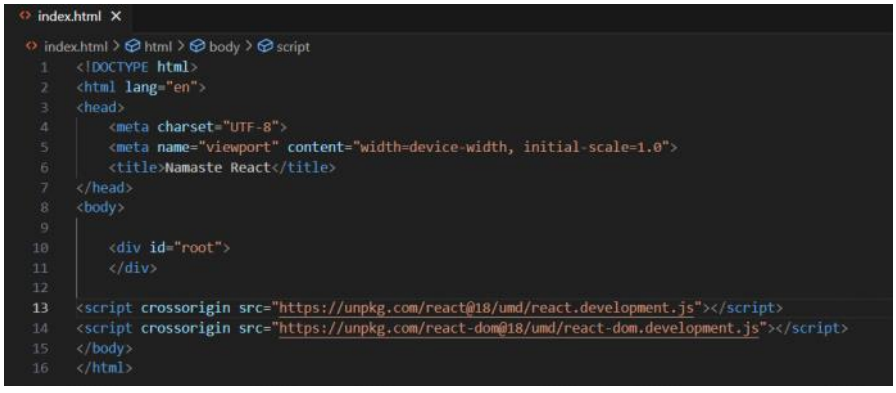


3)create basic hello world program using react
=>
->Browser understand only HTML,CSS,JS not react.
->first we need to download react in our project.
-> go to cdn react. It is content delivery network which simply pulling react project into our project.
Cdn is where our react library hosted.

Copy this link from CDN website and paste it just above ending body tag.

Both React and ReactDOM are available over a CDN.

```
<script crossorigin src="https://unpkg.com/react@18/umd/react.development.js"></script>
<script crossorigin src="https://unpkg.com/react-dom@18/umd/react-dom.development.js"></script>
```



->if we open any one of CDN link, we see code of react which is written in pure javascript.
->React is an JavaScript library.

->it is core react.

```
<script crossorigin src="https://unpkg.com/react@18/umd/react.development.js"></script>
```

->it is useful for Dom operation.

```
<script crossorigin src="https://unpkg.com/react-dom@18/umd/react-dom.development.js"></script>
```

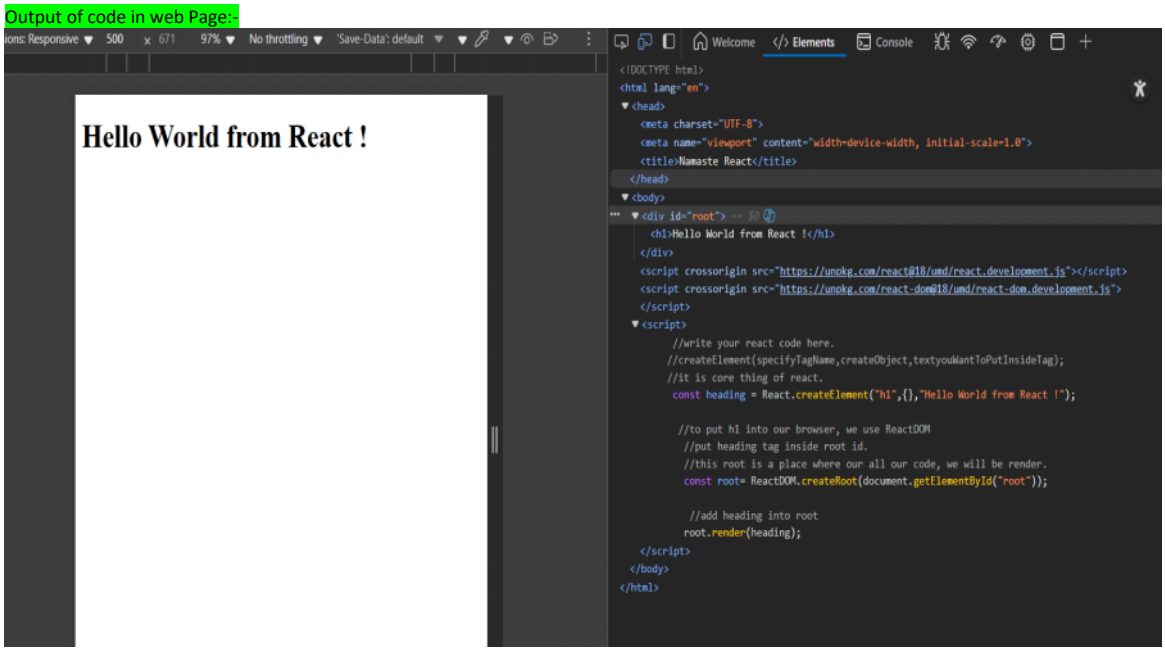
Q.) Why they separate react file ?
->bcz react does not only work on browser but also works on mobile also as react-native.
But main file of react is core react which can be used all over like browser & mobile.

Episode 01 Part 02
==========
3)create basic hello world program using react
=>

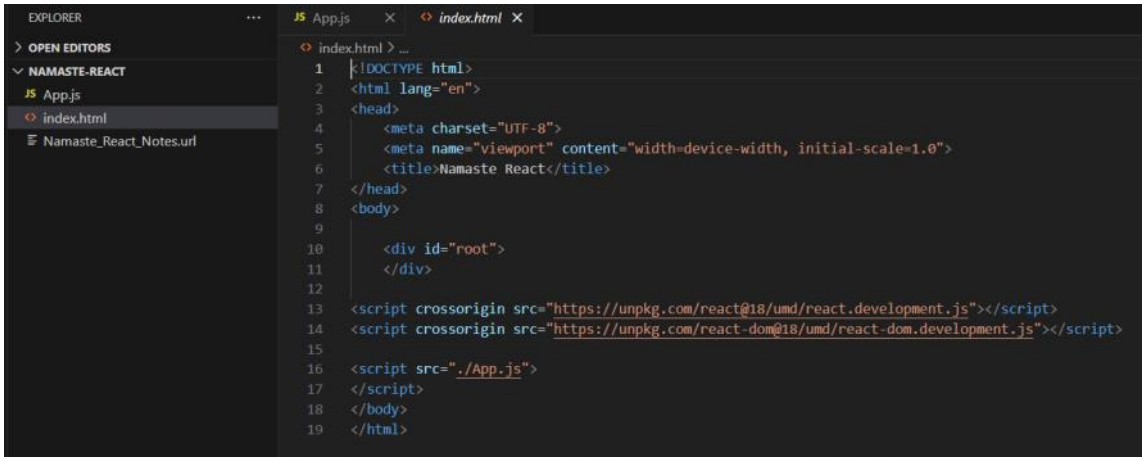
index.html

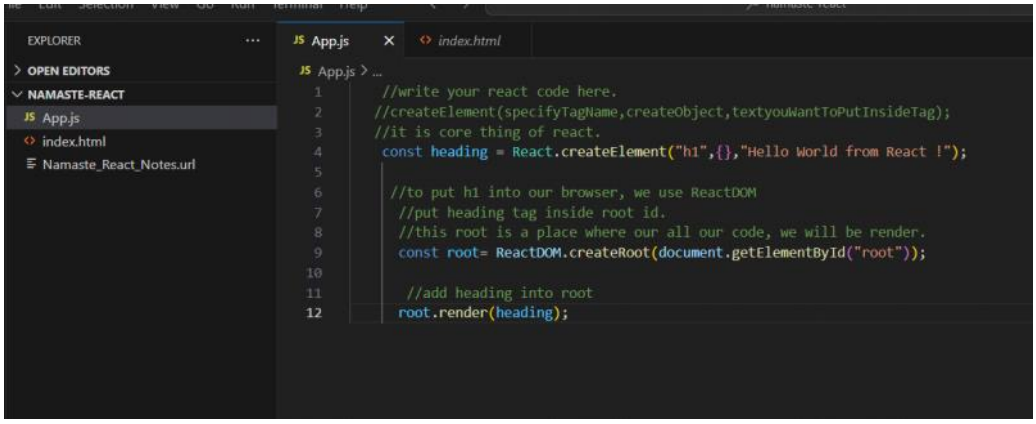
```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Namaste React</title>
</head>
<body>

    <div id="root">
    </div>

<script crossorigin src="https://unpkg.com/react@18/umd/react.development.js"></script>
<script crossorigin src="https://unpkg.com/react-dom@18/umd/react-dom.development.js"></script>

<script>
    //write your react code here.
   //createElement(specifyTagName,createObject,textyouWantToPutInsideTag);
   //it is core thing of react.
    const heading = React.createElement("h1",{},"Hello World from React !");

    //to put h1 into our browser, we use ReactDOM
     //put heading tag inside root id.
     //this root is a place where our all our code, we will be render.
     const root= ReactDOM.createRoot(document.getElementById("root"));

      //add heading into root
     root.render(heading);

</script>
</body>
</html>
```

Output of code in web Page:-

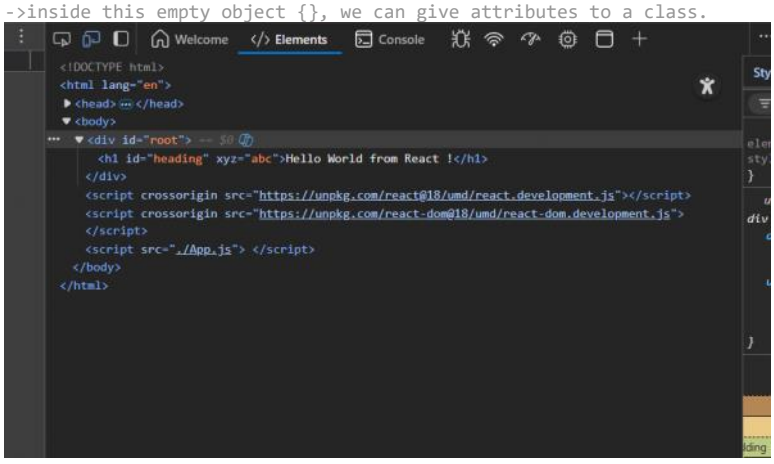


Episode 01 Part 03
==========
->we have write our all react code inside the script tag.
But it is not good practice. So we create one js file and write our code there.

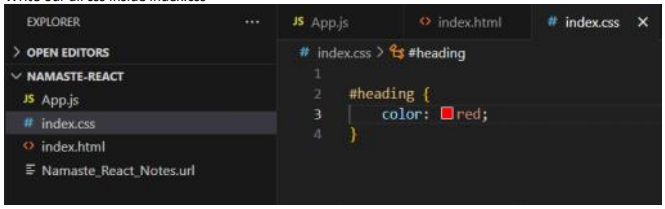





```
  const heading = React.createElement("h1",{},"Hello World from React !");
->inside this empty object {}, we can give attributes to a class.
```

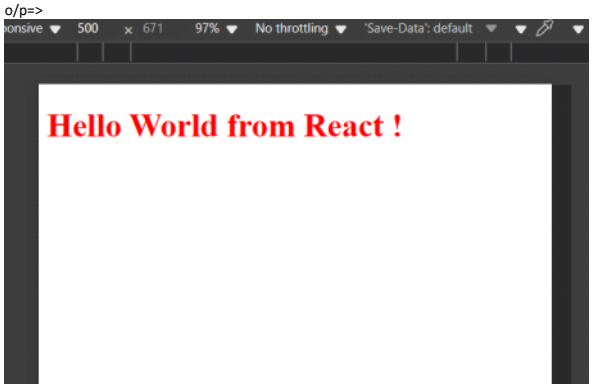


=>change the color of text in react using CSS.
Make one file index.css and link this file with index.html page.
Write our all css inside index.css



```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <link rel="stylesheet" href="./index.css" > </link>
    <title> Namaste React</title>
</head>
<body>

    <div id="root">
    </div>

<script crossorigin src="https://unpkg.com/react@18/umd/react.development.js"></script>
<script crossorigin src="https://unpkg.com/react-dom@18/umd/react-dom.development.js"></script>

<script src="./App.js">
</script>
</body>
</html>
```

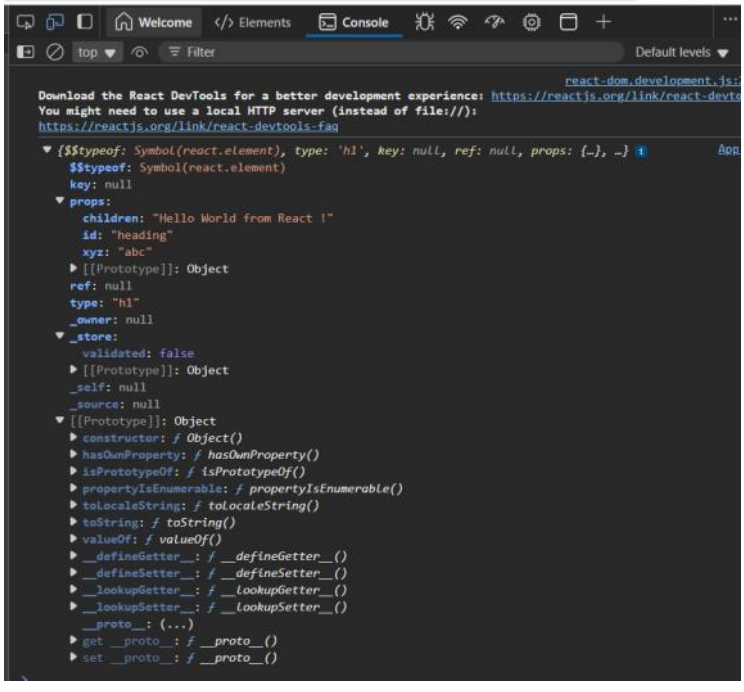
o/p=>



=>
We create heading react element using react.
const heading = React.createElement("h1",{ id:"heading",xyz:"abc" },"Hello World from React !");

console.log(heading); //heading is an react element which is nothing but a javascript object.

=>



=>props are children + attributes that we pass in.
Ex:-
const heading = React.createElement(
"h1",
{ id:"heading",xyz:"abc" }, //2nd is attributes
"Hello World from React !" //3rd is children
);

=>In simple words, render() convert react element into appropriate tag & put it into html page.
root.render(heading); //take the react element like heading react element and put it inside root element.

Episode 01 Part 04

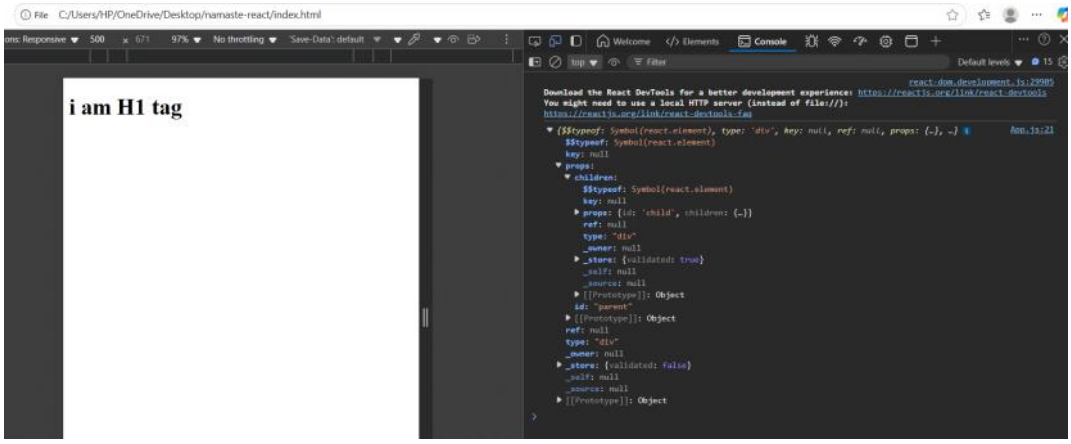
==========

=>Create this HTML nested structure using React.
Ex:-

```
App.js > ...
/*
<div id="parent">
  <div id="child">
  <h1>i am h1 tag</h1>
  </div>
</div>
*/
```

Code=>

```
EXPLORER
OPEN EDITORS
NAMASTE-REACT
App.js
index.css
index.html
Namaste_React_Notes.url

App.js  index.html  index.css
App.js > ...
    </div>
  </div>
*/


const parent= React.createElement(
  "div",
  {id:"parent"},
  React.createElement(
            "div",
            {id:"child"},
            React.createElement("h1",{},"i am H1 tag")
              )
);

  console.log(parent);

const root= ReactDOM.createRoot(document.getElementById("root"));

root.render(parent);
```



Note:=>
React.createElement() creating React Element is an object.
This React object becomes HTML that browser understands.

React Element(Object) ====> HTML(Browser Understands)

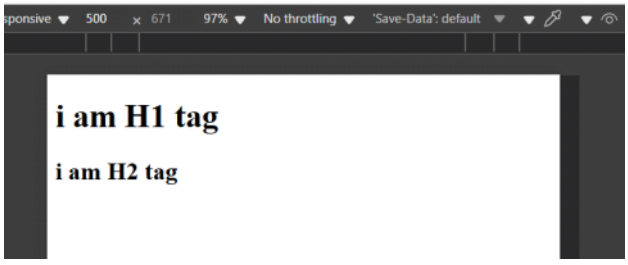
=>to Create multiple siblings,
We need to pass 3rd argument in array form React.createElement("",{},[]);
Ex:-

```
/*
<div id="parent">
  <div id="child">
  <h1>i am h1 tag</h1>
   <h2>i am h2 tag</h2>
  </div>
</div>
*/
```

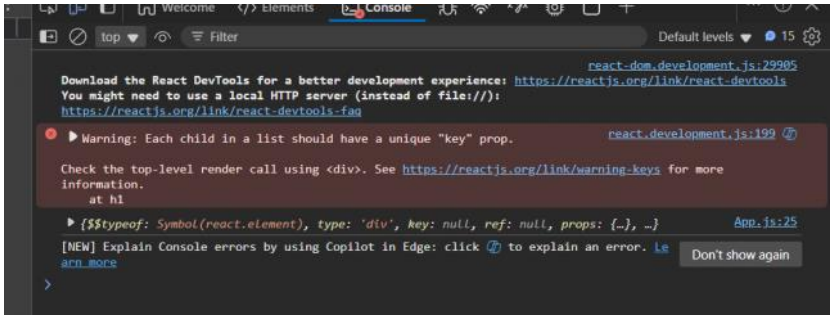
o/p=>

```
/*
        <div id="parent">
          <div id="child">
          <h1>i am h1 tag</h1>
           <h2>i am h2 tag</h2>
          </div>
        </div>
*/


const parent= React.createElement(
    "div",
     {id:"parent"},
     React.createElement(
                    "div",
                    {id:"child"},
                    [ //Array of children
                    React.createElement("h1",{},"i am H1 tag"),
                    React.createElement("h2",{},"i am H2 tag")
                    ]
                      )
);

     console.log(parent);

const root= ReactDOM.createRoot(document.getElementById("root"));

root.render(parent);
```

o/p on web page=>



But on console we are getting, error like

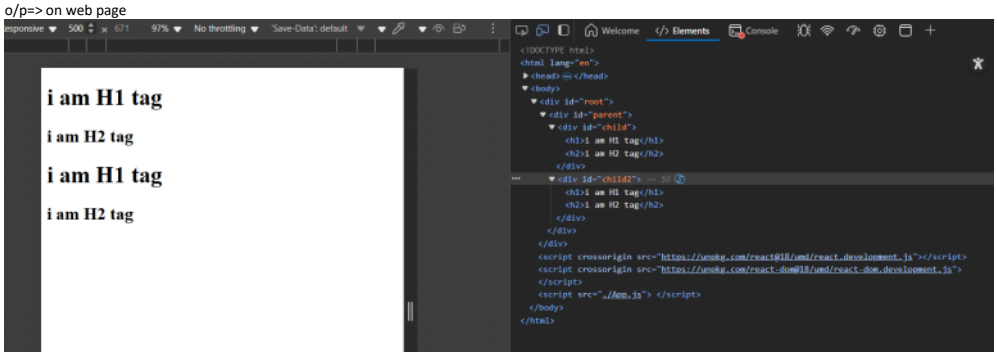


=> to create this HTML page,

Code=>

```
*/


const parent= React.createElement( "div", {id:"parent"},[
 React.createElement( "div", {id:"child"},  [ //Array of children
                      React.createElement("h1",{},"i am H1 tag"),
                      React.createElement("h2",{},"i am H2 tag")
                     ] ),
   React.createElement( "div", {id:"child2"},  [ //Array of children
                      React.createElement("h1",{},"i am H1 tag"),
                      React.createElement("h2",{},"i am H2 tag")
                     ] )
                    ]);

      console.log(parent);

const root= ReactDOM.createRoot(document.getElementById("root"));

root.render(parent);
```

o/p=> on web page



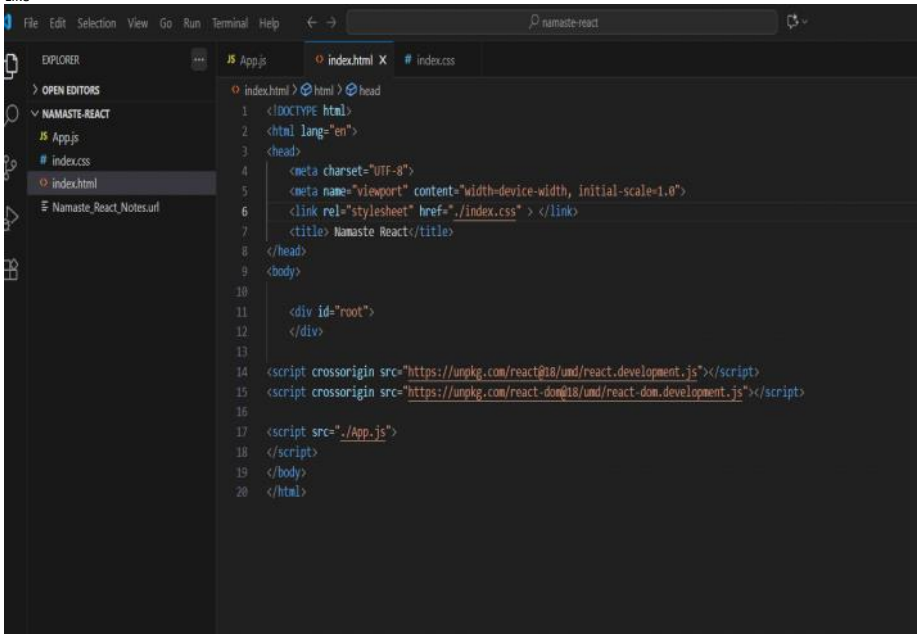
App.js

```
const parent= React.createElement( "div", {id:"parent"},[
 React.createElement( "div", {id:"child"},  [ //Array of children
                      React.createElement("h1",{},"i am H1 tag"),
                      React.createElement("h2",{},"i am H2 tag")
                     ] ),
   React.createElement( "div", {id:"child2"},  [ //Array of children
                      React.createElement("h1",{},"i am H1 tag"),
                      React.createElement("h2",{},"i am H2 tag")
                     ] )
                    ]);
```

But our code is very messy for complex structure to understand.
To solve this messy code problem we have JSX.
=>JSX makes our life easy when we want to create tags.

Episode 01 Part 05
==========

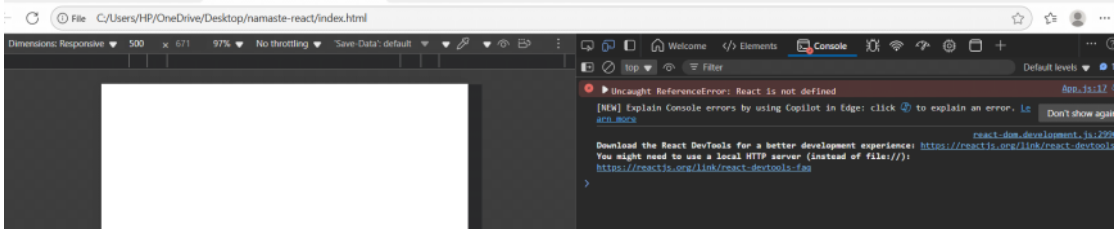
Until now we wrote code, we put libarary first then we add App.js file at end.
Like



But what if I change the order will my code will be executed ?
Like

```
App.js   index.html   index.css
index.html > html > body > script
1   <!DOCTYPE html>
2   <html lang="en">
3   <head>
4       <meta charset="UTF-8">
5       <meta name="viewport" content="width=device-width, initial-scale=1.0">
6       <link rel="stylesheet" href="./index.css" > </link>
7       <title> Namaste React</title>
8   </head>
9   <body>
10
11      <div id="root">
12      </div>
13
14
15  <script src="./App.js"></script>
16  <script crossorigin src="https://unpkg.com/react@18/umd/react.development.js"></script>
17  <script crossorigin src="https://unpkg.com/react-dom@18/umd/react-dom.development.js"></script>
18
19
20  </body>
21  </html>
```

We get an error on console like "React is not defined" so order matters.



So, always use React library before App.js.
=>I have code in my root tag like

```
ndex.html > html > body
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <link rel="stylesheet" href="./index.css" > </link>
    <title> Namaste React</title>
</head>
<body>

    <div id="root">
        <h1>Praful is here </h1>
    </div>



<script crossorigin src="https://unpkg.com/react@18/umd/react.development.js"></script>
<script crossorigin src="https://unpkg.com/react-dom@18/umd/react-dom.development.js"></script>
<script src="./App.js"></script>

</body>
</html>
```

Whole code inside the root tag will be replaced by render() when I passed react object into it
.

```
const parent= React.createElement( "div", {id:"parent"},[
 React.createElement( "div", {id:"child"},  [ //Array of children
                    React.createElement("h1",{},"i am H1 tag"),
                    React.createElement("h2",{},"i am H2 tag")
                  ] ),
   React.createElement( "div", {id:"child2"},  [ //Array of children
                    React.createElement("h1",{},"i am H1 tag"),
                    React.createElement("h2",{},"i am H2 tag")
                  ] )
                 ]);

    console.log(parent);

const root= ReactDOM.createRoot(document.getElementById("root"));

root.render(parent);
```

=>As we know our HTML code executed line by line from top to bottom,
So, first Praful is here is will load on web page. Then we have React library into it.
So, when our HTML code reaches to React,ReactDOM into our app.
It will load all libraries when our HTML code rwaches to App.js file it will execute the javascript code.
The root.render() method executed it will replace all info which is already present in root tag by react code which is written in App.js.

=>

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <link rel="stylesheet" href="./index.css" > </link>
    <title> Namaste React</title>
</head>
<body>

    <h1>Namaste React</h1>

    <div id="root">
        <h1>Praful is here </h1>
    </div>

        <h1>Namaste React End</h1>


<script crossorigin src="https://unpkg.com/react@18/umd/react.development.js"></script>
<script crossorigin src="https://unpkg.com/react-dom@18/umd/react-dom.development.js"></script>
<script src="./App.js"></script>

</body>
</html>
```

Only that will be replaced by react code which has id is root bcz we write react code to render our page on that id root.
Not other code.

=>Diff between library vs Framework
==========
->React library can work on small portion of code not on whole code.
But framework will work on whole code.

++
EPISODE 2: Igniting our App
++

Episode 02 Part 01
==========
->First push our episode-01 code into github.
1)create one reposioty in github named as "namaste-react" make as public.
2)then use following command in VS-code editor.

```
PS C:\Users\HP\OneDrive\Desktop\namaste-react> git init
Initialized empty Git repository in C:/Users/HP/OneDrive/Desktop/namaste-react/.git/
PS C:\Users\HP\OneDrive\Desktop\namaste-react> git branch -M main
PS C:\Users\HP\OneDrive\Desktop\namaste-react> git add .
PS C:\Users\HP\OneDrive\Desktop\namaste-react> git commit -m "episode-01"
[main (root-commit) 104c2e0] episode-01
 5 files changed, 78 insertions(+)
 create mode 100644 App.js
 create mode 100644 Namaste_React_Notes.url
 create mode 100644 README.md
 create mode 100644 index.css
 create mode 100644 index.html
PS C:\Users\HP\OneDrive\Desktop\namaste-react> git remote add origin https://github.com/praful-shahane/namaste-react.git
PS C:\Users\HP\OneDrive\Desktop\namaste-react> git push origin main
Enumerating objects: 7, done.
Counting objects: 100% (7/7), done.
Delta compression using up to 12 threads
Compressing objects: 100% (5/5), done.
Writing objects: 100% (7/7), 1.36 KiB | 465.00 KiB/s, done.
```

```
        </div>
    */



    const parent= React.createElement( "div", {id:"parent"},[
     React.createElement( "div", {id:"child"},  [ //Array of children
                          React.createElement("h1",{},"i am H1 tag"),
                          React.createElement("h2",{},"i am H2 tag")
                         ] ),
       React.createElement( "div", {id:"child2"},  [ //Array of children
                          React.createElement("h1",{},"i am H1 tag"),
                          React.createElement("h2",{},"i am H2 tag")
                         ] )
                        ]);

        console.log(parent);

    const root= ReactDOM.createRoot(document.getElementById("root"));

    root.render(parent);
```

The code we wrote is episode 1, is not optimized code so this code we not able to move to Production.

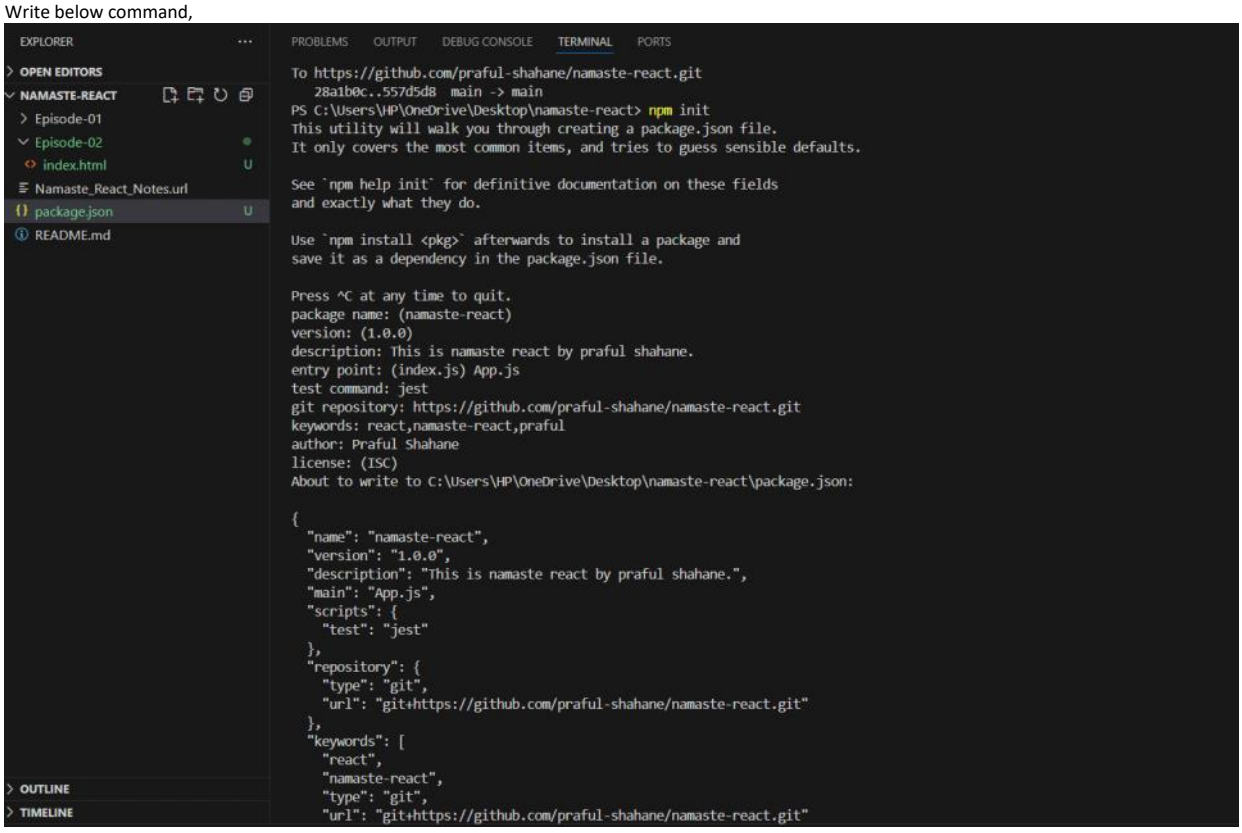
->only React could not make our app 100% optimized but if we use packages out application will be optimized.

NPM:-

Official website => npmjs.com

NPM is not node package manager.
NPM does not have a full form.
NPM manages packages but it does not stands for node package manager.
NPM is a standard repository where all packages are hosted are there.

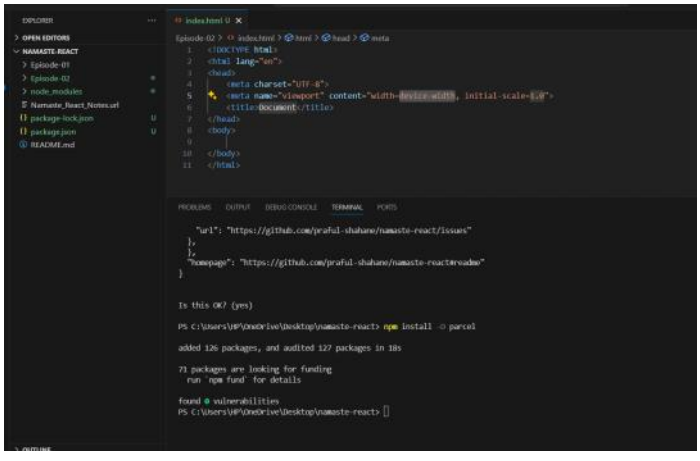
=>add NPM in our project.
Write below command,



->once done package.json file is create in our project this file is configuration for our NPM.
->NOTE:: package.json is configuration for NPM.

->Most important package in our project is bundler.
What is Bundler ?
->when we have normal HTML,CSS,JS files. Whole code needs to be bundles together/minified/cached/compressed/clean before it can be sent to production.
So, bundler helps to do all these things.
Webpack, parcel, vite this are bundelrs.
Job of bundlers is bundlers basically bundles our app. It packages our app so that it can be shipped to production.

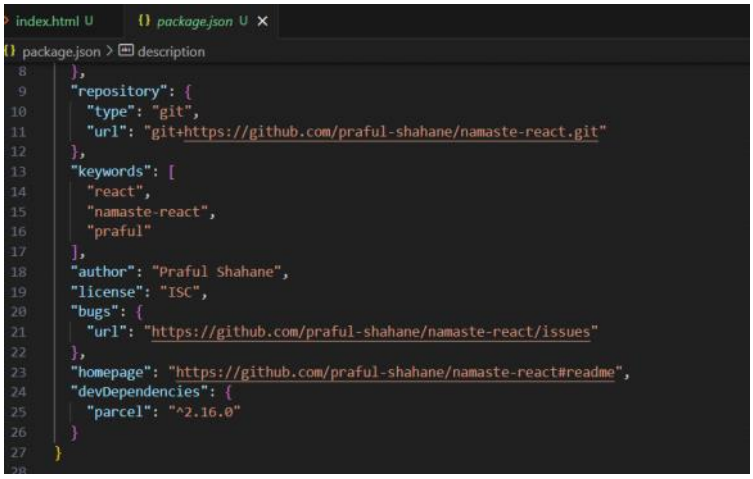
->we will be using parcel bundler & easy to configure.



-> **command** => npm install -D parcel
Q.)What is meaning of D ?
=> bcz I want to install parcel's dev dependencies only for development.
Due to this reason we wrote command "npm install -D parcel"
After command execution our parcel is installed from NPM repository.

Parcel app can have two dependencies , one is dev dependencies & one is normal dependency.
Dev Dependency -> it is generally required for our development phase.
Normal dependency ->it used in production also.

Package.json is a configuration for NPM it basically keeps a tracks of all the dependencies/packages our code is using.



Q.)Diff of ^ and ~ ?

```
"devDependencies": {
  "parcel": "^2.16.0"
}
```

(^) -> this sign is called caret. We can also put (~) tilde also.
Ex:-
Let say we have current version of parcel is 2.16.0, if we use ^, if new minor version like 2.16.3 of parcel comes in future, it will autonatically upgarde it,
But if we use ~(tilde), if major version like 2.17.0 comes in future, it will automatically upgrade it.
So, always use ^(caret) bcz it okay to upgrade minor version but don't upgrade major version.
But if don't want any upgrade remove tilde & caret. Just add version like

```
"devDependencies": {
  "parcel": "2.16.0"
}
```

->Package.json is a configuration for NPM it basically keeps a tracks of all the dependencies/packages our code is using.

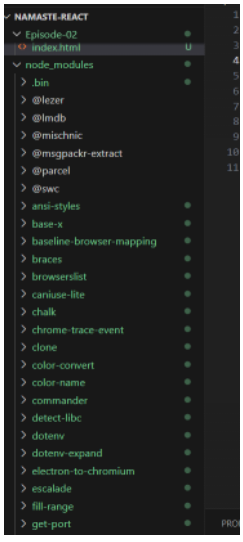
package-lock.json

->After we install, parcel, we got one file also package-lock.json.
->Package-lock.json keeps a track of exact version that is beign installed.
Basically package-lock.json keeps a hash to verify that whatever is there in my machine is the same version must be deployed on to production.
It keeps a track of all version of exact version of all the dependencies

Node Modules

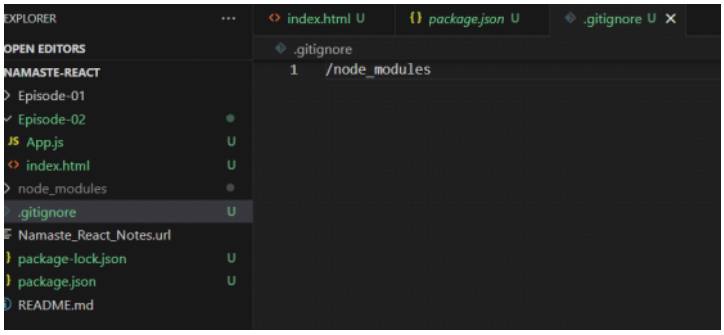
->when we install parcel, node modules also create in our project.
Node modules contain all code which we fetch from NPM.
Node modules contains packages that we need to code our application.
->Node modules is like a database where all our packages are exist.

->we install just parcel dependency, but why we getting other dependencies as well ?
Because parcel dependent on other dependencies & that other dependencies dependent on other, so it is called "**Transitive dependency**".
Parcel uses babel & other dependencies.



->Our project has one package.json & package-lock.json.
But each dependency has its own package.json

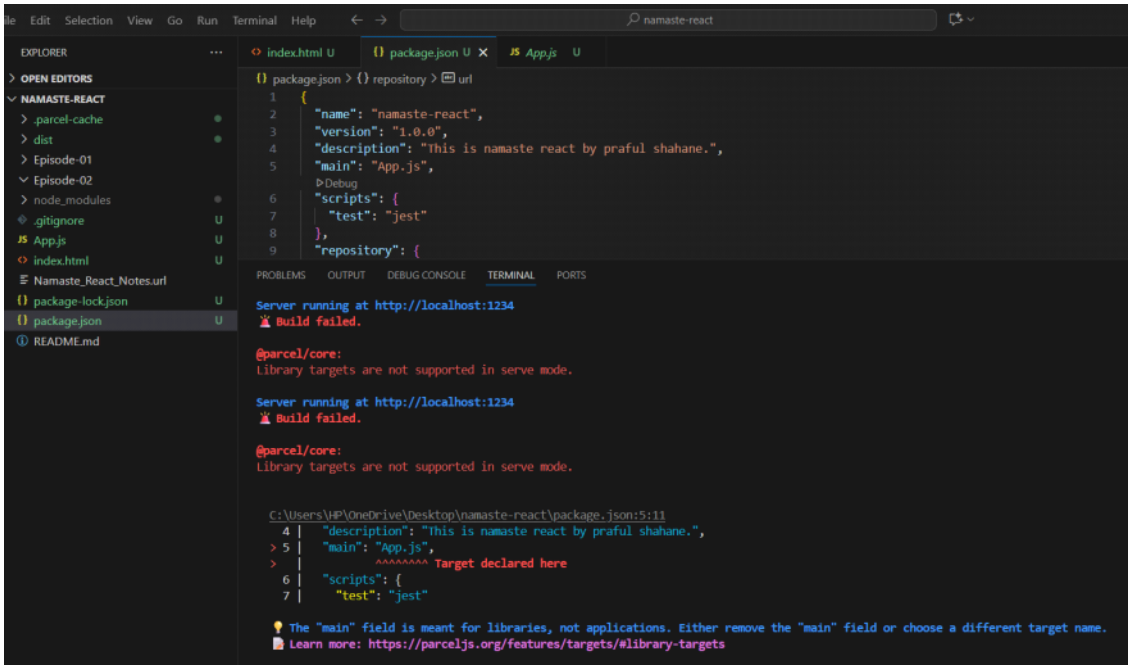
Q.)Shall we put whole Node Modules code on GitHub/production ?
=>NO.
If we want to some file not go to production, we put it in .gitignore file.



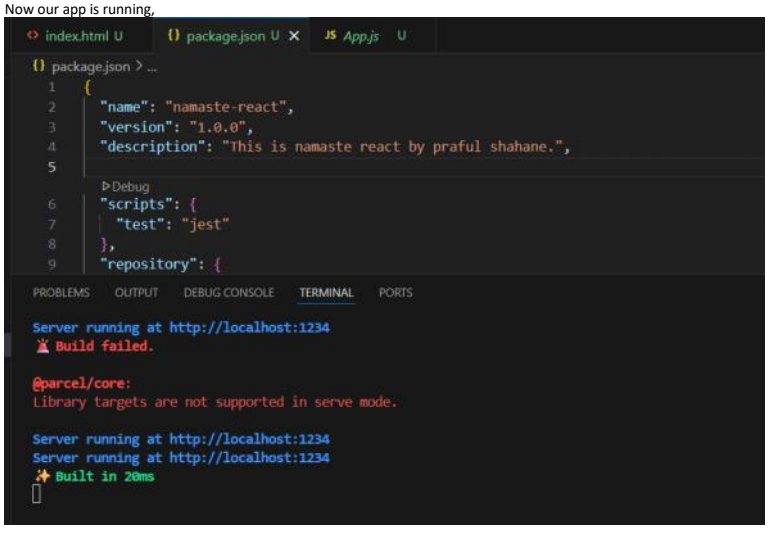
Q.) Shall we put package.json & package-lock.json on to git ? Why ?
=>Yes. Bcz package.json maintain a note of what all dependencies our project needs.
& package-lock.json maintain exact version of all dependencies.

NOTE:: If we have package.json & package-lock.json , we can regenerate node_modules just by typing command "**npm install**".

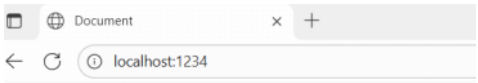
Episode 02 Part 02
==========
-> now run command, npx parcel index.html
If we get error while running app like then remove this line from "main":"App.js" and again run the app it will be running on server.



Now our app is running,



Url is ::=> Document



Hello World!

Command => npx parcel index.html
npx means executing the package.
Executing parcel package using source file index.html
Npm means install

->Earlier to do react coding, we add CDN link in index.html but it is not good way.
Why ?
Bcz fetching from CDN link is costly operation. If already have react in my node_modules then it will be very easy for us to use.
We don't to make another network call to react if we have already node_mdoules init.

```
<script crossorigin src="https://unpkg.com/react@18/umd/react.development.js"></script>
<script crossorigin src="https://unpkg.com/react-dom@18/umd/react-dom.development.js"></script>
```

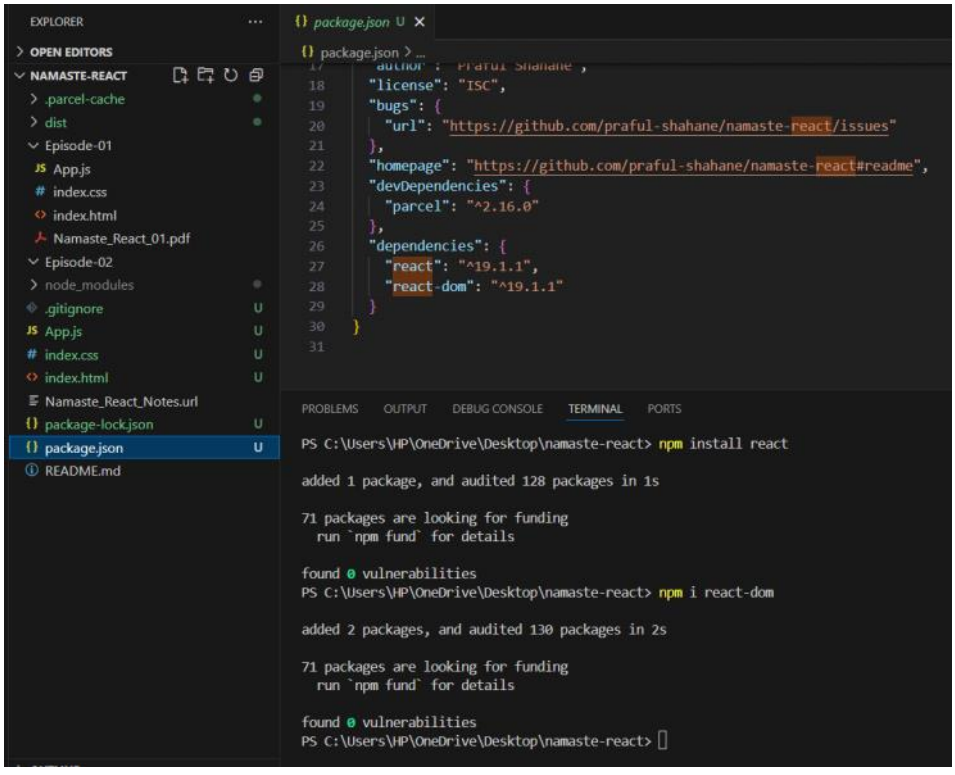
This is react 18, but tommorow if react 19 came, we need to keep changing the url so, instead of that we can change the version from package.json file.

->we have to install react & react-dom as normal dependency.

Command to install react =>
npm install react **or** npm i react

Where i is install

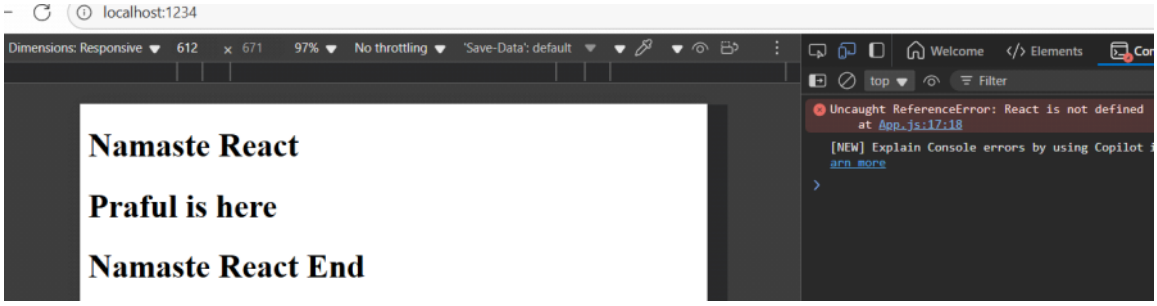
Command to install react-DOM =>
npm install react-dom **or** npm i react-dom



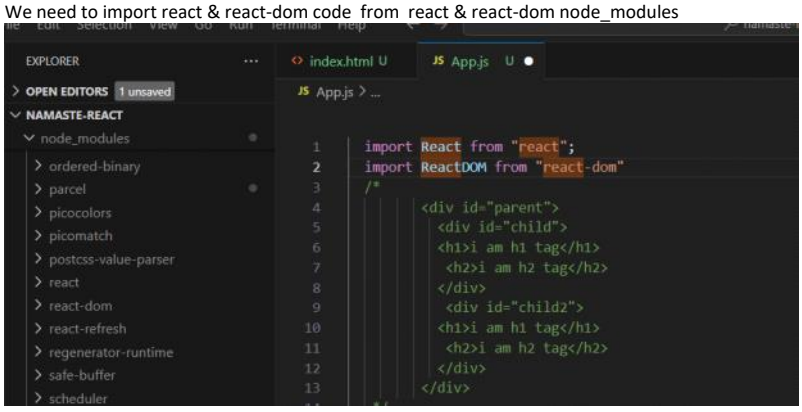
So, now we nolonger need the CDN just remove that 2 link.

```
<script crossorigin src="https://unpkg.com/react@18/umd/react.development.js"></script>
<script crossorigin src="https://unpkg.com/react-dom@18/umd/react-dom.development.js"></script>
```

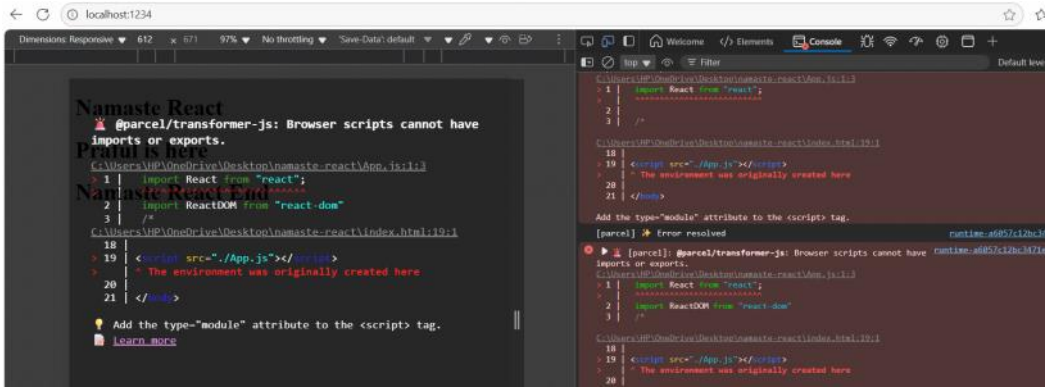
Now run our app.
Will our react code work ?
=>No, we get error in console like React is not defined.



We need to import react & react-dom code from react & react-dom node_modules



But still we are getting error like



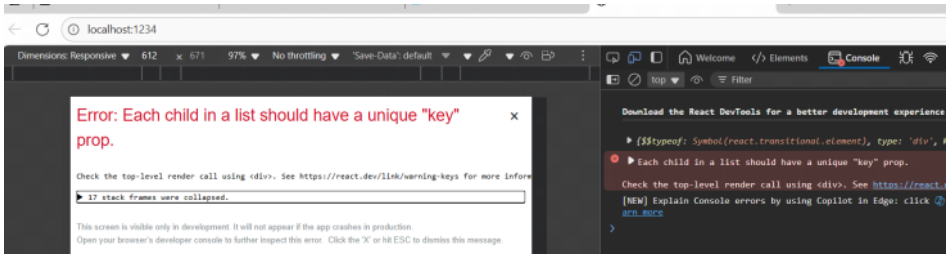
=>as we are using react, but we use script tag bydeault it thought it is for js file.
To specify we write react code in js, we use type="module" in it.

```
<script type="module"  src="./App.js"></script>
```

=>From React 19, we need to import createRoot() from

```
import ReactDOM from "react-dom/client";
```

=>Now we can see our react code in web page. & if we make changes in vs-code,after saving it automatically reflecting on web page.
This is done by parcel.



Parcel
1)doing dev build for us.
2)create local server for us.
3)Automatically refreshing page i.e. HMR(Hot Module Replacment)

Parcel uses File watching algorithm written in c++ for doing above operations.
Also parcel using caching for faster builds.
->Parcel does image optimization.
->parcel minification of files also.
->Parcel bundling our code.

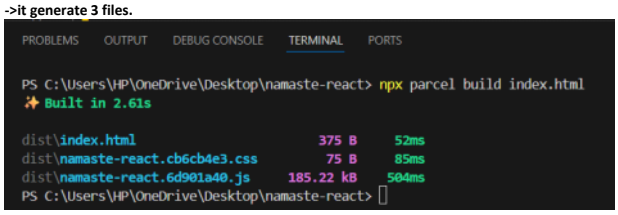
->Parcel Compressing our code.
->parcel uses consistent Hashing
->it does code splitting
->it does differential bundling so that it our app work in older browser as well.
->Parcel gives us Diagnostic gives us beautiful error.
->parcel gives us error handling.
->parcel help us to work with https us also.
->Parcel does tree shaking (remove unused code)
->it create a build diff build for dev & prod bundles

Read about parceljs.org website.

So, parcel also help to make our app fast.
Internally parcel uses other library also to make our fast.

->when we write command to build for production
npx parcel build index.html

->it generate 3 files.



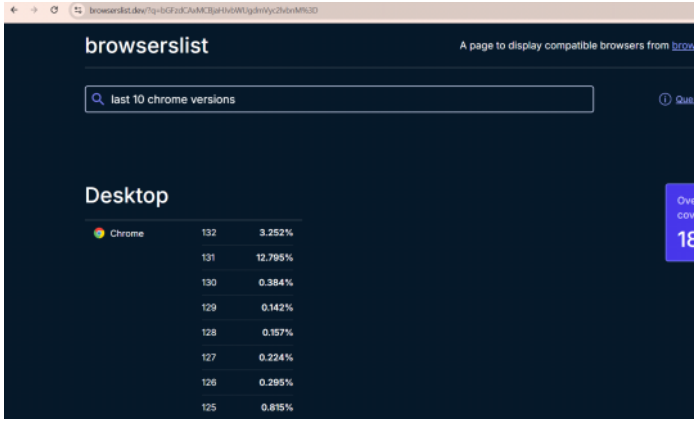
It will bundle all file and put it in **dist** folder.

The web page we see in our browser it is generated from **dist** folder.

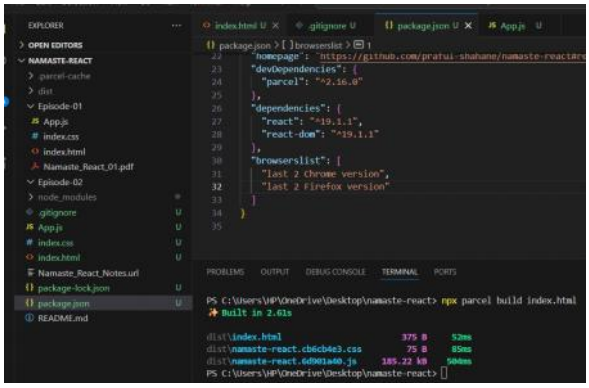
When we write **Command** => npx parcel index.html
It create a development build for our project & it host it on 1234.
When it generate a development build it puts it up into **dist** folder.
The web page we see in our browser it is generated from **dist** folder.
When you refrsh page when you save files it was using parcel-cache and dist to update it using HMR.

->parcel-cache, dist are temporary file just like node_modules it can generate automatically.
So we can add this inside .gitignore file.

=>Lets make out app compatible with older vserions as well.
We will use browserList in node_modules.
Go to website
https://browserslist.dev/?q=bGFzdCAyIHZlcnNpb25z



->add browserlist in package.json



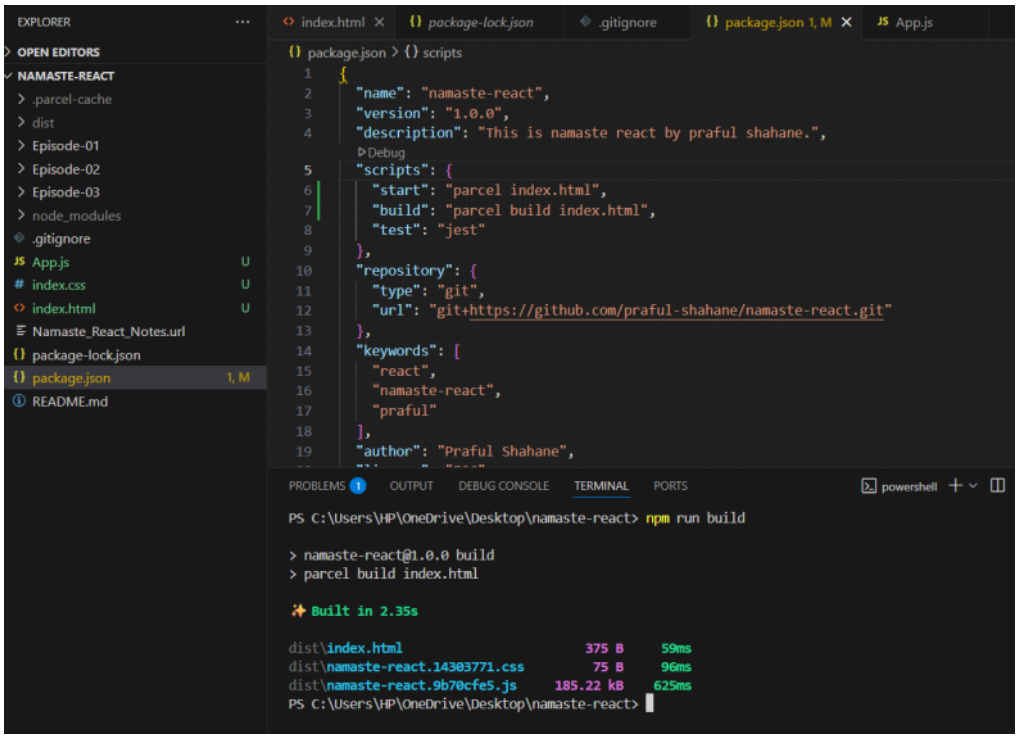
Q)What make your app fast ?
=>Not just React, parcel & its dependencies make its app fast.

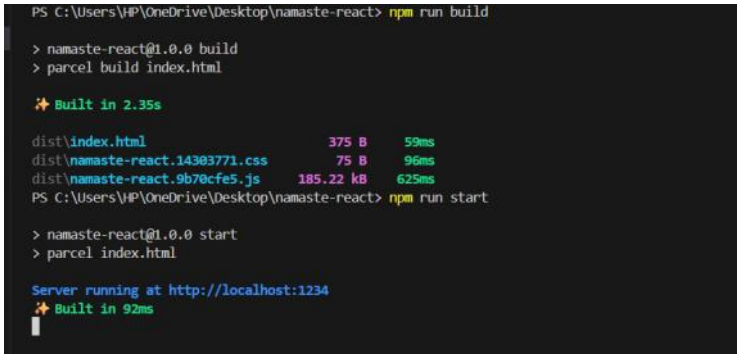
++
EPISODE 3: Laying the Foundation
++

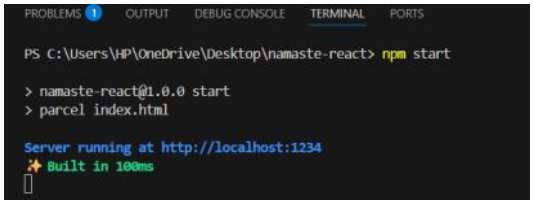
Episode 03 Part 01
==========
->
npx parcel index.html
Above command used to run our application.
So, writing again & again same command to build project, instead of that we can write one script to build our project.

It is an NPM script, we need to create under package.json file.
Go to scripts key, here we can specify our command.
Ex:-

```
  "scripts": {
    "start": "parcel index.html",  //to build our development code.
    "build": "parcel build index.html", //  //to build our production code.
    "test": "jest"
  }
```





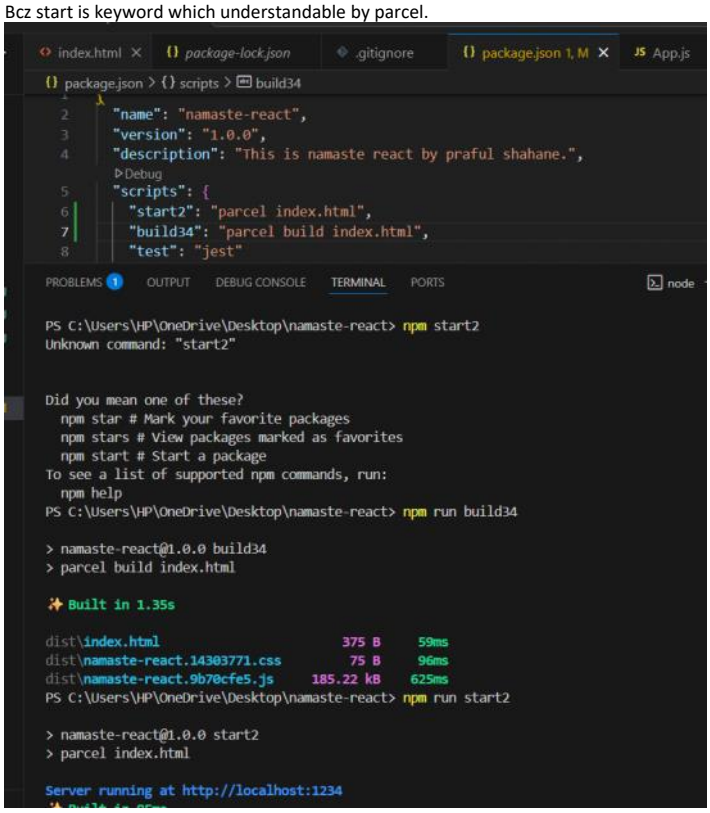


-> instead of **npm run start** we can write **npm start** also. But **npm build** won't work.
It only for start otherwise we get error.
-> instead of **npm run start** we can write **npm start**. Behind the scenes it calling **parcel index.html**
-> we can specify other name of key also.
For start, we can write start2. for build, we can write build34 also.

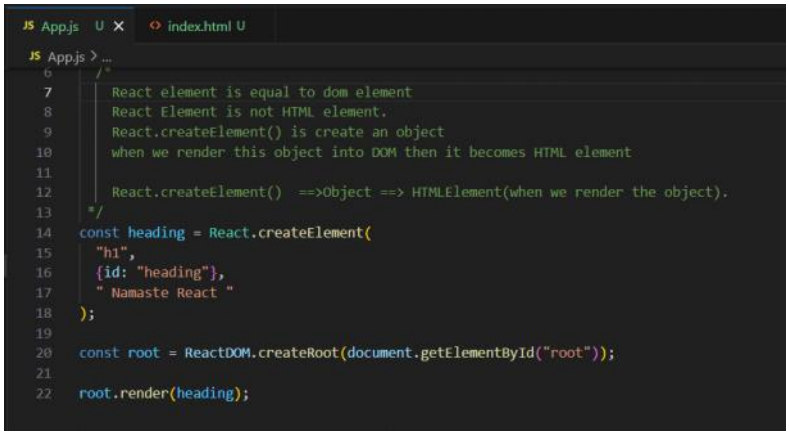
```
  "scripts": {
    "start": "parcel index.html",  //to start our project/App in  development code.
    "build": "parcel build index.html", //  to build our production code.
    "test": "jest"
```

```
}
```

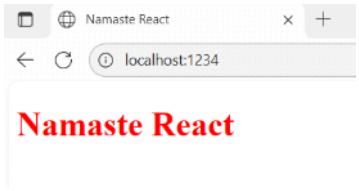
But if we use shortcut like **npm start**, it won't work here. Like **npm start2**, we get error as shown below.
Bcz start is keyword which understandable by parcel.



Episode 03 Part 02
==========
->we have just basic code using react to display "Namaste React " on web browser.



On browser o/p=>



Episode 03 Part 03
==========

```
const heading = React.createElement(
  "h1",
  {id: "heading"},
  " Namaste React "
);
```

But create a React Element using createElement() is not good way.
Bcz here we write HTML code inside it & makes out code clumsy.

So, faceBook developer create **JSX** to create ReactElement.
->React & JSX is different.
->we can write React code without JSX also.

JSX
->Before framework & libraries comes, we write all code in js files & HTML skelaton.
->to write both HTML & js code at one place we have libraries & framework.
Ex:-

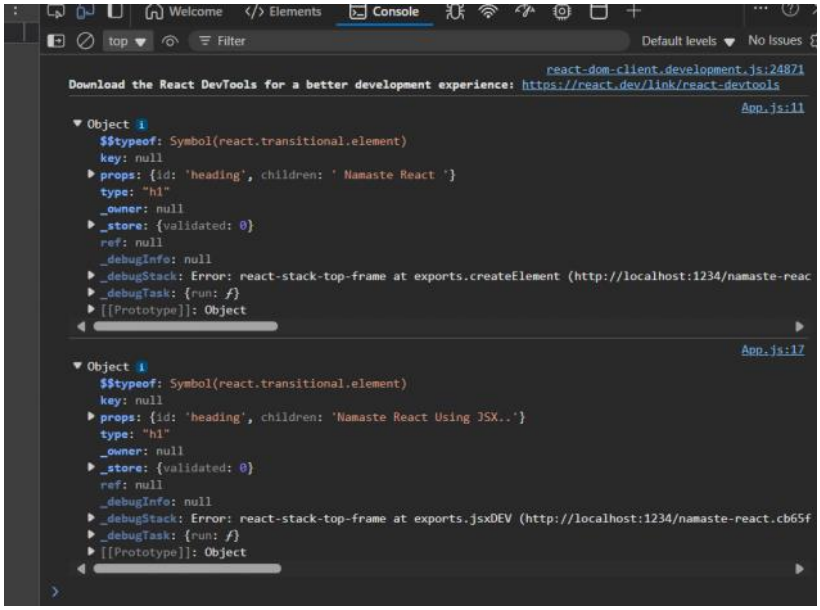
```
//JSX
const jsxHeading =<h1>Namaste React Using JSX..</h1>
```

```
9
0   //JSX
1   const jsxHeading =<h1>Namaste React Using JSX..</h1>
2
```

-> JSX is not HTML inside the javascript. JSX is HTML-like or XML-like syntax.

```
4
5    //Created React Element Using Core React
6   const heading = React.createElement(
7     "h1",
8     {id: "heading"},
9     " Namaste React "
0   );
1   console.log(heading);
2
3
4   //Created React Element Using JSX
5   const jsxHeading =<h1 id="heading">Namaste React Using JSX..</h1>
6
7   console.log(jsxHeading);
8
```

->Output of console is



->both object are same as we see in console as wee see in above pic.

=>

```
//Created React Element Using JSX
const jsxHeading =<h1 id="heading">Namaste React Using JSX..</h1>

console.log(jsxHeading);


const root = ReactDOM.createRoot(document.getElementById("root"));

root.render(jsxHeading);
```

o/p on browser

-> so from now we will create ReactElement using JSX.

->any piece of js code that js engine understand it.
Can js engine understand JSX code ? Why ?
=>No. bcz js engine understand EchmaScript(ES6) i.e.pure js.

```
const jsxHeading =<h1 id="heading">Namaste React Using JSX..</h1>
```

this is not valid pure javascript code.
Then why our JSX code is running inside the js ?
=> because parcel is doing job behind the scenes.

We write JSX code, this JSX code is transpiled(convert to other code that browser can understand) before it reaches the JS Engine.
Transpiling is done by PARCEL. Only PARCEL can't do this, it done by other help like BEBEL.
BEBEL is javascript compiler.

JSX(code is transpiled before it reaches the JS Engine)----PARCEL---BABEL.

->When we use React.createElement(), ReactElement which is an JS object is created.
Then using render(), we render as HTMLElement.

->When we use JSX code, it converted into React.createElement behind the scene.
ReactElement which is an JS object is created. Then using render(), we render as HTMLElement.

```
JSX => Babel transpiles it to  React.createElement => ReactElement(JS Object) => HTMLElement(render using render())
```

```
4
5    //Created React Element Using Core React
6    //React.createElement  ==> ReactElement(it is JS Object) ===>HTMLElement(render using render())
7    const heading = React.createElement(
8    "h1",
9    {id: "heading"},
10   " Namaste React "
11 );
12 console.log(heading);
13
14
15 //Created React Element Using JSX
16 //JSX (this JSX code is transpiled before it reaches the JS) - it is done by Pracel. Parcel uses Babel to convert JSX to React.createElement.
17 //JSX =>  React.createElement => ReactElement(JS Object) => HTMLElement(render using render())
18 const jsxHeading =<h1 id="heading">Namaste React Using JSX..</h1>
19
20 console.log(jsxHeading);
```

Q) How Babel convert JSX into ReactElement ?

-> Go to **bebeljs website**,
See how babel convert our code into React.createElement() behind the scene.

->Old browser not understand ES6 code, so babel convert it into old browser compatible js code.

In HTML, we give class as class but in JSX we give class as className.
To give attributes in JSX we use camelCase.

//HTML

```
<body>
  <div id="root" class="root">
    <h1 class="head" tabindex="1">Namaste React Using JSX..</h1>
  </div>
```

//JSX

```
const jsxHeading =<h1 className="head" tabIndex="1">Namaste React Using JSX..</h1>
```

q)how to write image tag in jsx ?
q)how to write img src in jsx & anchor tag in jsx ?

->If JSX is written is single line, then it is valid JSX.
Ex:-

```
const jsxHeading =<h1 className="head" tabIndex="1">Namaste React Using JSX..</h1>
```

->But if we write JSX in multiple line, it must be written inside round bracket ().
Ex:-

```
const jsxHeading =
    (<h1 className="head" tabIndex="1">
    Namaste React Using JSX..
    </h1>);
```

Extension installed in our VSCode ->

1) prettier Code formatter for beautfiy our code
2) Bracket Pair Colorization Toggle.
3) ES lint
4)Better Comments

Episode 03 Part 04
==========

React Component

->Everything in react is component.
There are two types of component in react
1)Class based components --(old way of Writing)(it uses javascript classes)
2)Functional Components --(New Way of Writing)(it uses js functions)

React Functional Component is an function that return a some piece of JSX code.
As we Know, JSX is an React Element.
Or we can say that React Functional Component is just javascript function which returns React Element.
always create an React component with first letter in capital.

```
//React Functional Component
//it is just javascript function which returns some piece of JSX React Element
//always create an React component with first letter in capital
const HeadingCompoenent = ()=>{
  return <h1>Namaste React Functional Component</h1>
};
```

In pure javascript, if our function statement is only one then return & {} bracket are optional.
Ex:-

```
const fn =()=>{return true;}
//or
const fn2 =()=> true;
```

So, in React above code can be written as,

```
//React Functional Component
//it is just javascript function which returns some piece of JSX React Element
//always create an React component with first letter in capital
const HeadingCompoenent = ()=><h1>Namaste React Functional Component</h1>;
```

->we can write React Functional Component in following ways.

```
//way 1 (with return and curly braces)
const HeadingCompoenent1 = ()=>{
  return <h1>Namaste React Functional Component</h1>
};
//Way 2 (if we have only one line of code then we can skip the return and curly braces)
const HeadingCompoenent2 = ()=><h1>Namaste React Functional Component</h1>;

//Way 3 (if our JSX has multiple lines then we have to wrap it in parenthesis)
const HeadingCompoenent3 = ()=> (
                        <h1 className="heading">
                Namaste React Functional Component
                 </h1>);
```

```
//way 1 (with return and curly braces)
const HeadingCompoenent1 = ()=>{
  return <h1>Namaste React Functional Component</h1>
};

//Way 2 (if we have only one line of code then we can skip the return and curly braces)
const HeadingCompoenent2 = ()=><h1>Namaste React Functional Component</h1>;

//Way 3 (if our JSX has multiple lines then we have to wrap it in parenthesis)
const HeadingCompoenent3 = ()=> (
                        <h1 className="heading">
                Namaste React Functional Component
                 </h1>);
```

Normal React Element & React Functional Component

```
// Normal React Element
const heading =
      (<h1 className="head" tabIndex="1">
      Namaste React Using JSX..
      </h1>);

//React Functional Component
const HeadingCompoenent1 = ()=>{
  return <h1>Namaste React Functional Component</h1>
};
```

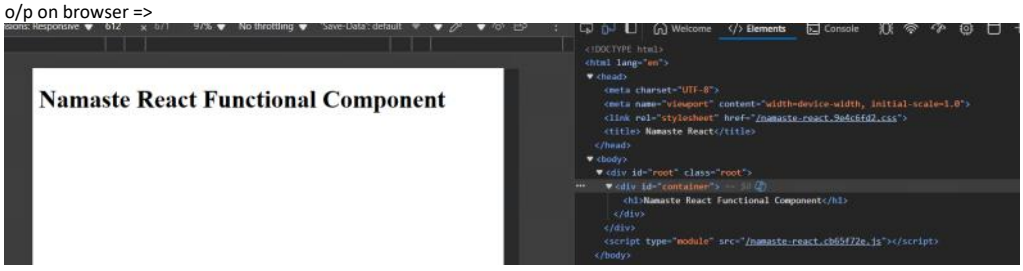
->React Functional Component can return nested JSX / Nested React Element also.

```
//React Functional Component
const HeadingCompoenent1 = ()=>{
  return <div id="container">
  <h1>Namaste React Functional Component</h1>
</div>;
};
```

->To render React Element, we just pass name of reactElement into render()
But to render React Functional Component, we need to pass **<ReactFuctionalComponentName />** into render()
Ex:-

```
//React Functional Component
const HeadingCompoenent = ()=>{
  return <div id="container">
  <h1>Namaste React Functional Component</h1>
</div>;
};

const root = ReactDOM.createRoot(document.getElementById("root"));

root.render(<HeadingCompoenent />);
```

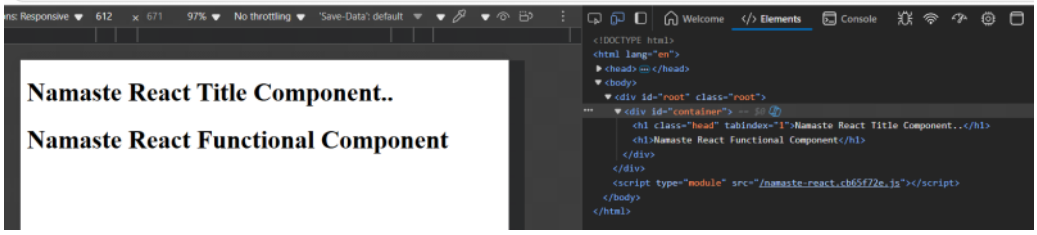
o/p on browser =>



->I want to inject one functional Component into another component.
Ex:-

```jsx
import React from "react";
import ReactDOM from "react-dom/client";

//React Functional Component
const TitleComponent = ()=>
      (<h1 className="head" tabIndex="1">
      Namaste React Title Component..
      </h1>);

//React Functional Component
const HeadingCompoenent = ()=>{
  return <div id="container">
    <TitleComponent />
   <h1>Namaste React Functional Component</h1>
</div>;
};


const root = ReactDOM.createRoot(document.getElementById("root"));

root.render(<HeadingCompoenent />);
```

o/p on browser =>

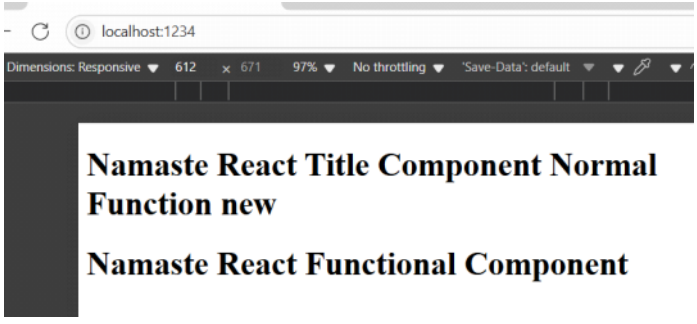


Q.) What is Component Composition ?
Ans :=> one functional Component into another component.

Episode 03 Part 05
==========

Q.)can we write React Functional Component using Normal Function ?
=>Yes.

```jsx
//React Functional Component
const TitleComponent = function(){
    return   (<h1 className="head" tabIndex="1">
     Namaste React Title Component Normal Function new
     </h1>);
}

//React Functional Component
const HeadingCompoenent = ()=>{
  return <div id="container">
    <TitleComponent />
   <h1>Namaste React Functional Component</h1>
</div>;
};


const root = ReactDOM.createRoot(document.getElementById("root"));

root.render(<HeadingCompoenent />);
```

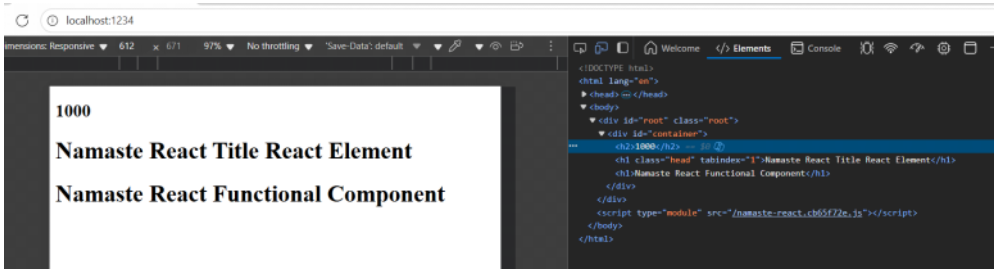


->We preferred to use arrow function to write React Function Component.

->1)Put ReactElement inside the component.
->at the end ReactElement is also an object, so we write inside the {}.it will treat like as Js object.

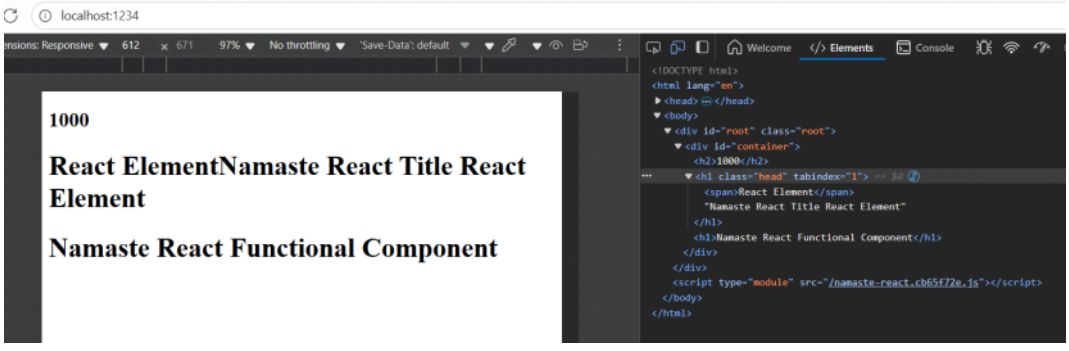
```jsx
import React from "react";
import ReactDOM from "react-dom/client";

//React Element
const title =  (<h1 className="head" tabIndex="1">
      Namaste React Title Component Normal Function new
      </h1>);

      const number = 1000;

//React Functional Component
const HeadingCompoenent = ()=>{
  return (<div id="container">

      <h2> {
        //write any JavaScript expression inside {}
      number}
      </h2>
      {title}

   <h1>Namaste React Functional Component</h1>
</div>);
};

const root = ReactDOM.createRoot(document.getElementById("root"));
root.render(<HeadingCompoenent />);
```

o/p>



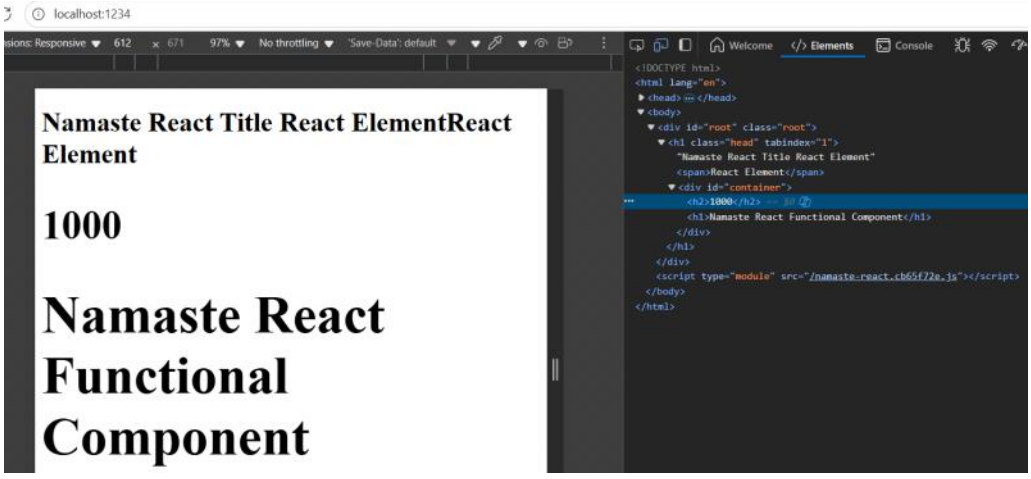
->2)Put ReactElement inside the ReactElement.

```jsx
import React from "react";
import ReactDOM from "react-dom/client";

//React Element
const elem= <span>React Element</span>;

const title =  (<h1 className="head" tabIndex="1">
        {elem}
      Namaste React Title React Element
      </h1>);
    const number = 1000;

//React Functional Component
const HeadingCompoenent = ()=>{
  return (<div id="container">

      <h2> {
        //write any JavaScript expression inside {}
      number}
      </h2>
      {title}

   <h1>Namaste React Functional Component</h1>
</div>);
};

const root = ReactDOM.createRoot(document.getElementById("root"));
root.render(<HeadingCompoenent />);
```



3)put reactCompoent into ReactElement

```jsx
import React from "react";
import ReactDOM from "react-dom/client";

//React Element
const elem= <span>React Element</span>;

//React Functional Component
const HeadingCompoenent = ()=>{
  return (<div id="container">

      <h2> {
        //write any JavaScript expression inside {}
      number}
      </h2>

    <h1>Namaste React Functional Component</h1>
</div>);
};

const title =  (<h1 className="head" tabIndex="1">
    Namaste React Title React Element
        {elem}

        <HeadingCompoenent />
      </h1>

    );

    const number = 10000;
const root = ReactDOM.createRoot(document.getElementById("root"));
root.render(title);
```



->let say we are calling any api to get some data,if that api has some malicious data, & if we are using that data our JSX will sanitize that data before rendering on browser.

```jsx
    const data = api.getData();
    //api call if this call contain malicious code then JSX sanitizer will remove that code before rendering on the browser

    //React Functional Component
const HeadingCompoenent = ()=>{
  return (<div id="container">

      <h2> {
        //write any JavaScript expression inside {}
      data}
      </h2>

    <h1>Namaste React Functional Component</h1>
</div>);
};

const root = ReactDOM.createRoot(document.getElementById("root"));
root.render(<HeadingCompoenent />);
```

=>
<Title /> or
<Title > </Title> or {Title()} all are same. We can write anything.

```jsx
import React from "react";
import ReactDOM from "react-dom/client";

//Title Component
const Title = ()=>  (<h1 className="head" tabIndex="1">
  Namaste React Title React Element
    </h1>

  );
    //React Functional Component
const HeadingCompoenent = ()=>{
  return (<div id="container">
    <Title />
   <Title > </Title>
   <h1>Namaste React Functional Component</h1>
</div>);
};

const root = ReactDOM.createRoot(document.getElementById("root"));
root.render(<HeadingCompoenent />);
```

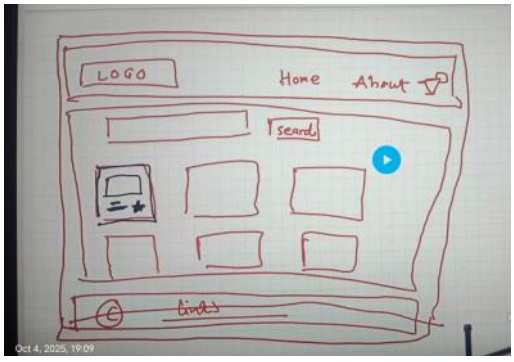
->we can call Title React Functional Component inside {} braces also.
We can call by similar way we call javascript function.

```jsx
//Title Component
const Title = ()=>  (<h1 className="head" tabIndex="1">
  Namaste React Title React Element
    </h1>

  );
    //React Functional Component
const HeadingCompoenent = ()=>{
  return (<div id="container">
    {Title()}
   <h1>Namaste React Functional Component</h1>
</div>);
};

const root = ReactDOM.createRoot(document.getElementById("root"));
root.render(<HeadingCompoenent />);
```

++
EPISODE 4: Talk me Cheap Show me the code
++

Episode 04 Part 01
==========

Food Ordering App
->first we do planning.
Planning
1)what are we going to built ?
->we build wireframe first.



->Now do low-level plannning, how my app look like.
Q)What component can my app have ?
=>

```
  Low Level Planning of the App
  ==============================
Header Component
   - Logo Component
   - Nav Items Component
Body Component
 - Search Component
 -Restro Container Component
   - Restro Card Component
Footer Component
  -Copyright Component
  -Links Component
  -Address Component
  -Contact Component
```

```
/*
Low Level Planning of the App
=============================
Header Component
   - Logo Component
   - Nav Items Component
Body Component
 - Search Component
 -Restro Container Component
   - Restro Card Component
    -image
    -name
    -star rating
    -cuisine
    -time for delivery
Footer Component
  -Copyright Component
  -Links Component
  -Address Component
  -Contact Component

*/
```

->Now we slowly start our development of app.

->We can write CSS by creating separate css file.
Or we can write css in JS file also like,

```
//Write CSS in JS Using JSX inline styling
  const styleCard={
    backgroundColor:"#f0f0f0",
  };

  const RestaturantCardComponent =()=>{
    return(
      <div className="res-card" style={styleCard}>
            <h3>Meghana Foods</h3>

      </div>
    );

  };
```

Also we can write it as below,

```
style={{ backgroundColor:"#f0f0f0"}}
```

First {} is telling some piece of Js code inside it. 2nd {} is our js object.

```
  const RestaturantCardComponent =()=>{
    return(
      <div className="res-card" style={{ backgroundColor:"#f0f0f0"}}>
            <h3>Meghana Foods</h3>
      </div>
    );

  };
```

->we can write CSS using other framework as well like tailwindCSS.

```
App.js > HeaderComponent
  const HeaderComponent =()=>{
    return (
      <div className="header">
        <div classNname="logo-container">
          <img classNname="logo" src="https://www.logodesign.net/logo/smoking-burger-with-lettuce-3624ld.png" />
        </div>
        <div classNname="nav-items">
          <ul>
            <li>Home</li>
            <li>About Us</li>
            <li>Contact Us</li>
            <li>Cart</li>
          </ul>
        </div>
      </div>
    );
  };

  const RestaturantCardComponent =()=>{
    return(
      <div classNname="res-card" style={{ backgroundColor:"#f0f0f0"}}>
      <img  classNname="res-logo"
      alt="res-logo"
      src="https://media-assets.swiggy.com/swiggy/image/upload/fl_lossy,f_auto,q_auto,w_264,h_288,c_fill/e0vvulfbahjxjz6k4uwi"/>
      <h3>Meghana Foods</h3>
      <h4>South Indian, North Indian</h4>
      <h4>4.5 Star</h4>
      <h4>30 mins</h4>
      </div>
    );
  };
```

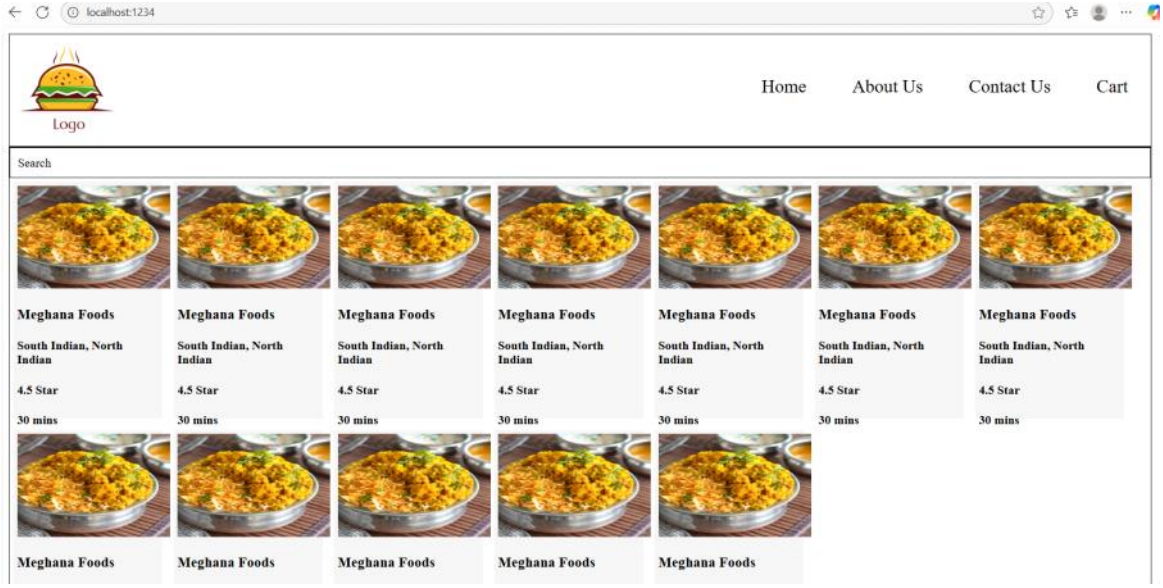
```jsx
App.js > HeaderComponent
59
60  const BodyComponent =()=>{
61    return (
62      <div className="body">
63       <div className="search">Search </div>
64       <div className="res-container">
65         <RestaturantCardComponent />
66          <RestaturantCardComponent />
67           <RestaturantCardComponent />
68                    <RestaturantCardComponent />
69          <RestaturantCardComponent />
70           <RestaturantCardComponent />
71                    <RestaturantCardComponent />
72          <RestaturantCardComponent />
73           <RestaturantCardComponent />
74                    <RestaturantCardComponent />
75          <RestaturantCardComponent />
76           <RestaturantCardComponent />
77       </div>
78       </div>
79     );
80   };
81  const AppLayOut =()=>{
82    return (
83    <div className="app">
84      <HeaderComponent />
85       <BodyComponent />
86    </div>);
87  };
88
89
90  const root = ReactDOM.createRoot(document.getElementById("root"));
91  root.render(<AppLayOut />);
```

```css
# index.css > .res-container
1
2   #heading {
3       color: red;
4   }
5   .header{
6       display: flex;
7       justify-content: space-between;
8       border: 1px solid black;
9   }
10  .logo{
11      width: 150px;
12  }
13
14  .nav-items > ul {
15      font-size: 24px ;
16      display: flex;
17      list-style-type: none;
18  }
19
20  .nav-items > ul  > li{
21      padding: 15px;
22      margin: 15px;
23  }
24
25  .body{
26       border: 1px solid black;
27  }
28  .res-card{
29      width: 185px ;
30      height: 300px;
31     margin: 5px;
32      padding: 5px;
33  }
```

```css
.res-card{
    width: 185px ;
    height: 300px;
   margin: 5px;
    padding: 5px;
}

.res-card:hover{
    cursor: pointer;
     border: 1px solid black;
}

.search{
    padding: 10px;
     border: 1px solid black;
}

.res-logo{
    width: 200px;
    height: 135px;
}
.res-container{
    display: flex;
    flex-wrap: wrap;
    gap: 5px;
}
```

```jsx
34
35  /*
36  after hover on the card border will be black
37  & cursor will be pointer
38  */
39  .res-card:hover{
40      cursor: pointer;
41       border: 1px solid black;
42  }
43
```

Web page on browser.



->but here we are hardcoding the restaurant card.
How we pass diff value in each card. See part2.

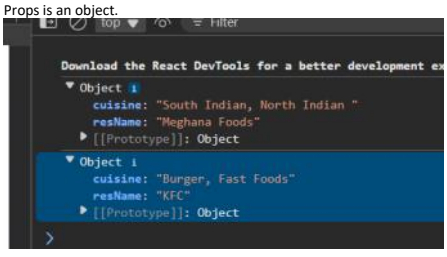
Episode 04 Part 02
==========
Props
->to write multiple restaurant info using same RestarurantComponentCard, we use Props.
->to pass data dynamically we use props.
->props are just arguments to an functions.

```
<RestaturantCardComponent  resName="Meghana Foods" cuisine="South Indian, North Indian "/>
 <RestaturantCardComponent  resName="KFC" cuisine="Burger, Fast Foods"/>
```

In this way we pass props to React Component.
We pass that props in to React Component, our ReactComponent taking props as argument.

```
const RestaturantCardComponent =(props)=>{
  console.log(props);

  return(
    <div className="res-card" style={{ backgroundColor:"#aca51fff"}}>
    <img  className="res-logo"
    alt="res-logo"
    src="https://media-assets.swiggy.com/swiggy/image/upload/fl_lossy,f_auto,q_auto,w_264,h_288,c_fill/e0vvulfbahjxjz6k4uwi"/>
    <h3>Meghana Foods</h3>
    <h4>South Indian, North Indian </h4>
    <h4>4.5 Star</h4>
    <h4>30 mins</h4>
    </div>
  );
};
```

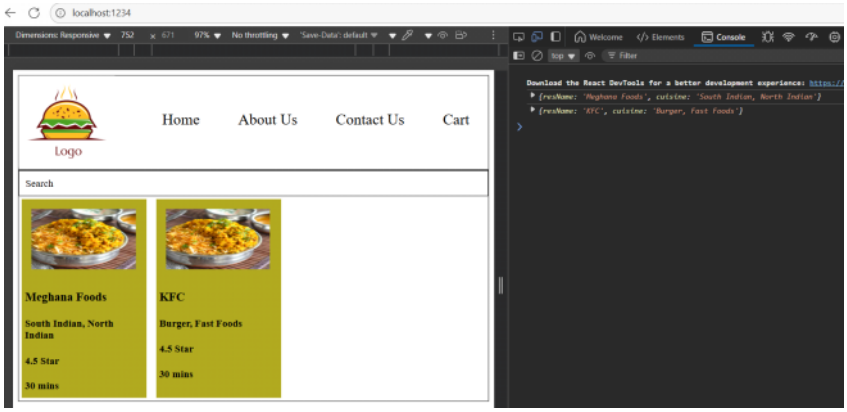
If we check console logs in web page,
We see our data is getting inside the ReactComponent.
So,React wrap everthing into props & pass to its Component.
Props is an object.




```
const BodyComponent =()=>{
  return (
    <div className="body">
     <div className="search">Search </div>
     <div className="res-container">
       <RestaturantCardComponent
       resName="Meghana Foods"
       cuisine="South Indian, North Indian"
       />
        <RestaturantCardComponent
        resName="KFC"
        cuisine="Burger, Fast Foods"
        />
    </div>
    </div>
  );
};
```

```jsx
const RestaturantCardComponent =(props)=>{
  console.log(props);

  return(
    <div className="res-card" style={{ backgroundColor:"#aca51fff"}}>
    <img  className="res-logo"
    alt="res-logo"
    src="https://media-assets.swiggy.com/swiggy/image/upload/fl_lossy,f_auto,q_auto,w_264,h_288,c_fill/e0vvulfbahjxjz6k4uwi"/
    <h3>{props.resName}</h3>
    <h4>{props.cuisine}</h4>
    <h4>4.5 Star</h4>
    <h4>30 mins</h4>
    </div>
  );
};
```

On, web page our o/p is



->Some React Developer can write code in this way also by destructing the object.
Way=>1

```jsx
const RestaturantCardComponent =({resName,cuisine})=>{

  return(
    <div className="res-card" style={{ backgroundColor:"#aca51
    <img  className="res-logo"
    alt="res-logo"
    src="https://media-assets.swiggy.com/swiggy/image/upload/fl_
    <h3>{resName}</h3>
    <h4>{cuisine}</h4>
    <h4>4.5 Star</h4>
    <h4>30 mins</h4>
    </div>
  );
};
```

Way=>2

```jsx
const RestaturantCardComponent =(props)=>{
  console.log(props);

  return(
    <div className="res-card" style={{ backgroundColor:"#aca51fff"}}>
    <img  className="res-logo"
    alt="res-logo"
    src="https://media-assets.swiggy.com/swiggy/image/upload/fl_lossy,f_auto,q_auto,w_264,h_288,c_fill/e0vvulfbahjxjz6k4uwi"/>
    <h3>Meghana Foods</h3>
    <h4>South Indian, North Indian </h4>
    <h4>4.5 Star</h4>
    <h4>30 mins</h4>
    </div>
  );
};
```

Way=>3

```jsx
const RestaturantCardComponent =(props)=>{
  const {resName,cuisine} =props;
  return(
    <div className="res-card" style={{ backgroundColor:"#aca51fff"
    <img  className="res-logo"
    alt="res-logo"
    src="https://media-assets.swiggy.com/swiggy/image/upload/fl_loss
    <h3>{resName}</h3>
    <h4>{cuisine}</h4>
    <h4>4.5 Star</h4>
    <h4>30 mins</h4>
    </div>
  );
};
```

Note::
Destructuring of our Object.

```jsx
const {resName,cuisine} =props;
```

=>Install **JSON Viewer** plugin in browser to see good JSON.

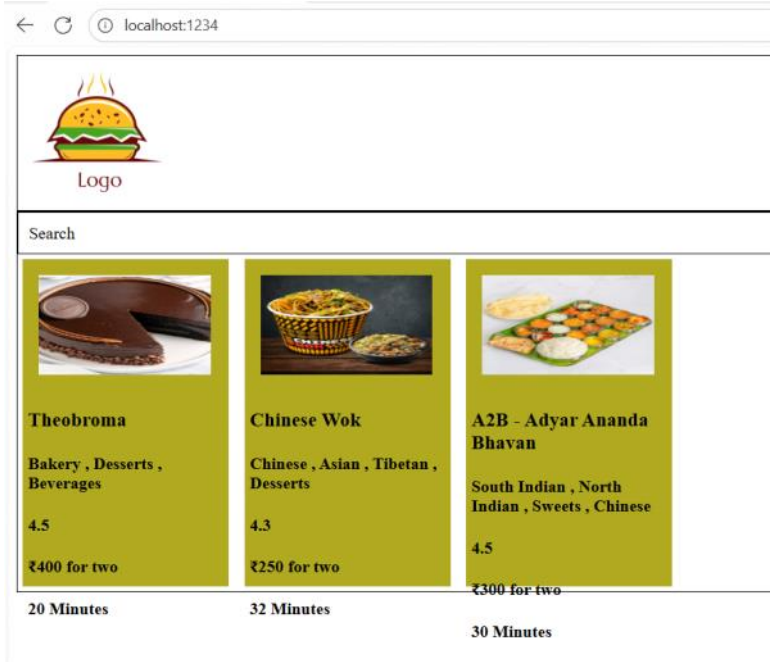
->ConfigDriven UI

It means our UI is driven by config. Config means our data.
Ex:-
Let say we have swiggy app.
If user is from Banglore, he can see restaurant from banglore & some offers on it.
But if user from Mumbai, he can see restaurant from Mumbai not from banglore.
This is ConfigDrivenUI.

->We get a real time data from swiggy api for list of restaurant & we used in our project.

```
const RestaturantCardComponent =(props)=>{
  const {resData} =props;
  console.log(resData);

  return(
    <div className="res-card" style={{ backgroundColor:"#aca51fff"}}>
    <img  className="res-logo"
    alt="res-logo"
    src={
    "https://media-assets.swiggy.com/swiggy/image/upload/fl_lossy,f_auto,q_auto,w_660/"
     +resData.info.cloudinaryImageId
    }/>
    <h3>{resData.info.name}</h3>
    <h4>{resData.info.cuisines.join(" , ")}</h4>
    <h4>{resData.info.avgRating}</h4>
    <h4>{resData.info.costForTwo}</h4>
    <h4>{resData.info.sla.deliveryTime} Minutes</h4>
    </div>
  );
};


//list of restaurants from swiggy api
const restaurantList = [ ···
]
const resObj={ ···
};
```

```
//list of restaurants from swiggy api
const restaurantList = [ ···
]
const resObj={ ···
};

const BodyComponent =()=>{
  return (
    <div className="body">
     <div className="search">Search </div>
     <div className="res-container">

      <RestaturantCardComponent
      resData={restaurantList[0]}
      />

      <RestaturantCardComponent
      resData={restaurantList[1]}
      />
              <RestaturantCardComponent
      resData={restaurantList[2]}
      />
    </div>
    </div>
  );
};
```



This below code not looks goods we modify our code. So we destructure our code.

```
const RestaturantCardComponent =(props)=>{
  const {resData} =props;
  console.log(resData);

  return(
    <div className="res-card" style={{ backgroundColor:"#aca51fff"}}>
    <img  className="res-logo"
    alt="res-logo"
    src={
    "https://media-assets.swiggy.com/swiggy/image/upload/fl_lossy,f_auto,q_auto,w_660/"
     +resData.info.cloudinaryImageId
    }/>
    <h3>{resData.info.name}</h3>
    <h4>{resData.info.cuisines.join(" , ")}</h4>
    <h4>{resData.info.avgRating}</h4>
    <h4>{resData.info.costForTwo}</h4>
    <h4>{resData.info.sla.deliveryTime} Minutes</h4>
    </div>
  );
};
```

Better way to write code.

```
const RestaturantCardComponent =(props)=>{
  const {resData} =props;
  console.log(resData);

  //Destructuring of our object
  // const cloudinaryImageId = resData.info.cloudinaryImageId;
     const {
      cloudinaryImageId,
      name,
      cuisines,
      avgRating,
      costForTwo
        } = resData?.info;
     // ? is for optional chaining


  return(
    <div className="res-card" style={{ backgroundColor:"#aca51fff"}}>
    <img  className="res-logo"
    alt="res-logo"
    src={
    "https://media-assets.swiggy.com/swiggy/image/upload/fl_lossy,f_auto,q_auto,w_660/"
     +cloudinaryImageId
    }/>
    <h3>{name}</h3>
    <h4>{cuisines.join(" , ")}</h4>
    <h4>{avgRating}</h4>
    <h4>{costForTwo}</h4>
    <h4>{resData.info.sla.deliveryTime} Minutes</h4>
    </div>
  );
};
```

=>
Our restaurantList is an array, we can not use direct index like this . It is bad practice.

```
  //list of restaurants from swiggy api
  const restaurantList = [ ···
]
  const resObj={ ···
};

  const BodyComponent =()=>{
    return (
      <div className="body">
       <div className="search">Search </div>
       <div className="res-container">

        <RestaturantCardComponent  resData={restaurantList[0]}/>
        <RestaturantCardComponent   resData={restaurantList[1]}/>
        <RestaturantCardComponent  resData={restaurantList[2]} />
      </div>
      </div>
    );
  };
```

So, we use map function for it.

If we are getting error like,

Keeping list items in order with `key`

Notice that all the sandboxes above show an error in the console:

Console

Warning: Each child in a list should have a unique “key” prop.

You need to give each array item a `key` — a string or a number that uniquely identifies it among other items in that array:

Add key also for unqiue purpose.
=>

```
//list of restaurants from swiggy api
const restaurantList = [...
]
const resObj={...
};

const BodyComponent =()=>{
  return (
    <div className="body">
      <div className="search">Search </div>
      <div className="res-container">
      {restaurantList.map((restaurant)=>
      ( <RestaturantCardComponent  key={restaurant.info.id} resData={restaurant} />
      ))}
      </div>
      </div>
  );
};
const AppLayOut =()=>{
  return (
  <div className="app">
    <HeaderComponent />
     <BodyComponent />
  </div>);
};
```

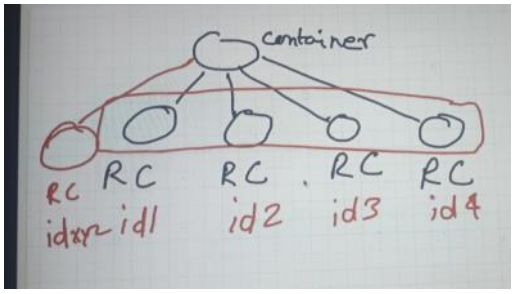
Important Note=>

```
<RestaturantCardComponent  key={restaurant.info.id} resData={restaurant} />
```

Key is used for Unique purpose Each child is unqiue in a list.
& restData here we pass & restData we use in RestaurantReactComponent implemenation must be same. Otherwise code will not work.

```
const {resData} =props;
```

QU) What is need of Key in React Component ?
=>
Let say we have parent div class res-container & itch child div class is res-card.
There are n no.of res-card.
If res-card has not contain any id,
& if we add any new res-card our all card will re-render again with new card.
But if we our res-card contain id & we use it, & if we add any new res-card our all card will not be re-render again with new card.
Only new-card will be re-render again. Other remain as it is.
It is optimization done by React.



=>We can use index instead of id for keys. But it not good practice.
Like. React does not recommeded it.

```
//list of restaurants from swiggy api
const restaurantList = [...
]
const resObj={...
};

const BodyComponent =()=>{
  return (
    <div className="body">
      <div className="search">Search </div>
      <div className="res-container">
      {restaurantList.map((restaurant,index)=>
      ( <RestaturantCardComponent  key={index} resData={restaurant} />
      ))}
      </div>
      </div>
  );
};
```

Bcz index as key as anti-pattern.

Note::
//Not using keys (not acceptable) <<< index as key <<<<<<< unique id as key(Best Practice..).

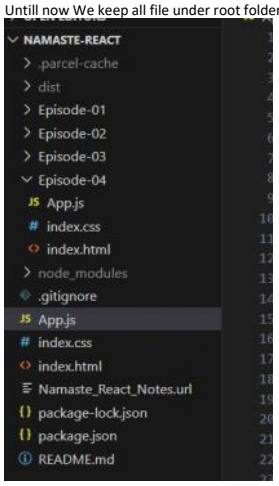
++
EPISODE 5: Let's Get Hooked
++

Episode 05 Part 01
==========

->EveryThing we can do using React, we can do same thing using HTML,CSS,JavaScript.
Why we use React ?
=>
Bcz if we use library or framework it makes our developer experience easy.
It makes you write less code.

->
Untill we write all our code In one file i.e. App.js but it is not good practice to write.
So we make separate file for separate component.

Untill now We keep all file under root folder as below:-

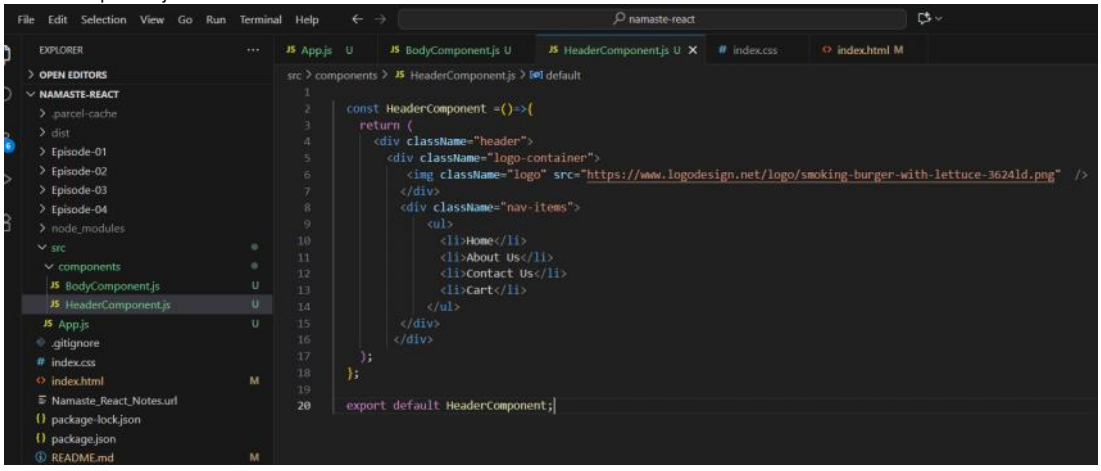


->all Main code of React library we keep it in src folder.so now on we keep all our js file inside the src folder.
We write all components inside the components folder.
->some people write .js or jsx extension.
What we should follow ?
->you whatever you want.
Both .js or jsx are similar.

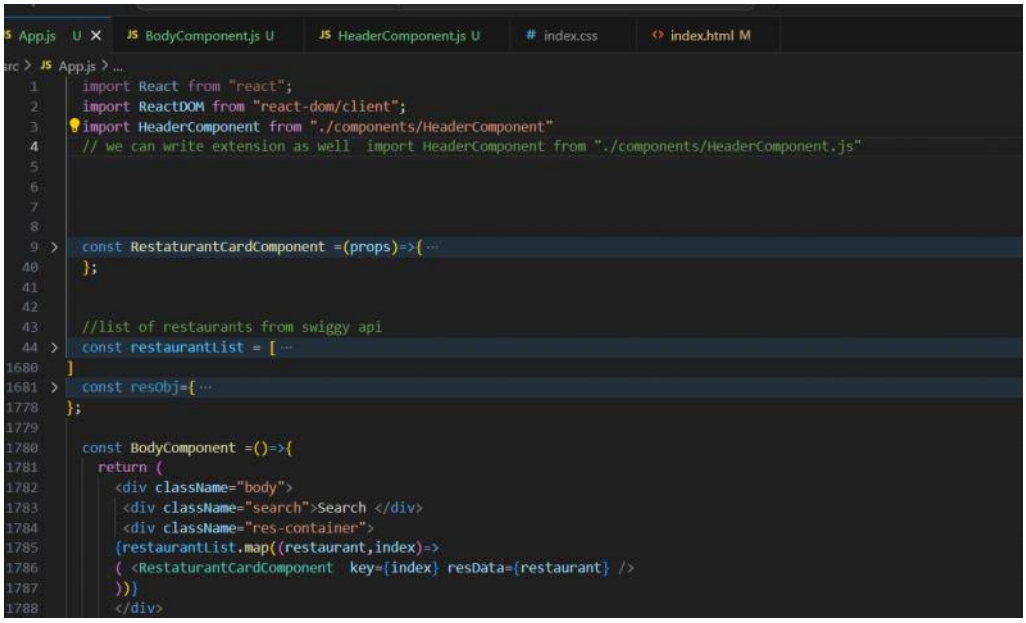
->in React folder stucture, we keep it anyway as you wish.

->let say we cut & paste our HeaderComponent in HeaderComponent.js file.
We will get error our code will not work.
We need to import this file into App.js & export our HeaderComponent code from HeaderComponent.js file.

HeaderComponent.js



App.js



Note::
For import Components,
We can write

```
  import HeaderComponent from "./components/HeaderComponent"
Or

import HeaderComponent from "./components/HeaderComponent.js"
```

Both are same.

->whenever we have hardcoded data, never keep it in Components.
Url is also hardcoded.

```
There 2 ways to export multiple things
1. Named export (To export multiple things)
2. Default export (to export only one thing)

1. Named export
export const  a=10;
export const  b=20;
export const  c=30;
so, using this we can export multiple things.


syntax to export
=> export const ComponentName/VariableName = value ;

to import named export
=>
 import {ComponentName/VariableName} from "filePath" ;


2. Default export
export default a;  //Way 2 (using this we can only export one thing)
export default b; (we get error if we try to export multiple things using default export)
export default CDN_URL;  //Way 1 (Normal WAY TO export) (using this we only  export one thing)


syantax to export
=> export default ComponentName/VariableName ;

To import default export
=> import VariableName from "filePath" ;
=> import ComponentName from "filePath" ;
```

Ther is slight diff while importing in both way of exporting.

1)to import Named export , we use curly braces.

```
import { CDN_URL } from "../utils/constants";  //if we use Named export
```

2)to import Default export , we don't need to use curly braces.

```
import  CDN_URL from "../utils/constants";  //if we use Default export
```

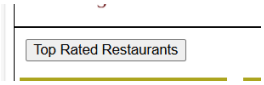
Q)Can we use default export with named export ?
=>

->In Some company, we write each component of max 100 line only.

Episode 05 Part 02
==========

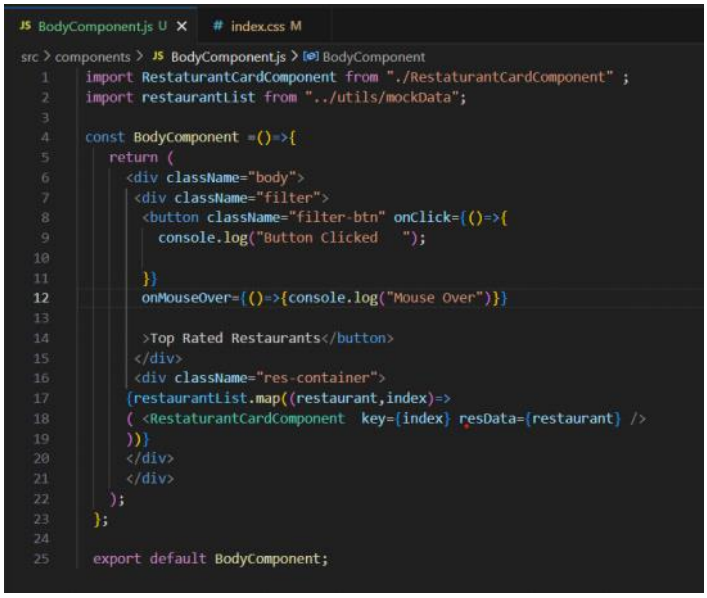
React Hooks

->We build a feature to show top rated Restaurant just by clicking button.



->create button using JSX inside bodyCompoent.
Also we can give multiple event listener also. Like onClick() , onMouseOver().

```
<div className="filter">
       <button className="filter-btn" onClick={()=>{
         console.log("Button Clicked   ");
        }}
       onMouseOver={()=>{console.log("Mouse Over")}}
       >Top Rated Restaurants</button>
      </div>
```

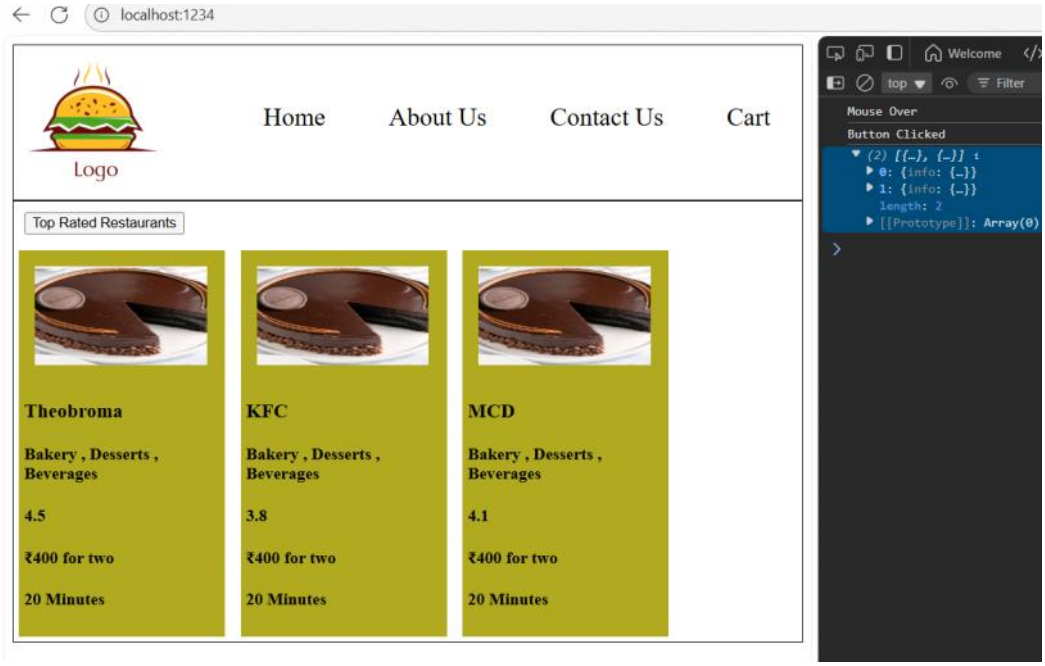


->On click of button, we must get top rated Restaurant list only. Not all restaurant.

->we wrote logic to filter out data based on rating>4.
But still we not able to see changes on UI.

```
<button className="filter-btn" onClick={()=>{
  console.log("Button Clicked   ");
  //On clck of button,we need to  filter the restaturants whose rating is >4
  listOfRestaurants=listOfRestaurants.filter(
    res => res.info.avgRating>4
  );
  console.log(listOfRestaurants);
//
}}
onMouseOver={()=>{console.log("Mouse Over")}}

>Top Rated Restaurants</button>
</div>
```

On console, we are getting, filtered 2 record. But not showing filtered record.
->React is faster bcz of DOM manipulation.



`listOfRestaurants` Currently, is normal javascript variable.

```
//Normal Javacsript Variable
let  listOfRestaurants =[
  {…
},  {…
},  {…
}
];
```

->To Create a super powerful React Variable i.e. State Variable
State Variable is super powerful variable.
To Create super powerful variables, for that we use **React Hook**. Also know an **Used State**.

React Hook is an normal javascript function given by React. This function has some super power.
This function has some logic behind the scene of React.

React Hooks
(it is Normal JS utility functions.
Facebook developer wrote this function inside the React).

->There are 2 important hooks.
1)useState() -- to generate superpowerful state variables in React.
2)useEffect()

1)useState()
=>

```
import {useState } from "react";  //import useState like name import.
```

src > components > BodyComponent.js > BodyComponent > listOfRestaurantsNew

```
import React from "react";  //Default Import
import ReactDOM from "react-dom/client";
 import RestaturantCardComponent from "./RestaturantCardComponent" ;
 import restaurantList from "../utils/mockData";
import {useState } from "react";  //Named Import


const BodyComponent =()=>{

  //Local State Variable  in React
   const [listOfRestaurants]=useState([]);  //default empty array we passed in useState.


   //Normal JS variables
   let listOfRestaurantsNew=[]; //default empty array
```

Syntax::

```
const [xx, setXX]=useState();  //way of Declaration of useState Hook.
Where xx is local state variable , setXX is function to update the value of xx.
```

=>whenever I want to update my React local state variable value, we not update value by reassigning as we do in normal javascript but,
In React we use setXX function to update the xx React local state variable value.

```
//  //Normal JS variables
 let list=[]; //default empty array we set.
   list =["ABC","PQR"];  //Re-assigning the value of list variable
    console.log(list);  //["ABC","PQR"]
```

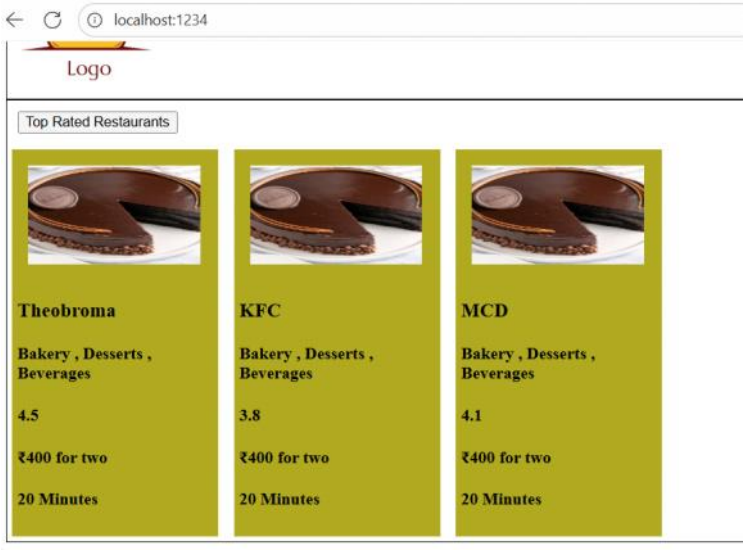
```
const BodyComponent =()=>{

  //Local State Variable  in React
   const [listOfRestaurants, setListOfRestaurant]=useState([
      {…
},   {…
},   {…
}
   ]);  // we passed  3 objects of array in useState.

   setListOfRestaurant([]); //updating the local state variable to empty array
```

=>Complete code

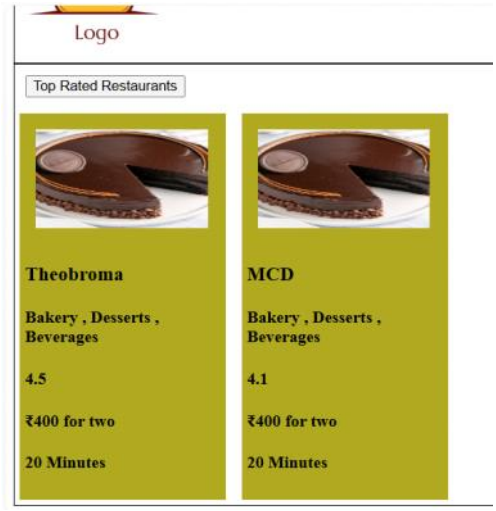
```
JS BodyComponent.js U ×   README.md M ●
src > components > JS BodyComponent.js > BodyComponent
 1   import React from "react";  //Default Import
 2   import ReactDOM from "react-dom/client";
 3   import RestaturantCardComponent from "./RestaturantCardComponent" ;
 4   import restaurantList from "../utils/mockData";
 5   import {useState } from "react";  //Named Import
 6
 7
 8   const BodyComponent =()=>{
 9     //Local State Variable  in React
10      const [listOfRestaurants, setListOfRestaurant]=useState([
11 >    {…
27 > },{…
43 > }, {…
59    }]);  // we passed  3 objects of array in useState.
60
```

```
61    return (
62      <div className="body">
63       <div className="filter">
64        <button className="filter-btn" onClick={()=>{
65        // filtering the restaurants based on rating>4. is is pure JS filter code.
66          const filteredList = listOfRestaurants.filter(res => res.info.avgRating>4);
67         setListOfRestaurant(filteredList);  //React way to update the local state variable
68        }}
69         >Top Rated Restaurants</button>
70       </div>
71       <div className="res-container">
72       {listOfRestaurants.map((restaurant,index)=>
73       ( <RestaturantCardComponent  key={index} resData={restaurant} />
74       ))}
75       </div>
76       </div>
77     );
78   };
79
80   export default BodyComponent;
```

Once we run application, we get UI as.



As soon we click on Top Rated Restaurant button, we get top list having Rating is greater than 4. the logic we wrote in behind.
Then we get output as.



=>so, if we use just normal JS variable to update the value, our value will be update but it will not be reflected on UI.
But if we use React's Local State variable, to update the value, our value will be updated & able to see on UI also.

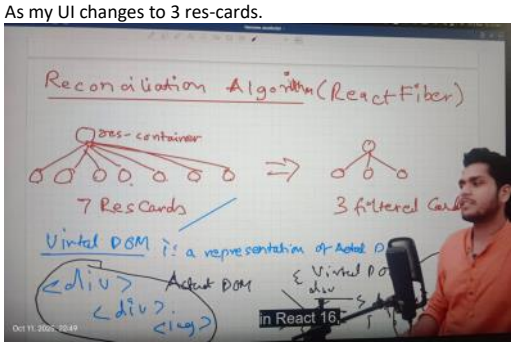
Note:: Whenever a State variable updates/changes, React will reRendering the component.

=>Now we learn React Hooks, we have create a mock Data for our use. So pass mockData into useState();
Like as below;

```
import React from "react";  //Default Import
import ReactDOM from "react-dom/client";
import RestaturantCardComponent from "./RestaturantCardComponent" ;
import restaurantList from "../utils/mockData";
import {useState } from "react";  //Named Import


const BodyComponent =()=>{
   //Local State Variable  in React
    const [listOfRestaurants, setListOfRestaurant]=useState(restaurantList);

```

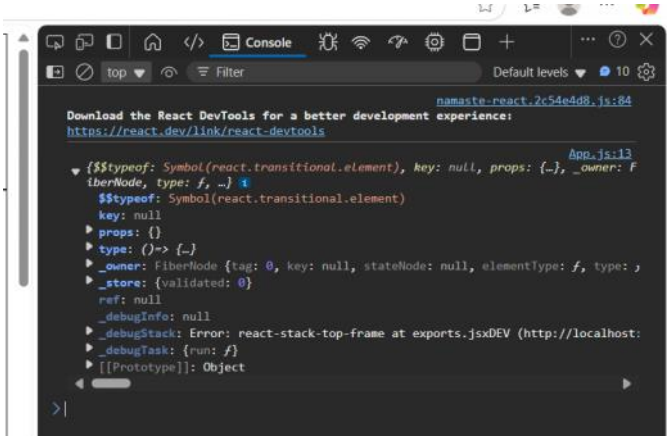
=>React is fast at reRendering our code.

=>React uses Reconciliation Algorithm behind the scenes. Reconciliation Algorithm is also know as React Fiber.
Ex:-
In DOM, I have one res-container class & its childs are 7 res-cards class. Which is orginal UI.
As my UI changes to 3 res-cards.



Q)What React do & Why react is fast ?
->
whenever we have original UI, React create virtual DOM of it.
Virtual DOM is not an actual DOM.
Actual DOM is
** **
** **
<image />
** **

Virtual DOM is representation of actual DOM.
Virtual DOM is basically those React Elements.
React Elements is an Object.
Below img is Virtual DOM of React Element.



->React Virtual DOM is basically kind of an Normal Javascript Object.

Q)What is Diff Algorithm ?

=>
Diff algorithm find out the diff between updated(New) virtual DOM & previous(Old) virtual DOM.
Once it find out the diff, then it updates on UI.

Whenever there is change in state variable, react will find out diff between virtual DOM & it will rerender our component.
It will update the DOM.

Go to Github architecture document & read it. :: https://github.com/acdlite/react-fiber-architecture

React is Fast bcz React is doing efficent DOM manipulation bcz it has virtual DOM.
React can efficiently find out diff between virtual DOMS & update the UI. This is core of React's algorithms.

Q)Why React Fast ?
->it has Virtual DOM.it has a diff algorithms which is very efficent. It can do efficient DOM manipulation. It can find out difference & update the UI.
This is core of React.

```
  //Diff ways to write local state variable in React
//Way=>1
 const arr=useState(restaurantList);
 const [listOfRestaurants, setListOfRestaurant]=arr; //Destructuring of array

//Way=>2
// const [listOfRestaurants, setListOfRestaurant]=useState(restaurantList); //Destructuring of array

  //way=>3
  const arr1=useState(restaurantList);
   const listOfRestaurants1=arr1[0]; //first element of array
   const setListOfRestaurant1=arr1[1]; //second element of array
```

React Hooks are nothing but normal javascript utility functions in React. It gives us super powerful state variables inside the react.
React is keeping an eye on react super powerful variables & keep track on it. Whenever they updates reacts updates diff algorithm.
Diff algorithms find out diff between virtual DOM's & it will automatically updates the UI. It keeps UI layer & Data layer in sync.
That is what core of React Algorithm.

++
EPISODE 6: Exploring the World
++

Episode 06 Part 01
==========

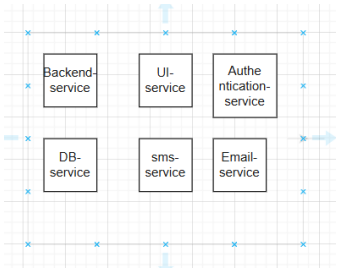
MonoLithic & Microservices Architecture

MonoLithic Architecture

->The whole one big project which contains
API code, UI code, Authentication logic, Database Code,
SMS code.
->everything is there in one project.
If any changes done, we need to deploy whole app.

Microservices Architecture

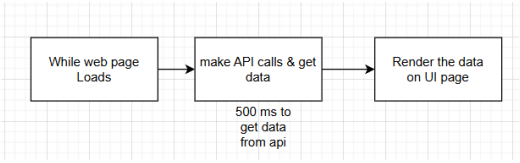
->we have diff diff services for each task.
Like Apl service, UI service, Authentication service,
Database service, SMS service, Email service.
->all this service talk with each other depending on requirement.
->it is known as separation of concerns & it follows single responsibility principles.
->Each service is deployed independently.



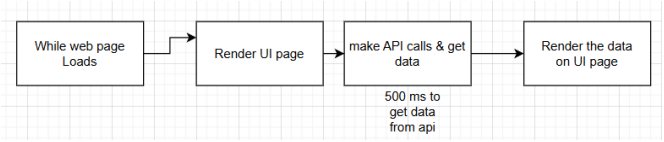
->Each service has its own portNo.
The namaste React project we building are falls under UI service.

Episode 06 Part 02
==========
->Currently we have mock data in pir project.
But now we will fetch data from live api i.e. from backend.

There are two approaches to fetch data from backend.
1)As soon as our page load, we make an API call, wait for data to come & then render the data.



2)As soon as the page loads, we will render the UI.
After we quickly render we will make an API call.
As soon as we get data from api, we will render the data on UI.



->In React, we always use 2nd approach.

But Why ?
=>it gives us better UX.
In first approach, initial 500ms, we did not see anything on UI. Then after that we suddenly we see everything.
It not good user Experience.
But in 2nd approach, initially we render whatever we can render then we make Api call & render that data on UI.
It is better user experience.

Even if render twice in react, it is not big problem bcz react has fast render mechanism.
So need to bother how many times we are rendering.

Episode 06 Part 03
==========
->Now we write code using 2nd approach.

What is Hook ?
->Hook is an Normal javascript function which is given by React.

useEffect Hook

->it is a normal react hook.
->Once the component is rendered fully completed, we will call callBackFunction we pass inside the useEffect.
so, if you have to do anything once the component is rendered, you can do it inside the useEffect.

Syntax::
useEffect(callBack Function,dependecy Array);

```
useEffect(()=>{
 console.log("Use effect called");

},[]);  //1st argument is callback function, 2nd argument is dependency array.
```

Ex:-

```
    /*
    Once the component is rendered fully completed, we will call callBackFunction we pass inside the us
    so, if you have to do anything once the component is rendered, you can do it inside the useEffect.
    */
    useEffect(()=>{
     console.log("UseEffect called");

    },[]);  //1st argument is callback function, 2nd argument is dependency array.

  console.log("Body component rendered..");
```

o/p=>
Body Component rendered.
UseEffect called.

Bcz first our component is rendered then useEffect call Back function is called.

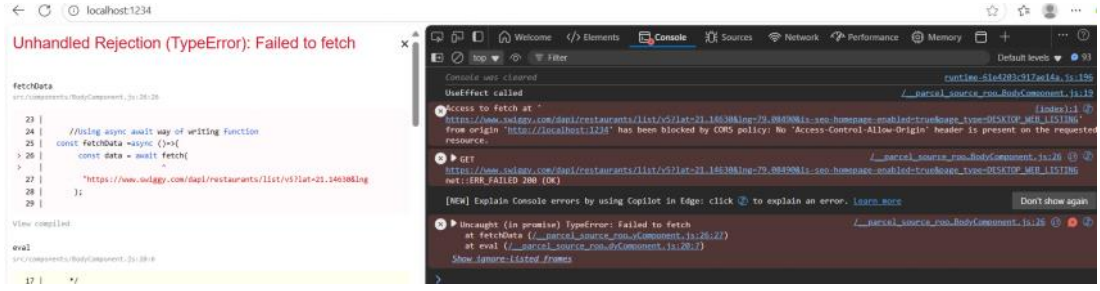
Q.)Where we use this useEffect ?
->we use in 2nd approach where first we rendered our compoent then we call the api.

Q.)can we call swiggy api in out console ?
=> lets try Using pure javascript.

```
 //Traditional way of writing function in pure Javascript
 WAY=>1.  //Pure Javascript function to fetch data from API
   const fetchData1 = ()=>{
    //fetch is given by browser to make API calls. not by Javascript.
    //fetch always returns a promise.
    const data = fetch(
     "https://www.swiggy.com/dapi/restaurants/list/v5?lat=21.14630&lng=79.08490&is-seo-homepage-enabled=true&page_type=DESKTOP_WEB_LISTING"
    );
   };


 WAY=>2.  //Using async await way of writing function
const fetchData =async ()=>{
     const data = await fetch(
      "https://www.swiggy.com/dapi/restaurants/list/v5?lat=21.14630&lng=79.08490&is-seo-homepage-enabled=true&page_type=DESKTOP_WEB_LISTING"
     );

     const jsonData= await data.json();
     console.log(jsonData);
}
```

```
    /*
    Once the component is rendered fully completed, we will call callBackFunction we pass inside the useEffect.
    so, if you have to do anything once the component is rendered, you can do it inside the useEffect.
    */
    useEffect(()=>{
     console.log("UseEffect called");
     fetchData();

    },[]);  //1st argument is callback function, 2nd argument is dependency array.

    //Using async await way of writing function
  const fetchData =async ()=>{
      const data = await fetch(
       "https://www.swiggy.com/dapi/restaurants/list/v5?lat=21.14630&lng=79.08490&is-seo-homepage-enabled=true&page_type=DESKTOP_WEB_LISTING"
      );

      const jsonData= await data.json();
      console.log(jsonData);
  }
```

O/P in console=>



We get error in console saying.
(index):1 Access to fetch at 'https://www.swiggy.com/dapi/restaurants/list/v5?lat=21.14630&lng=79.08490&is-seo-homepage-enabled=true&page_type=DESKTOP_WEB_LISTING' from origin 'http://localhost:1234' has been blocked by CORS policy: No 'Access-Control-Allow-Origin' header is present on the requested resource.

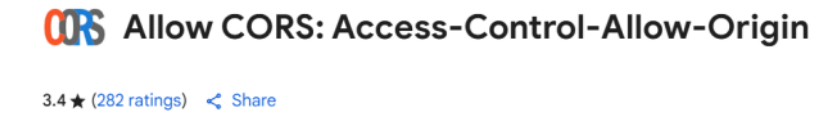
SO basically calling swiggy api from localHost has been blocked due to CORS policy.

Q.) Who is blocking us ?
=>Browser blocking us to call API from one origin to diff origin.

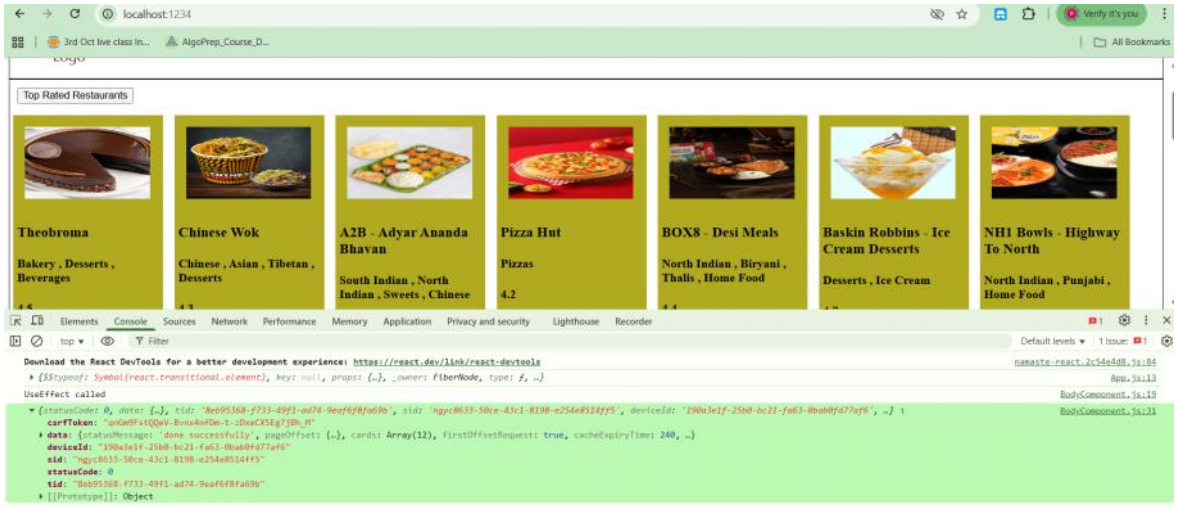
If origin mismatch browser blocks that api call.
This is CORS policy.

Watch CORS policy video by akshay saini in YT.

Q.) How to byPass this CORS ?
=>Add Allow CORS extension in our browser.
& enable it.



->Now we are able to access swiggy API.



->Now we want to render our page with new data which we fetch from external api.
->As sson we get data from api call, we will puta that data into listOfRestaurants.
As soon as our listOfRestaurants, update react will rerender the this compoent & update UI with new data.

=>
This is the way we can set data after we get from API.

```
    /*
    Once the component is rendered fully completed, we will call callBackFunction we pass inside the useEffect.
    so, if you have to do anything once the component is rendered, you can do it inside the useEffect.
    */
    useEffect(()=>{
     console.log("UseEffect called");
     fetchData();

    },[]);  //1st argument is callback function, 2nd argument is dependency array.

    //Using async await way of writing function
  const fetchData =async ()=>{
      const data = await fetch(
       "https://www.swiggy.com/dapi/restaurants/list/v5?lat=21.14630&lng=79.08490&is-seo-homepage-enabled=true&page_type=DESKTOP_WEB_LISTING"
      );
       //get JsON from the response object
      const jsonData= await data.json();
      console.log(jsonData.data.cards[4].card.card.gridElements.infoWithStyle.restaurants);
      //set the data which we fetch from API to the local state variable
      setListOfRestaurant(jsonData.data.cards[4].card.card.gridElements.infoWithStyle.restaurants);
}
```

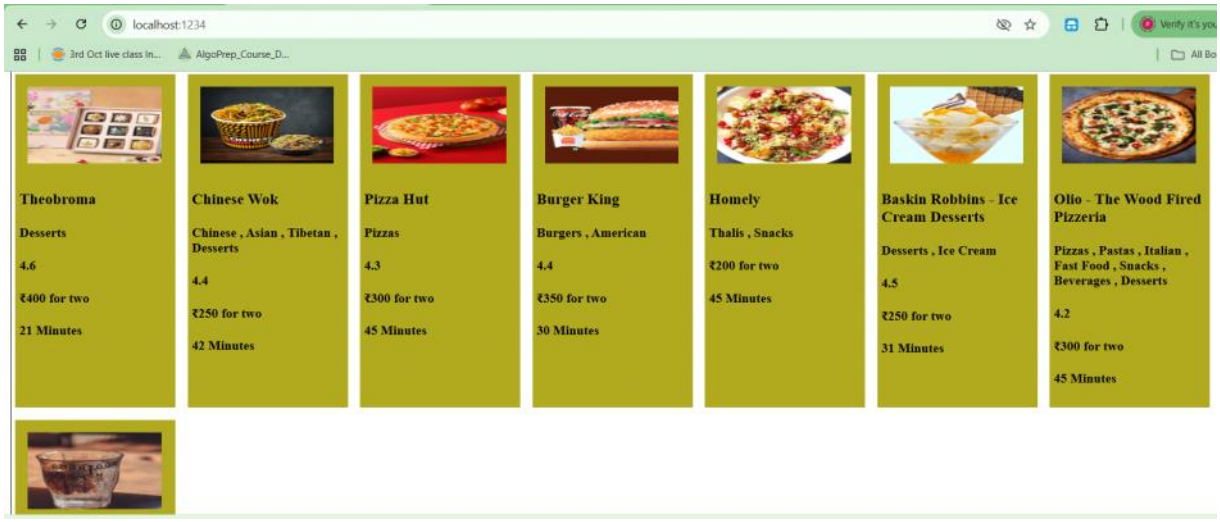
->we set mockData of restaurantList from mockData.js.
Before our data load completely it will show some mock data.
Once load data from external api, it will show that data from external api.

```
  //Local State Variable  in React
  const arr1=useState(restaurantList);   //to set mockData initially before API data loads.
  const arr=useState([]);  //Initially empty array before API data loads.
   const [listOfRestaurants, setListOfRestaurant]=arr; //Destructuring of array
```

->we can set initial data as empty.
So we delete that mock data.js file

```
 const arr=useState([]);  //Initially empty array before API data loads.
```

->Now our external api data looks like:-



->This is not a good way to write a code.

```
setListOfRestaurant(jsonData.data.cards[4].card.card.gridElements.infoWithStyle.restaurants);
```

So, we use optional chaining.

```
//Optional Chaining Operator (?.) is used to avoid runtime errors if any property is undefined or null.
setListOfRestaurant(jsonData?.data?.cards[4]?.card?.card?.gridElements?.infoWithStyle?.restaurants);
```

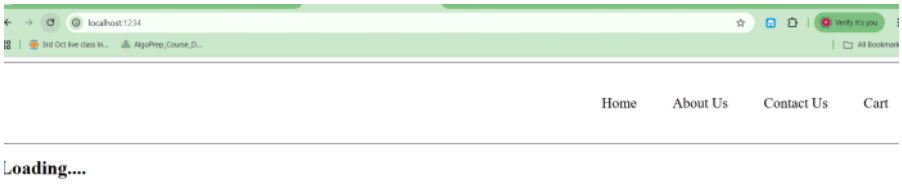
Note :: Read about optional Chaining in Javascript.

Episode 06 Part 04
==========
->now we have fetched the data & we seen that data which we fetch from api.
But as soon as we load the webpage, we see blank page for some seconds until component is rendered.

So, how can we improve user experience of user ?
=> we can show spinning loader.
Until our data fully load, we use spinning loader.
Ex:-

```
50
51
52  // listOfRestaurants is empty array initially. So, we will show loading....
53  //  until the data is fetched from API and set to the local state variable.
54    if(listOfRestaurants.length===0){
55     return <h1>Loading....</h1>;
56    }
57
58
59     return (
60       <div className="body">
61        <div className="filter">
62         <button className="filter-btn" onClick={()=>{
63         // filtering the restaurants based on rating>4. is is pure JS filter code.
64           const filteredList = listOfRestaurants.filter(res => res.info.avgRating>4.5);
65         setListOfRestaurant(filteredList);  //React way to update the local state variable
66         }}
67         >Top Rated Restaurants</button>
68        </div>
69        <div className="res-container">
70        {listOfRestaurants.map((restaurant,index)=>
71        ( <RestaturantCardComponent  key={index} resData={restaurant} />
```

So our Until Our BodyComponent not load full, we show Loading.. Loader.



Q.)Is showing spinner is a good way ?
=>No. it is not.

Today in industry there is a concept called as "Shimmer UI".
Shimmer UI resembles the page's actual UI so that user understands how quickly web app will load.
So it is kind of like we load fake page until we get actual data from the Api.

->Lets make shimmer card manually,

1)Create ShimmerComponent

```
const ShimmerComponet= ()=>{
    return ( <div className="shimmer-container">
        <div className="shimmer-card">Cards</div>
        <div className="shimmer-card">Cards</div>
        <div className="shimmer-card">Cards</div>
    </div>

    );
}

export default  ShimmerComponet;
```

2) Add some css,

```
.shimmer-container{
    display: flex;
    flex-wrap: wrap;
}

.shimmer-card{
    margin: 20px;
    width:200px;
    height:400px;
    background-color: rgb(211, 200, 200);
}
```

3)import ShimmerComponent in BodyComponent.

```
// listOfRestaurants is empty array initially. So, we will show loading....
//  until the data is fetched from API and set to the local state variable.
 if(listOfRestaurants.length===0){
  return <ShimmerComponet />;
 }
```

o/p=>
So, untill our web page load completely we get this below shimmer UI.

->Using Shimmer UI makes user feels that our page is loading.

Episode 06 Part 05
==========

Conditional Rendering concept.

Rendering on the basis of condition is conditional rendering.

```
// listOfRestaurants is empty array initially. So, we will show loading....
//  until the data is fetched from API and set to the local state variable.

//This  is Conditional Rendering concept.
//Rendering on the basis of condition is conditional rendering.
 if(listOfRestaurants.length===0){
  return <ShimmerComponet />;
 }
```

->we can write our code using ternary operator instead of if condition for conditional rendering.

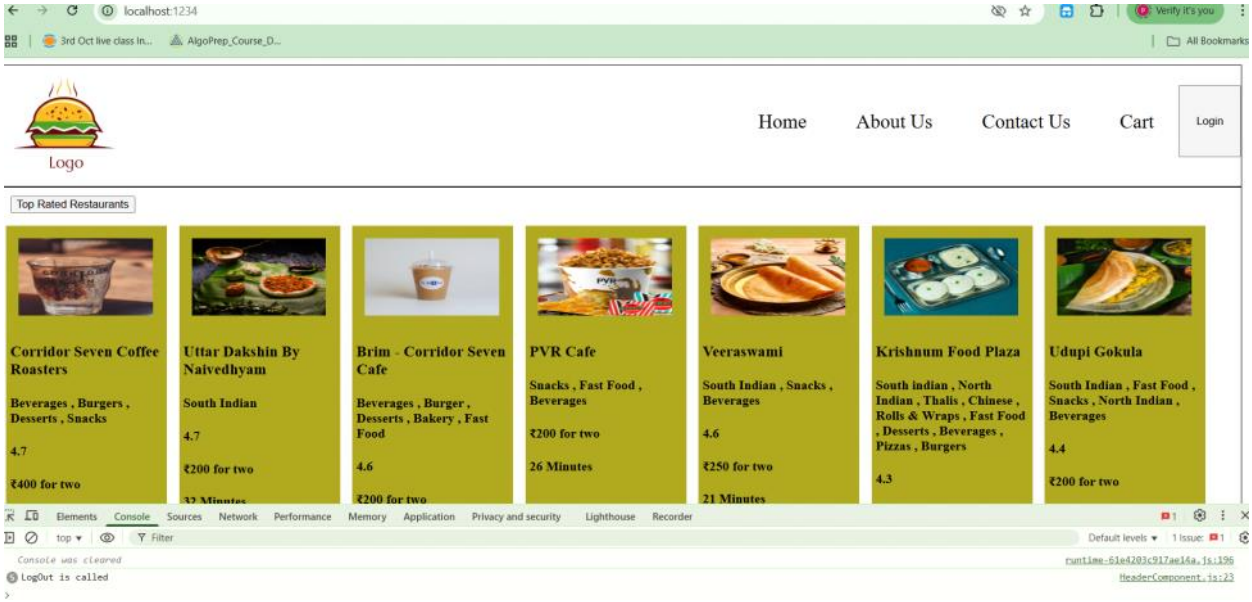
```
  return     listOfRestaurants.length===0 ? <ShimmerComponet/> : (
    <div className="body">
     <div className="filter">
      <button className="filter-btn" onClick={()=>{
      // filtering the restaurants based on rating>4. is is pure JS filter code.
        const filteredList = listOfRestaurants.filter(res => res.info.avgRating>4.5);
       setListOfRestaurant(filteredList);  //React way to update the local state variable
       }}
       >Top Rated Restaurants</button>
     </div>
     <div className="res-container">
    {listOfRestaurants.map((restaurant,index)=>
    ( <RestaturantCardComponent  key={index} resData={restaurant} />
    ))}
    </div>
    </div>
  );
};

export default BodyComponent;
```

Q)what is need to state variables ? When ? Why we used it useState() ?
->
So, lets build one button in header for Login/Logout.
We not use any Authentication here.

->First we try using Pure JavaScript variables approach.
Ex:-

```
const HeaderComponent =()=>{

  //Normal Javascript variable
  let btnName="Login";


  return (
    <div className="header">
      <div className="logo-container">
        <img className="logo" src={LOGO_URL} />
      </div>
      <div className="nav-items">
          <ul>
            <li>Home</li>
            <li>About Us</li>
            <li>Contact Us</li>
            <li>Cart</li>
            <button  className="login" onClick={
              ()=>{
                btnName ="LogOut";
                console.log(btnName,"is called ");
              }}>{btnName}</button>
          </ul>
      </div>
    </div>
```

Our variable name is updated from Login to Logout when we click on button,
But still we not able to see our output on UI page.
Our variable name is change from Login to Logout, we can see in console.
It is because our UI component is not rendered.
So, if we want to change something dynamically & that changes must be reflected on UI, we use useState state variables in React.



=>Using React variables approach,

const [btnNameReact,setBtnNameReact]= useState("Login");
-> btnNameReact is speacial react variable. we can not directly modify it.

Ex:- btnNameReact intial value is "Login".
we want to change it to "LogOut".
Using Direct way by assigning value is not work here.
btnNameReact="Logout" --> this is wrong.
so, we have to use the function setBtnNameReact to modify it.
setBtnNameReact("Logout") --> this is correct.

->Whenver our React state variable changes, using setBtnNameReact function,
React will re-render the HeaderComponent & we get Updated value.

```
const HeaderComponent =()=>{
  const [btnNameReact,setBtnNameReact]= useState("Login");
  /* btnNameReact is speacial react variable.
     we can not directly modify it.

     ex:- btnNameReact intial value is "Login".
     we  want to change it to "LogOut".
     Using Direct way by assigning value is not work here.
      btnNameReact="Logout"  --> this is wrong.

     so, we have to use the function setBtnNameReact to modify it.
     setBtnNameReact("Logout") --> this is correct.
  */
  return (
    <div className="header">
      <div className="logo-container">
        <img className="logo" src={LOGO_URL} />
      </div>
      <div className="nav-items">
          <ul>
            <li>Home</li>
            <li>About Us</li>
            <li>Contact Us</li>
            <li>Cart</li>
            <button  className="login" onClick={
              ()=>{
                  setBtnNameReact("LogOut");
                console.log(btnNameReact,"is called ");
              }}>{btnNameReact}</button>
          </ul>
```

Q) Is react rendering whole code again or just changes will re-render ?
=> It will re-render the whole header component.

Q) const [btnNameReact,setBtnNameReact]= useState("Login");
btnNameReact is an constant variable then how is it updating value in JavaScript ?
Is it violating JavaScript principles ?
->
When we rerender our component this function is calling once again & that whole component will re-render again.
So it will treat all variables as new variables & assign value to it.

Ex:- let say intialTime our btnNameReact state variable is "Login".
As soon as we pass "LogOut" to setBtnNameReact function, React will re-render whole component.
At that time it will treat all variables inside that component are new variables & assign new value to it.
React keeps track of everything.

While re-rendering it will find difference between older version & newer version.

->We want to implement toggle functionality to change from login to logout & vice versa.

```
            <li>Home</li>
            <li>About Us</li>
            <li>Contact Us</li>
            <li>Cart</li>
            <button  className="login" onClick={
              ()=>{
                  btnNameReact==="Login"? setBtnNameReact("LogOut"):  setBtnNameReact("Login");
              }}>{btnNameReact}</button>
          </ul>
      </div>
    </div>
```

Episode 06 Part 06
==========
Now, let's Build Search Functionality.

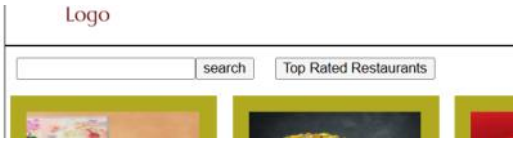
1)add search HTML in Body.

```
return     listOfRestaurants.length===0 ? <ShimmerComponet/> : (
  <div className="body">
   <div className="filter">
    <div  className="search">
        <input  type="text"  className="search-box"  />
        <button>search</button>
    </div>
    <button className="filter-btn" onClick={()=>{···
    >Top Rated Restaurants</button>
   </div>
   <div className="res-container">
  {listOfRestaurants.map((restaurant,index)=>
  ( <RestaturantCardComponent  key={index} resData={restaurant} />
  ))}
  </div>
  </div>
 );
};

export default BodyComponent;
```

2)add some css.

```
.filter{
    display: flex;;
}
```

Output on UI:-



->to get data from input box we need to use value.
We need to bind this input box to local state variables.

```
const[searchText, setSearchText]= useState(""); //Local State Variable for search box

return     listOfRestaurants.length===0 ? <ShimmerComponet/> : (
  <div className="body">
   <div className="filter">
    <div  className="search">
        <input  type="text"  className="search-box"  value={searchText} />
        <button
        onClick={
         ()=>{
           //Filter the restaurant cards & update the UI.
           //we need searchText here
           console.log(searchText);

         }
        }
        >search</button>
    </div>
```

->I have bind the state variable with input box but still I am able to give input in my input box.
Why ?
->Bcz my input box Is tied to the searchText.
So my searchText is not getting updated but I am trying to modify my input box.
It can't happen.
So, our input box will not change unless we change the search text.
It will be handle using onChange event.

```
62
63      const[searchText, setSearchText]= useState(""); //Local State Variable for search box
64
65    return     listOfRestaurants.length===0 ? <ShimmerComponet/> : (
66      <div className="body">
67       <div className="filter">
68        <div  className="search">
69            <input  type="text"  className="search-box"  value={searchText}
70              onChange={ (e)=>{
71                    setSearchText(e.target.value);
72              }}
73             />
74             <button
75             onClick={
76              ()=>{
77                //Filter the restaurant cards & update the UI.
78                //we need searchText here
79                console.log(searchText);
80
81              }
82             }
83             >search</button>
```

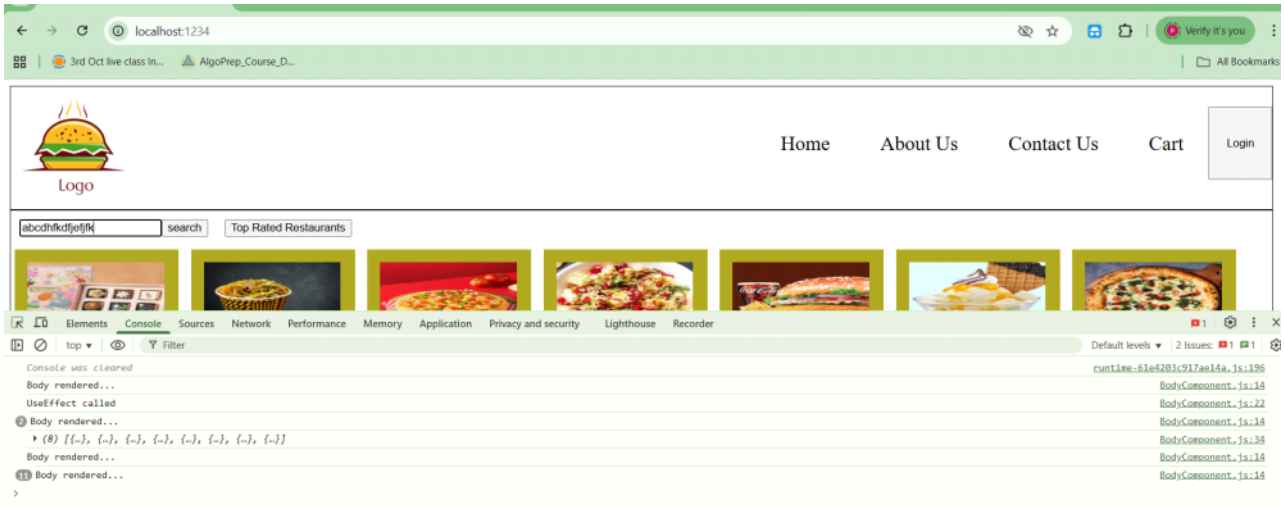
->To get input from UI,
First we need to bind our input value with React state variable with empty string or any default value.
Ex:-

const[searchText, setSearchText]= useState(""); //Local State Variable for search box

<input type="text" className="search-box" value={searchText} />

Here we tied our input value with searchText state variable.

Now, I want as soon as my input changes, I want to update into the searchText variable.
So, we use onChange event.



-> as soon as we type any character that no.of times it will rendered body that no.of times.

NOTE::
1)Whenever React's state variables update, React triggers a reconciliation cycle(Re-Renders the Component).
2)Even if re-render whole component but only that things update by comparing older & newer version of component.
Due to this React is faster.

3)Virtual DOM is not making React Faster.
The React Fiber the new Reconciliation algorithm which finds out difference between two virtual DOM & updates the DOM only when required.
& portion of DOM which only that portion is required.

4)Watch map,filter,reduce video of javascript from youtube.

Now, our search buttion is working.
We need to filter out our records based on search.

So, wrote our code,

```
const[searchText, setSearchText]= useState(""); //Local State Variable for search box

return    listOfRestaurants.length===0 ? <ShimmerComponet/> : (
  <div className="body">
   <div className="filter">
    <div  className="search">
        <input  type="text"  className="search-box"  value={searchText}
         onChange={ (e)=>{
            setSearchText(e.target.value);
         }}
        />
        <button
        onClick={()=>{
          //Filter the restaurant cards & update the UI.
          //we need searchText here
const filterRestaurantList=listOfRestaurants.filter(res=>
res.info.name.toLowerCase().includes(searchText.toLowerCase())
);
//Update the restaurant list by filtered restaurant list.
setListOfRestaurant(filterRestaurantList);
        }}
        >search</button>
    </div>
     <button className="filter-btn" onClick={()=>{
     >Top Rated Restaurants</button>
    </div>
    <div className="res-container">
    {listOfRestaurants.map((restaurant,index)=>
    ( <RestaurantCardComponent  key={index} resData={restaurant} />
    ))}
    </div>
    </div>
  );
```

->One problem is after we search first, we get result.
But if we search for second time we not get result. Because on our UI still we have 1st output result.
& our search 2nd time will depend on 1st output result.

So, need to solve this issue ? So that even if we type 2nd time for search we get data from a all ListOfRestaurant which are not filtered.

Solution

Episode 06 Part 07
==========
So, we will create a separate react variable store filtered data.
& we will use a separate react variable to store data which we fetch from api.
So, we did not modify data which we fetch from api.

So, we make all changes in
```
  const arr=useState([]);  //Initially empty array before API data loads.
   const [listOfRestaurants, setListOfRestaurant]=arr; //Destructuring of array

   //Whenever React's state variables update, React triggers a reconciliation cycle(Re-Renders the Component)
console.log("Body rendered...");
```

const[searchText, setSearchText]= useState(""); //Local State Variable for search box
const[filteredRestaurants, setFilteredRestaurants]= useState([]); //Local State Variable for filtered restaurant list.

Where listOfRestaurants stored data which we fetch from API.
filteredRestaurants stored filtered data.

```
import ReactDOM from "react-dom/client";
import RestaturantCardComponent from "./RestaturantCardComponent" ;
import {useState, useEffect } from "react";  //Named Import
import ShimmerComponet from "./ShimmerComponent";


const BodyComponent =()=>{
   //Local State Variable  in React
  // const arr1=useState(restaurantList);   //to set mockData initially before API data loads.
   const arr=useState([]);  //Initially empty array before API data loads.
   const [listOfRestaurants, setListOfRestaurant]=arr; //Destructuring of array

   //Whenever React's state variables update, React triggers a reconciliation cycle(Re-Renders the Component)
console.log("Body rendered...");

    const[searchText, setSearchText]= useState(""); //Local State Variable for search box
    const[filteredRestaurants, setFilteredRestaurants]= useState([]); //Local State Variable for filtered restaurant list.


    /*
    Once the component is rendered fully completed, we will call callBackFunction we pass inside the useEffect.
    so, if you have to do anything once the component is rendered, you can do it inside the useEffect.
    */
    useEffect(()=>{
     console.log("UseEffect called");
     fetchData();

    },[]);  //1st argument is callback function, 2nd argument is dependency array.
```

```
  //Using async await way of writing function
const fetchData =async ()=>{
    const data = await fetch(
      "https://www.swiggy.com/dapi/restaurants/list/v5?lat=21.14630&lng=79.08490&is-seo-homepage-enabled=true&page_type=DESKTOP_WEB_LISTING"
    );
     //get JsON from the response object
    const jsonData= await data.json();
    console.log(jsonData.data.cards[4].card.card.gridElements.infoWithStyle.restaurants);
    //set the data which we fetch from API to the local state variable

    //Optional Chaining Operator (?.) is used to avoid runtime errors if any property is undefined or null.
    setListOfRestaurant(jsonData?.data?.cards[4]?.card?.card?.gridElements?.infoWithStyle?.restaurants);

    //when we fetch the data from API, we have to set the filtered restaurant list also.
    setFilteredRestaurants(jsonData?.data?.cards[4]?.card?.card?.gridElements?.infoWithStyle?.restaurants);
}
```

```
    return     listOfRestaurants.length===0 ? <ShimmerComponet/> : (
      <div className="body">
       <div className="filter">
        <div  className="search">
           <input  type="text"  className="search-box"  value={searchText}
            onChange={ (e)=>{
                setSearchText(e.target.value);
            }} />
           <button
           onClick={()=>{ //Filter the restaurant cards & update the UI.  //we need searchText here ignore case.
const filterRestaurantList=listOfRestaurants.filter(res=>
 res.info.name.toLowerCase().includes(searchText.toLowerCase())
);
//update the restaurant list by filtered restaurant list.
 setFilteredRestaurants(filterRestaurantList); }}
            >search</button>
        </div>
        <button className="filter-btn" onClick={()=>{
        // filtering the restaurants based on rating>4. is is pure JS filter code.
          const filteredList = listOfRestaurants.filter(res => res.info.avgRating>4.5);
         setFilteredRestaurants(filteredList);  //React way to update the local state variable
        }}
        >Top Rated Restaurants</button>
       </div>
       <div className="res-container">
      {filteredRestaurants.map((restaurant,index)=>
      ( <RestaturantCardComponent  key={index} resData={restaurant} />
      ))}
      </div>
      </div>
    );
  };
```

++
EPISODE 6.1: Swiggy API Issue Resolved
++

Episode 06.1
==========

Our API is changed to new API. So we need to change our API in our UI code.
& make necessary changes.

Install :- JSON viewer in chrome as extension.

If api changes it might me data in which there is diff path,

So ,we need to fix the path.

->here we have set id of each restaurant also.

```
        </div>
        <div className="res-container">
        {filteredRestaurants.map((restaurant)=>
        ( <RestaturantCardComponent  key={restaurant.info.id} resData={restaurant} />
        ))}
        </div>
        </div>
      );
    };
```

++
EPISODE 6.2: CORS plugin Issue
++

Episode 06.2
==========
If we not use CORS plugin extension in browser , we not able to access other domain API.
So we study how to access api without using CORS extension.
When we make a call from one domain name to another domain name which both are diff.
Then browser blocks requests.
Ex:- www.localHost.com to www.swiggy.com

Website:: https://corsproxy.io/
To bypass this CORS issue,
We have https://corsproxy.io/
So, we need to copy our url & paste it before https://corsproxy.io/?

So now we are not calling directly swiggy api. We calling swiggy api through corsproxy.io

URL=>
https://corsproxy.io/?url=https://example.com

Here you can paste URL to bypass that CORS

So, we can access that api without CORS plugin in browser.

But I am getting error while using https://corsproxy.io/?url=https://example.com
So, I will use CORS plugin in browser.

++
EPISODE 7: Finding the Path
++

Episode 07 Part 01
==========

Routing
-> we create diff pages like diff urls.

How UseEffect is called ?
->
Lets learn about useEffect().
---
1)useEffect will every time called after the component is rendered.
Syntax:- useEffect(()=>{},[]);
 2) [] --> this is dependency array. it is an optional parameter.
  3)only callback function is mandatory parameter.

  1) if no depndency array is provided.

ex:-
useEffect(()=>{
log("Header useEffect called");
});
->if we don't provide dependency array, useEffect will be called every time after the component is rendered.
so, as soon as we make changes, our Component as well as useEffect will call every time.

```
*/
const [btnNameReact,setBtnNameReact]= useState("Login");
console.log("Header Rendered");

 useEffect(()=>{
   console.log("Header useEffect called");
 });
```

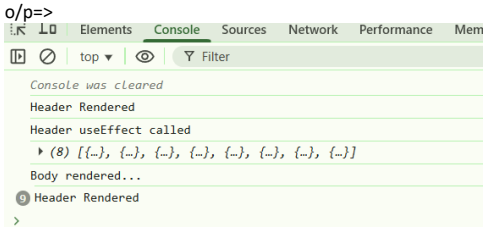
o/p on console=>

```
Body rendered...
Header useEffect called
UseEffect called
 ▸ (8) [{…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}]
Body rendered...
Header Rendered
Header useEffect called
Header Rendered
Header useEffect called
Header Rendered
Header useEffect called
Header Rendered
Header useEffect called
```

2) if empty depndency array is provided.
ex:-
useEffect(()=>{
log("Header useEffect called");
},[]);
->if we provide empty dependency array, useEffect function will be called only one time after the first render of the component.
then component can be called again when component re-render & again but not useEffect function.

```
const HeaderComponent =()=>{
  const [btnNameReact,setBtnNameReact]= useState("Login");
  console.log("Header Rendered");

   useEffect(()=>{
     console.log("Header useEffect called");
   },[]);
```

o/p=>



=>Default behaviour of useEffect is called after each re-render.
But if we give dependency array, it just called once.
=>if we put something inside the dependency array, then it will be called when dependency changes.

3) if we provide dependency array with some variables.
then useEffect will be called only when those variable values are changed.
ex:-
useEffect(()=>{
console.log("Header useEffect called");
},[btnNameReact]);

```
const [btnNameReact,setBtnNameReact]= useState("Login");
console.log("Header Rendered");

 useEffect(()=>{
   console.log("Header useEffect called");
 },[btnNameReact]);
```

o/p=>

```
Header Rendered
Body rendered...
Header useEffect called
UseEffect called
 ▸ (8) [{…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}]
Body rendered...
Header Rendered
Header useEffect called
Header Rendered
Header useEffect called
Header Rendered
Header useEffect called
Header Rendered
Header useEffect called
```

Episode 07 Part 02
==========

Few more things about useState hook now.

->Never create useState variables out of our component scope/area.
Otherwise it will throw error.
->useState is used to create local state variables inside our functional components.
->Always try to use when the function starts at TOP.
Ex:- in this image we have create our state variables first.

```
const BodyComponent =()=>{
    //Local State Variable  in React
    // const arr1=useState(restaurantList);   //to set mockData initially before API data loads.
     const arr=useState([]);  //Initially empty array before API data loads.
      const [listOfRestaurants, setListOfRestaurant]=arr; //Destructuring of array
     const[searchText, setSearchText]= useState(""); //Local State Variable for search box
     const[filteredRestaurants, setFilteredRestaurants]= useState([]); //Local State Variable for filtered restaurant list.
      //Whenever React's state variables update, React triggers a reconciliation cycle(Re-Renders the Component)
console.log("Body rendered...");
```

Because JavaScript is synchronous single threaded language. It execute our code line by line.
So that react can understand properly.
->Never create our useState variables inside the if-else statement.
Ex:-

```
// not good practice to create state variable inside conditional block
 if(true){
 const[searchText, setSearchText]= useState(""); //Local State Variable for search box
 }
```

If we try to use, our code may be mess up or inconsistency in program.

-> never create our useState variables inside the for loop. Don't do that inside the functions.

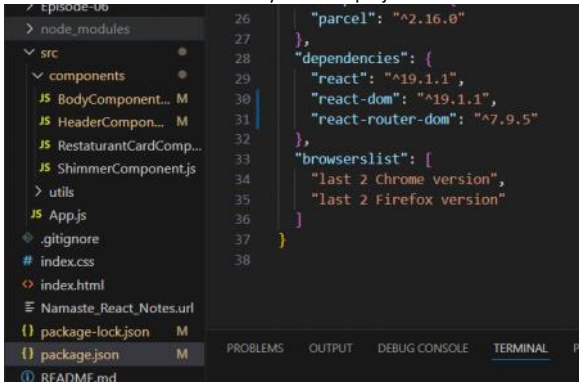
```
// Do not create state variable inside function or loop or conditional block
 function test(){
     const[searchText, setSearchText]= useState(""); //Local State Variable for search box
 }

 for(let i=0;i<10;i++){
   const[searchText, setSearchText]= useState(""); //Local State Variable for search box
 }
```

=>If we follow above best practice, we did not see any issue in our code.

Episode 07 Part 03
==========

Routing

->To use react router, we need to add react-router library.
Open terminal type command:- npm i react-router-dom
now our react-router-dom library added in project.



->Currently our Home page url is http://localhost:1234
As soon as I write http://localhost:1234/about I get aboutUs page.
How do I do that ?
->
Whenever we need to create routes, we need to create routing configuration.
So, we will create routes configuration at App.js which is mail file.

configuration means what will happens on a specific routes.

How to create Configuration ?
->I need to import createBrowserRouter from react-router-dom.
This createBrowserRouter takes some configuration.
That configuration is an list.
Each & every object in a list defines a different paths.

createBrowserRouter is used to create a router object.
We can define multiple routes inside the array we pass to createBrowserRouter.

Each route is an object with path and element properties.
path: URL path for the route.
element: React component to be rendered for that route.
Here, we have defined two routes:
1) "/" route which renders AppLayOut component.
2) "/about" route which renders About component.

We have created one About component.

```
components > JS About.js > [@] About
1   const About=()=>{
2       return(
3           <div>
4               <h1>About Us Page</h1>
5               <p>This is Namaste React Live Course 2023</p>
6               <p>We are learning React JS</p>
7               <p>Our Mentor is  Akshay Sir</p>
8           </div>
9       )
10  }
11
12  export default About;
```

App.js

```
import { createBrowserRouter,RouterProvider } from "react-router-dom"; //This is wrong import statement
import { createBrowserRouter,RouterProvider } from "react-router"; //This is correct import statement
import About from "./components/About";
```

```
/**
 * createBrowserRouter is used to create a router object.
 * We can define multiple routes inside the array we pass to createBrowserRouter.
 * Each route is an object with path and element properties.
 * path: URL path for the route.
 * element: React component to be rendered for that route.
 * Here, we have defined two routes:
 * 1) "/" route which renders AppLayOut component.
 * 2) "/about" route which renders About component.
 */
const appRouter=createBrowserRouter([
  {
    path:"/",
    element:<AppLayOut/>
  },{
    path:"/about",
    element:<About/>
  }
]);
```

So, if I hit url "/", our AppLayout component will load.
, if I hit url "/about", our About component will load.

Now we have created this configuration. Now we need to provide this configuration to render it.
For this we have RouterProvider component from react-router-dom.
RouterProvider providing our configuration to our app.

->Earlier we render our AppLayout component directly.
Like this,

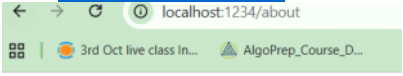
```
root.render(<AppLayOut />);
```

Now, instead of this, we will provide it my router configuration.

```
root.render(<RouterProvider  router={appRouter} />);
```

Now, we website able to handle routing facility.

URL=> http://localhost:1234/about



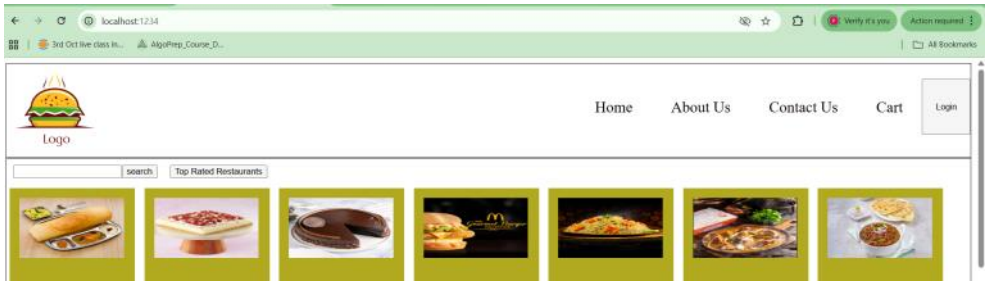
About Us Page

This is Namaste React Live Course 2023

We are learning React JS

Our Mentor is Akshay Sir

URL=> http://localhost:1234/
We get home page.



->
Website for react-router.
https://reactrouter.com/start/modes

There are some many routers are available but most recommended router is createBrowserRouter

->Lets create contact Us page for Routing.
First we create Contact component .
Shortcut to create component is rafce + Enter(but not recommended).

```
BodyComponent.js M   App.js M   Contact.js U   About.js U
src > components > Contact.js > Contact
1   const Contact=()=>{
2       return(
3           <div>
4               <h1>Contact Us Page</h1>
5               <p>This is Namaste React Live Course 2023</p>
6               <p>Contact No :: 1234567890 </p>
7               <p>Our Email is :: pr1234@gamil.com</p>
8           </div>
9       )
10  }
11
12  export default Contact;
```

App.js

```
// import { createBrowserRouter,RouterProvider } from "react-router-dom"; //This is wrong import statement
import { createBrowserRouter,RouterProvider } from "react-router"; //This is correct import statement
import About from "./components/About";
import Contact from "./components/Contact";
```

```
const appRouter=createBrowserRouter([
  {
    path:"/",
    element:<AppLayOut/>
  },{
    path:"/about",
    element:<About/>
  },{
    path:"/contact",
    element:<Contact/>
  }
]);

const root = ReactDOM.createRoot(document.getElementById("root"));

root.render(<RouterProvider  router={appRouter} />);
```

O/p=>
http://localhost:1234/contact

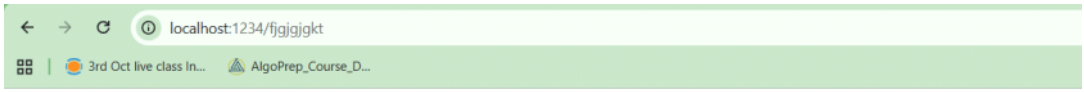
Contact Us Page

This is Namaste React Live Course 2023

Contact No :: 1234567890

Our Email is :: pr1234@gamil.com

-> if we type anything other than url routing, we get deafult Error/Exception page which is provided by react-router.



Unexpected Application Error!

404 Not Found

Hey developer

You can provide a way better UX than this when your app throws errors by providing your own `ErrorBoundary` or `errorElement` prop on your route.

Lets create our Error Custom page.

Create one Error Component.

Error.js

```
src > components > Error.js > ...
1   const Error =()=>{
2
3       return(
4           <div>
5               <h1>Oops !! </h1>
6               <h2>Something went wrong.</h2>
7               <h3>Please try again later.</h3>
8           </div>
9       );
10  }
11
12  export default Error;
```

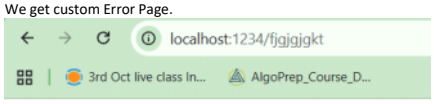
Import Error component into mail js file.
we have errorElement in object, if something wrong happens, we need to load Error component.

```
import Error from "./components/Error";
```

```
28    * path: URL path for the route.
29    * element: React component to be rendered for that route.
30    * Here, we have defined two routes:
31    * 1) "/" route which renders AppLayOut component.
32    * 2) "/about" route which renders About component.
33    * 3) "/contact" route which renders Contact component.
34    * errorElement: React component to be rendered in case of an error.
35    */
36   const appRouter=createBrowserRouter([
37     {
38       path:"/",
39       element:<AppLayOut/>,
40       errorElement: <Error/>
41     },{
42       path:"/about",
43       element:<About/>
44     },{
45       path:"/contact",
46       element:<Contact/>
47     }
48   ]);
```

URL=>http://localhost:1234/fjgjgjgkt

We get custom Error Page.



Oops !!

Something went wrong.

Please try again later.

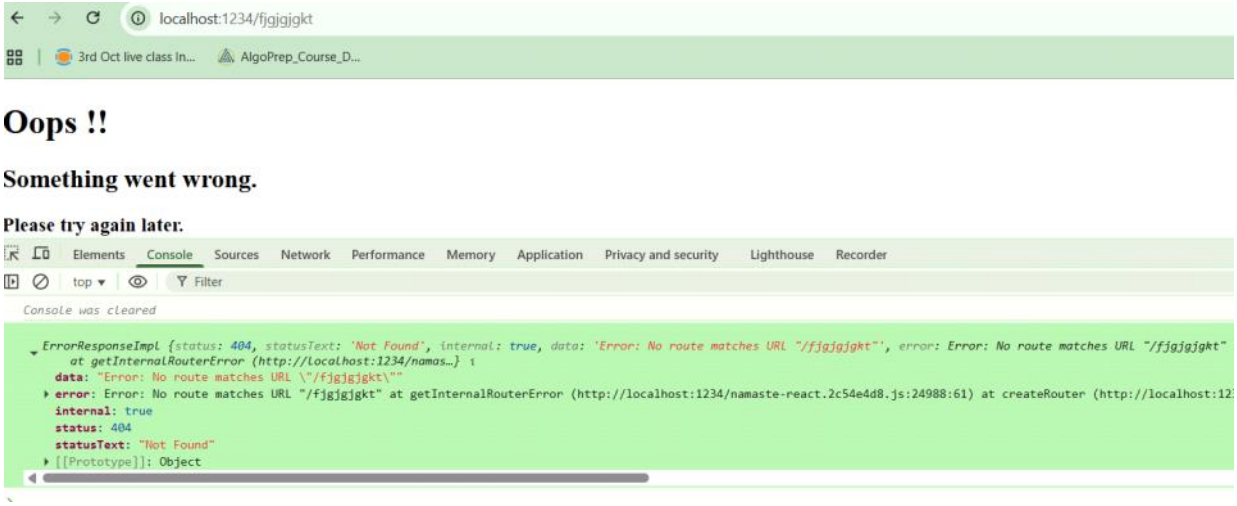
Note::- React-router gives as Hook which is function at end of the day. But hook has specific purpose.
That hook is UseRouteError.

```
import { useRouteError } from "react-router";
```

How to identify Hook ?
->Whenever function starting with useXXXX this is hook.

->UseRouteError this gives more information about the error.
Instead of just showing display error page.

```jsx
import { useRouteError } from "react-router";

const Error =()=>{

    const err = useRouteError();
    console.log(err);
    return(
        <div>
            <h1>Oops !! </h1>
            <h2>Something went wrong.</h2>
            <h3>Please try again later.</h3>
        </div>
    );
}

export default Error;
```



Now, we can use this error data to show on Ui page.

See how I show error on UI page as below:-

```jsx
import { useRouteError } from "react-router";

const Error =()=>{

    const err = useRouteError();
    console.log(err);
    return(
        <div>
            <h1>Oops !! </h1>
            <h2>Something went wrong.</h2>
            <h3>Please try again later.</h3>
            <h3>{err.status + " : " + err.statusText}</h3>
        </div>
    );
}

export default Error;
```

Now will try to create children routes.
If I go to about Us page, I lose my header & footer component.
My requirement is always header & footer component must be on the stays whether we changing URL or not.
So that my about Us page load below between Header & Footer Component.
So, we will develop as our URL changes body will be changes only & Header & Footer stays there not change.

Episode 07 Part 04

==========

Children Routes inside our App

->when we on home page i.e. /, we want

```
const AppLayOut =()=>{

    console.log(<BodyComponent />);

  return (
  <div className="app">
    <HeaderComponent />
     <BodyComponent />
  </div>);
};
```

But when I was in about page i.e. /about, we want,

```
  const AppLayOut =()=>{

      console.log(<BodyComponent />);

    return (
    <div className="app">
      <HeaderComponent />
       | <About />
    </div>);
  };
```

We want our HeaderComponent should not be change.

To make This type of functionality, we need to create children routes.
Children Routes of AppLayout.
So, we write like this to create children of AppLayout.

```
const appRouter=createBrowserRouter([
  {
    path:"/",
    element:<AppLayOut/>,
    children:[
    {
      path:"/about",
       element:<About/>
    },{
      path:"/contact",
      element:<Contact/>
    }
    ],
    errorElement: <Error/>
  }
]);
```

Note:- To write any comment in JSX, we use {/** write your comment *\}

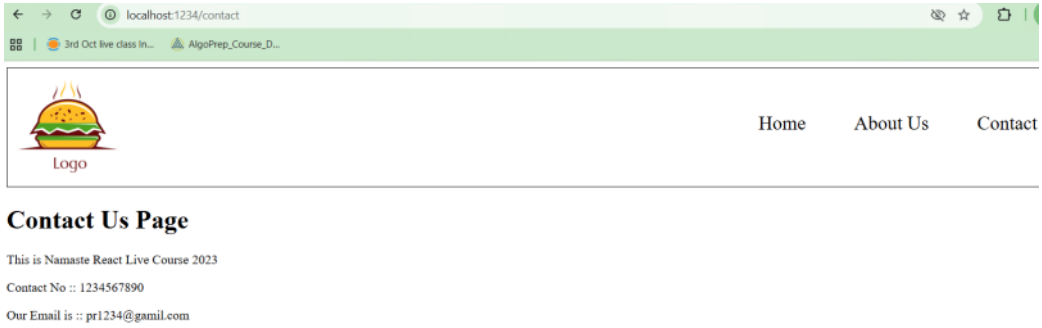
```
      console.log(<BodyComponent />);

    return (
    <div className="app">
      <HeaderComponent />
      {/** if path is = /  render BodyComponent */}
       <BodyComponent />
       {/** if path is = /about  render About Component */}
       <About/>
       {/** if path is = /contact  then render Contact Component */}
       <Contact />
    </div>);
  };
```

How to load Component based on path?
->react Router DOM comes to rescue.
it gives us something like Outlet.
Outlet will load the component based on the path we provide in the URL.
We need to import Outlet Component.

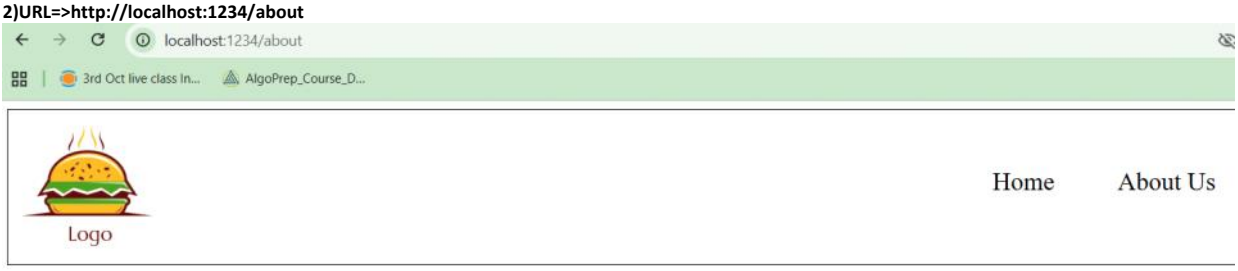
```
import { createBrowserRouter,RouterProvider ,Outlet} from "react-router"; //This is correct import statement
```

This Outlet will be filled with the children according to the path on what page we are.
If path is / it will load BodyComponent.
If path is /about it will load About Component.

1)URL=> http://localhost:1234/contact



2)URL=>http://localhost:1234/about



About Us Page

This is Namaste React Live Course 2023

We are learning React JS

Our Mentor is Akshay Sir

Note:- Outlet Component will be replaced by Another Component based on URL.

Code we write to create Childer Routes

App.js

```
import { createBrowserRouter,RouterProvider ,Outlet} from "react-router"; //This is correct import statement
```

```
const AppLayOut =()=>{

    console.log(<BodyComponent />);

  return (
  <div className="app">
    <HeaderComponent />
  {/** How to load Component based on path, react Router DOM comes to rescue.
   * it gives us something like Outlet.
   * Outlet will load the component based on the path we provide in the URL. */}
    <Outlet />
  </div>);
};
```

```
  /**
   * createBrowserRouter is used to create a router object.
   * We can define multiple routes inside the array we pass to createBrowserRouter.
   * Each route is an object with path and element properties.
   * path: URL path for the route.
   * element: React component to be rendered for that route.
   * Here, we have defined two routes:
   * 1) "/" route which renders AppLayOut component.
   * 2) "/about" route which renders About component.
   * 3) "/contact" route which renders Contact component.
   * errorElement: React component to be rendered in case of an error.
   * children: Array of child routes for nested routing.
   * Each child route is also an object with path and element properties.
   */
```

```
  */
  const appRouter=createBrowserRouter([
    {
      path:"/",
      element:<AppLayOut/>,
      children:[
        {
        path:"/",
         element:<BodyComponent/>
      },

      {
        path:"/about",
         element:<About/>
      },{
        path:"/contact",
        element:<Contact/>
      }
      ],
      errorElement: <Error/>
    }
  ]);
```

->Now develop when click on about Us , it must redirect to /about url.

1)Using anchor tag

->

```
          <li>Home</li>
          <li> <a href="/about">About Us</a></li>
          <li>Contact Us</li>
          <li>Cart</li>
          <button  className="login" onClick={
            ()=>{
```

It is working as well.

Note:- Whenever we use React & we want to route to some other page, never ever use anchor tag.
Q)Why we not use anchor tag?
->if we use, our whole page got refreshed when we click on anchor tag.
I don't want to refresh my whole page, but still I still move to new route.
i.e. we can navigate to different page without reloading the whole page.

So, instead of **href**, we will use **Link** component. Link comes from react-router dom.
Link is a superpower which is given by react-router dom.

```
import { Link } from "react-router";
```


```
import { Link } from "react-router";
```


```
    <ul>
      <li> <Link to="/">Home</Link></li>
      <li> <Link to="/about">About Us</Link></li>
      <li>
        <Link to="/contact">Contact Us</Link>
      </li>
      <li>Cart</li>
      <button  className="login" onClick={
        ()=>{
            btnNameReact==="Login"? setBtnNameReact("LogOut"):  setBtnNameReact("Login");
        }}>{btnNameReact}</button>
    </ul>
</div>
```

->in React, our page is not refresh or reloading the page, it is just changing the components so, React is called as Single Page application.

->There are two types of Routing we have in web application,
1)Client Side Routing.
2)Server Side Routing.

Server Side Routing mean if we click on aboutUs page, it make a network call & get aboutUs.html page from server & show on UI.
Client Side Routing mean we are not making any network call when we moving from one page to another.
It just Components getting interchange.

Episode 07 Part 05
==========
Now we will make one feature, after we click on any restaurant Card, we must be visit that restaurant menu to see more details.
Different pages for different restaurant.

Dynamic Routing

->for every restaurant, we have dynamic routing in swiggy website.
& we get restaurant menu details of that restaurant.

For this we create one RestaurantMenuCard component.
Lets create static component with static data.
RestaurantMenuCard .js

src > components > JS RestaurantMenuCard.js > default

```
const RestaurantMenuCard =()=>{
    return (
        <div className="menu">
        <h1>Name of The Restaurant</h1>
        <h2>Menu</h2>
        <ul>
            <li>Pizza</li>
            <li>Burger</li>
            <li>Pasta</li>
            <li>French Fries</li>
        </ul>
        </div>
    );
};

export default RestaurantMenuCard;
```

Use it our App.js main file where we define our routing.

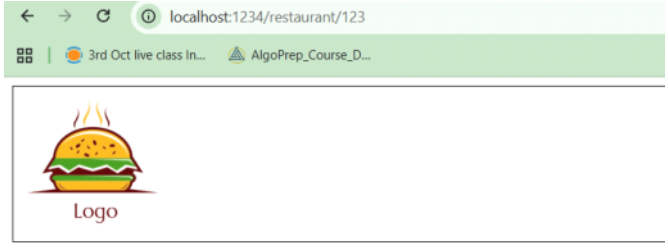
```
import RestaurantMenuCard from "./components/RestaurantMenuCard";
```

```
const appRouter=createBrowserRouter([
    {
      path:"/",
      element:<AppLayOut/>,
      children:[
        {
        path:"/",
         element:<BodyComponent/>
      },

      {
        path:"/about",
         element:<About/>
      },{
        path:"/contact",
        element:<Contact/>
      },{
        path:"/restaurant/:resId",  // resId is a dynamic parameter. to write we use /: before the parameter name.
        element:<RestaurantMenuCard/>
      }
      ],
      errorElement: <Error/>
    }
  ]);
```

->Below we specify the dynamic routing.

```
{
      path:"/restaurant/:resId",  // resId is a dynamic parameter. to write we use /: before the parameter name.
      element:<RestaurantMenuCard/>
    }
```

->as Soon as we hit,Url-> http://localhost:1234/restaurant/123 with any resId like 123 we specify here,
We get RestrtaurantMenuCard component on UI.



Name of The Restaurant

Menu

- Pizza
- Burger
- Pasta
- French Fries

->so, resId is dynamic id, accroding to the resId our data on RestaurantMenuCard also gets changes.
->Now gets the dynamically data from swiggy API. So, instead of static data get data from Swiggy API.
->When the ResataurantMenuCard Component gets loads, load the data, make an API call.
We will use the hook which is useEffect Hook.

Info about UseEffect:-

useEffect(()=>{},[]); //1st argument is function, 2nd argument is optional

we don't have dependency array here, so useEffect will be called every time after the component is rendered.
If we want to call useEffect only once after the first render, we have to provide empty dependency array.
i do not want to call useEffect every time after the component is rendered.
So, we have to provide empty dependency array.

->To fetch data from url, create one fetchMenu function & fetch api,

```
const fetchMenu= async()=>{
   const data = await fetch();  // provide API URL to fetch menu data inside the fetch()
}
```

->sometime we get this type of error,
If we write code as

```
*/
const [resInfo,setResInfo]=useState(null);
        useEffect(
         ()=>{
             console.log("Calling fetch menu");
          fetchMenu();
         },[]);

         const fetchMenu = async ()=>{
             console.log("fetchin api");

             const data = await fetch(
        "https://www.zomato.com/webroutes/getPage?page_url=/nagpur/krishnum-restaurant-mominpura/order&locat
                        );
         const jsonData= await data.json();
         console.log(jsonData);

          setResInfo(jsonData.page_data);
         };

         // console.log(resInfo.sections.SECTION_BASIC_INFO);

          const[itemsCards]=resInfo?.order?.menuList?.menus[0]?.categories?.items;

          console.log(itemsCards);



    return   (resInfo===null)  ?
            (<ShimmerComponet/>

                ):(
        <div className="menu">
          <h1>{resInfo.sections.SECTION_BASIC_INFO.name}</h1>
         <h3>{resInfo.sections.SECTION_BASIC_INFO.cuisine_string}</h3>
```

Bcz we get above error, bcz intially resInfo is null as we set in useState(). So we get above error.
To prevent from this, write this as ,

```
const [resInfo,setResInfo]=useState(null);
        useEffect(
         ()=>{
             console.log("Calling fetch menu");
          fetchMenu();
         },[]);

         const fetchMenu = async ()=>{
             console.log("fetchin api");

             const data = await fetch(
        "https://www.zomato.com/webroutes/getPage?page_url=/nagpur/krishnum-restaurant-mominpura/order&location=&
                        );
         const jsonData= await data.json();
         console.log(jsonData);

          setResInfo(jsonData.page_data);
         };
         if(resInfo===null)return <ShimmerComponet/>;

         // console.log(resInfo.sections.SECTION_BASIC_INFO);

          const[itemsCards]=resInfo?.order?.menuList?.menus[0]?.categories?.items;

          console.log(itemsCards);
```

->
How to use map effectively to understnd how to write.
Let say we have array of object. Ex:- [{},{},{}]
This is basic,

```
    <ul>
        <li>{itemCards[0].item.name}</li>
        <li>{itemCards[1].item.name}</li>
    </ul>
```

Now apply for map, on itemCards array,

```
<ul>

    { itemCards.map((res) =>
        (<li>{res.item.name}</li>))}

</ul>
```

-> to get dynamic data from URL, we have useParams component in React-Router.
Using this,we get resId from URL.

Ex:-
1)
If URL is http://localhost:1234/restaurant/3300071
Then,
const [resInfo,setResInfo]=useState(null);
const params= useParams(); // o/p=> {resId: '3300071'}
params is an object which contains all the dynamic parameters in the URL.

The path we map in App.js file is

```
},{
  path:"/restaurant/:resId",  // resId is a dynamic parameter. to write we use /: before the parameter name.
  element:<RestaurantMenuCard/>
}
```

2)if we change the pathName to resIdNew, we get as,

```
},{
  path:"/restaurant/:resIdNew",  // resIdNew is a dynamic parameter. to write we use /: before the parameter name.
  element:<RestaurantMenuCard/>
}
```

If URL is http://localhost:1234/restaurant/3300071
Then,
const [resInfo,setResInfo]=useState(null);
const params= useParams(); // o/p=> {resIdNew: '3300071'}
params is an object which contains all the dynamic parameters in the URL.

3)if our URL is http://localhost:1234/restaurant/nagpur/veeraswami-sadar i.e. after /restaurant everything is dynamic.
Then,
To get values from it,
We use below in App.js

```
},{
  path:"/restaurant/:resId/:resName",  // resId is a dynamic parameter. to write we use /: before the parameter name.
  element:<RestaurantMenuCard/>
}
```

We get value from the URL using useParams()

```
17   const {resId,resName}= useParams();  //destructuring to get resId,resName directly.
18   console.log(resId,resName); //resId=nagpur resName=veeraswami-sadar
19
```

4)this us our URL which fetch data from server,
https://www.zomato.com/webroutes/getPage?page_url=/nagpur/krishnum-restaurant-mominpura/order&location=&isMobile=0
As highlighted in red color part is dyanamic,so we break this URL into parts.
Constant part we keep it in our Constant.js file.
Like

```
export const MENU_API_URL="https://www.zomato.com/webroutes/getPage?page_url=/";
```

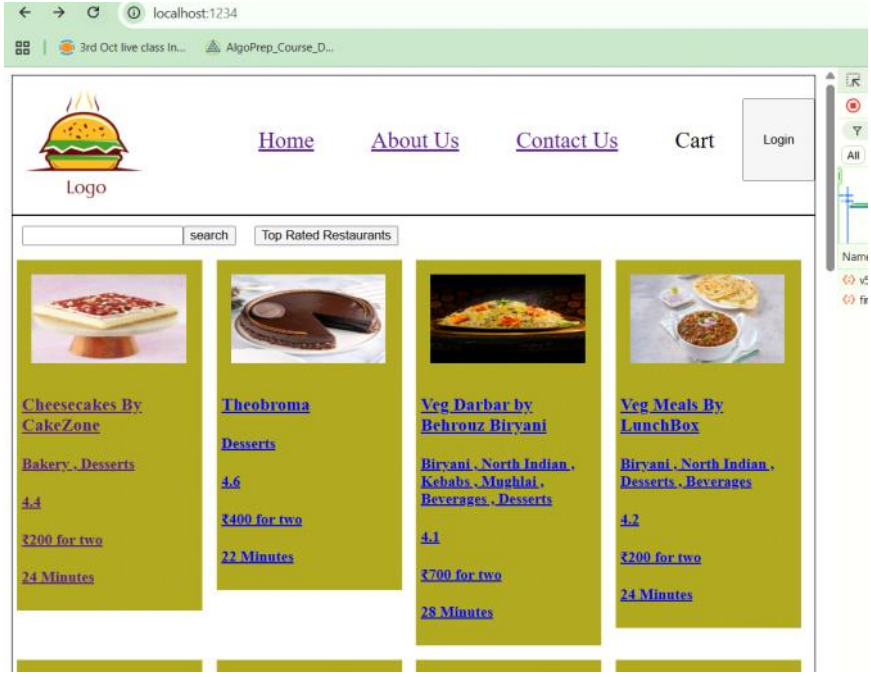
```
const data = await fetch(
MENU_API_URL+resId+"/"+resName+"/order&location=&isMobile=0"
);
```

->Now we need to build, Once we click on Restaurant Card, it must go to that restaurant Menu component.
For with willl we use anchor tag ? No.
We will use Link Component.

```
( <Link
  key={restaurant.info.id}
 to={"/restaurant/"+restaurant.info.id+"/"+restaurant.info.name}  >
<RestaturantCardComponent  resData={restaurant} />
</Link>
```

Now here we use key inside the Link tag Link is parent & RestaurantCardComponent id child of Link.
So better to keep it in parent Component.

->My api is not working bcz I use for menuOfEach restaurant I use Zomato api.
While for to get listof restaurant I use swiggy api.



Q)What is Link Component ?
->Link is special component which is given by react-router. Behind the scenes link is using the anchor tag.
So, due to this we see on HTML anchor tag as below.

```
<div className="nav-items">
    <ul>
      <li> <Link to="/">Home</Link></li>
      <li> <Link to="/about">About Us</Link></li>
      <li>
        <Link to="/contact">Contact Us</Link>
      </li>
      <li>Cart</li>
      <button  className="login" onClick={
        ()=>{
          btnNameReact==="Login"? setBtnNameReact("LogOut"):  setBtnNameReact("Login");
        }}>{btnNameReact}</button>
    </ul>
</div>
```

```
<div class="nav-items">
  <ul> flex
    <li>
      <a href="/" data-discover="true">Home</a>
    </li>
    <li> == $0
      <a href="/about" data-discover="true">About Us</a>
    </li>
    <li> ... </li>
```

->Link is wrapper over anchor tag.
So, we don't have refresh the page using Link tag.
Link is not understand by browser. Given by react-router.
Link internally converts into anchor tag to understand to browser.

++
EPISODE 8: Let's get classy
++

classBased Components in React

In this lecture, we study about class-based components.
class-based components is an older way to create a component in react.
Untill Episode 7, we study functional based component only.
functional based component is an new way of writing code.

Q)Why are we studying then now if not used class-based compoenets ?
->1)Asked in interviews.
2)used in legacy project.

Episode 08 Part 01
==========
In About component, we will create one component to show info of team members.
Data of team members we will fetch from GitHub api.

First we will create using Functional component then we create using Class-based component.

Way 1)using Functional component

1)User.js component

```
src > components > User.js > default
const User =()=>{

    return(
        <div className="user-card">
            <h2>Name: Akshay</h2>
            <h3>Location: Nagpur</h3>
            <h4>Contact: @akshaymarch7</h4>

        </div>
    );
}

export default User;
```

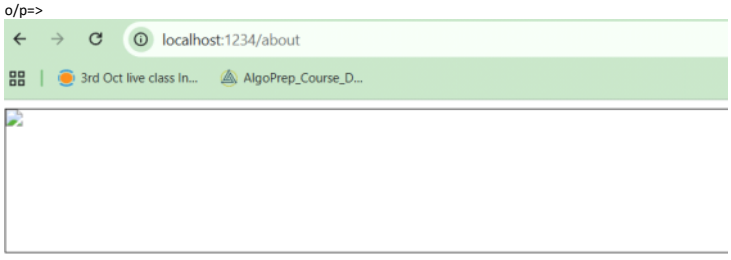
2)Place user component inside About component.

```
src > components > About.js > About
import User from "./User";

const About=()=>{
    return(
        <div>
            <h1>About Us Page</h1>
            <p>This is Namaste React Live Course 2023</p>
            <p>We are learning React JS</p>
            <User/>
        </div>
    )
}

export default About;
```

3)give some css.

```
.user-card{
    padding: 10px;
    border: 1px solid black;
}
```

o/p=>



About Us Page

This is Namaste React Live Course 2023

We are learning React JS

Name: Akshay

Location: Nagpur

Contact: @akshaymarch7

WAY 2) class-based Component.
React funtional component is at the end is an normal javascript function.
Similarily React Class-based component is at the end is an Normal Javascript class.

Using normal javascript :-
Class UserClass{
}

Using React class based component:-
class UserClass extends React.Component {

```
    render(){
      //Write JSX code & return it.
    }
}
```

->we need to extends our custom class using React.Component so that React know this is a class based component.
->we have render() inside the class, which return piece of JSX which will be displayed on UI.

So ,React functional component is an function which return a some piece of JSX.
While React Class-based component is an class which has render() which return a some piece of JSX.
-> React.Component is an class which is given by react & custom class is inheriting some properties from it.

Which is comes from react library.

```
import React from "react";
```

WAY 2) class-based Component.

1)create UserClass.js class component.

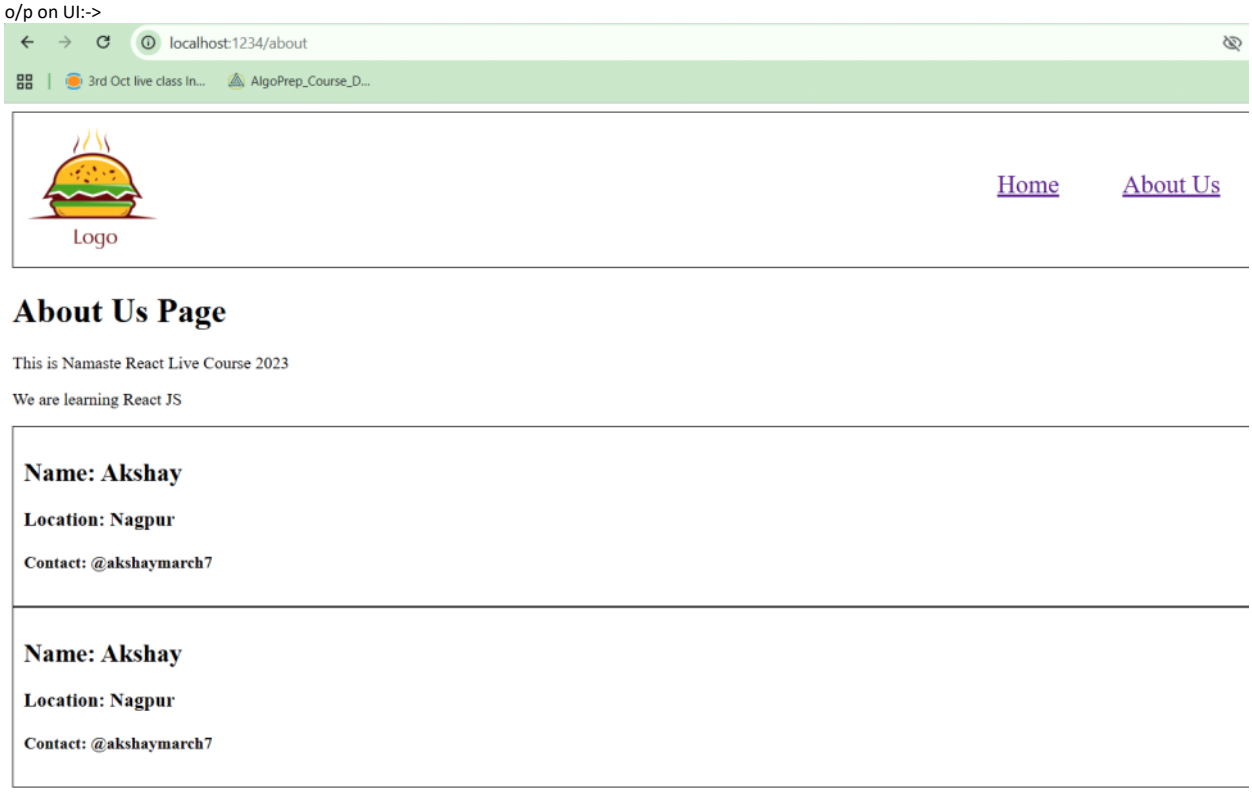
```
src > components > UserClass.js > default
import React from "react";
class UserClass extends React.Component {

    render(){
      return(
       <div className="user-card">
           <h2>Name: Akshay</h2>
           <h3>Location: Nagpur</h3>
           <h4>Contact: @akshaymarch7</h4>

       </div>
    );
    }

}

export default UserClass;
```

2)used this UserClass component in About Component.

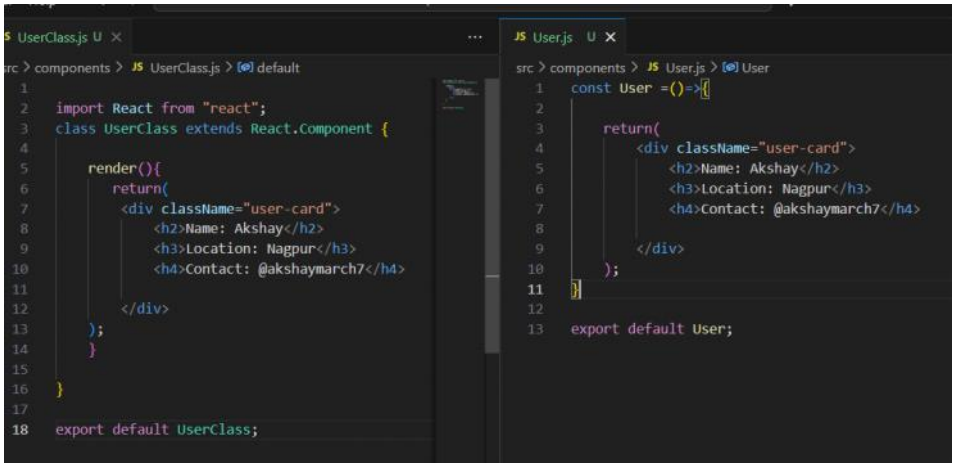
```
User.js U    About.js M
components > About.js > ...
import User from "./User";
import Userclass from "./UserClass";
const About=()=>{
    return(
        <div>
            <h1>About Us Page</h1>
            <p>This is Namaste React Live Course 2023</p>
            <p>We are learning React JS</p>
             <User/>
             <Userclass/>
        </div>
    )
}

export default About;
```

3)

o/p on UI:->



Comparison of Classbased & Functional Component code.



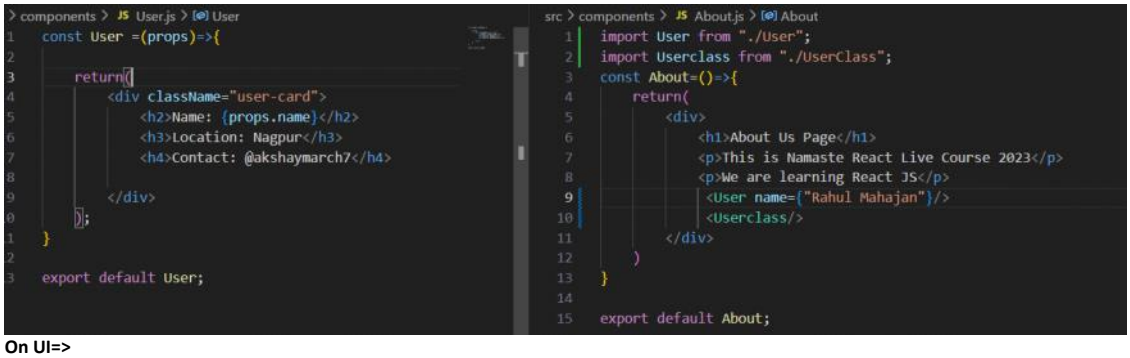
UserClass.js U

```
import React from "react";
class UserClass extends React.Component {

    render(){
       return(
         <div className="user-card">
             <h2>Name: Akshay</h2>
             <h3>Location: Nagpur</h3>
             <h4>Contact: @akshaymarch7</h4>

         </div>
    );
    }

}

export default UserClass;
```

User.js U

```
const User =()=>{

    return(
        <div className="user-card">
            <h2>Name: Akshay</h2>
            <h3>Location: Nagpur</h3>
            <h4>Contact: @akshaymarch7</h4>

        </div>
    );
}

export default User;
```

Q)How we pass argument in functional component ?



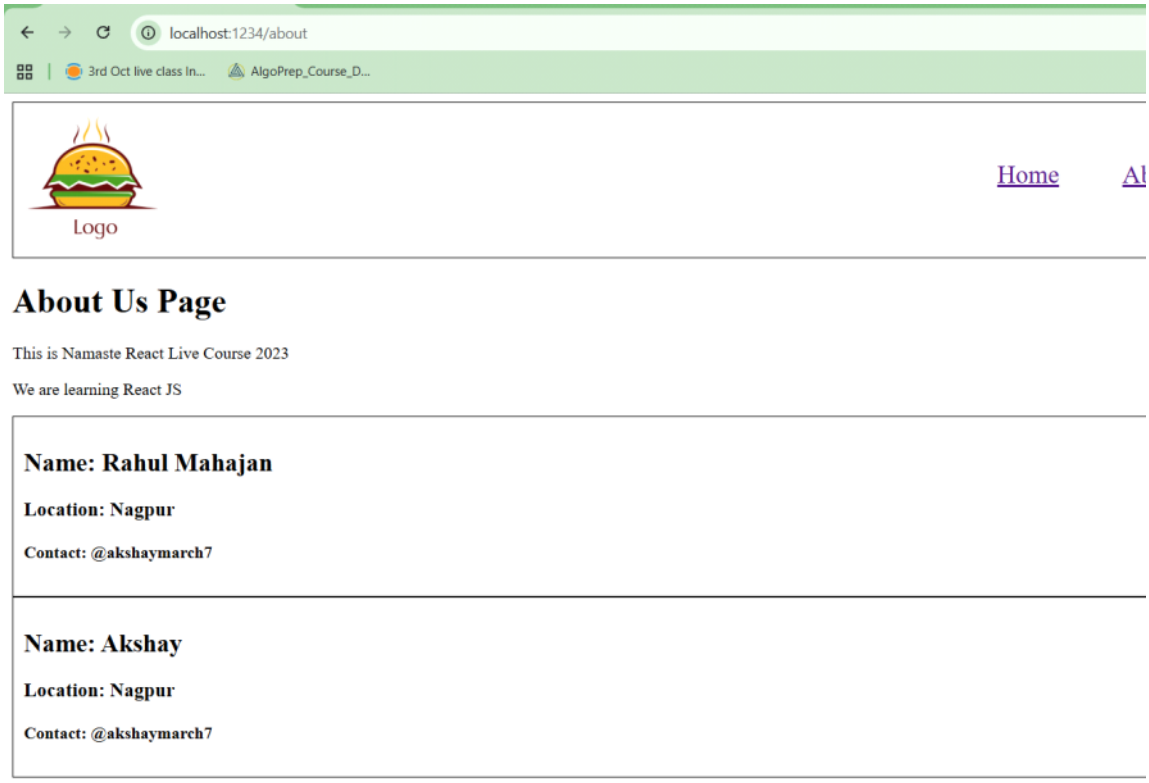
User.js

```
const User =(props)=>{

    return(
        <div className="user-card">
            <h2>Name: {props.name}</h2>
            <h3>Location: Nagpur</h3>
            <h4>Contact: @akshaymarch7</h4>

        </div>
    );
}

export default User;
```

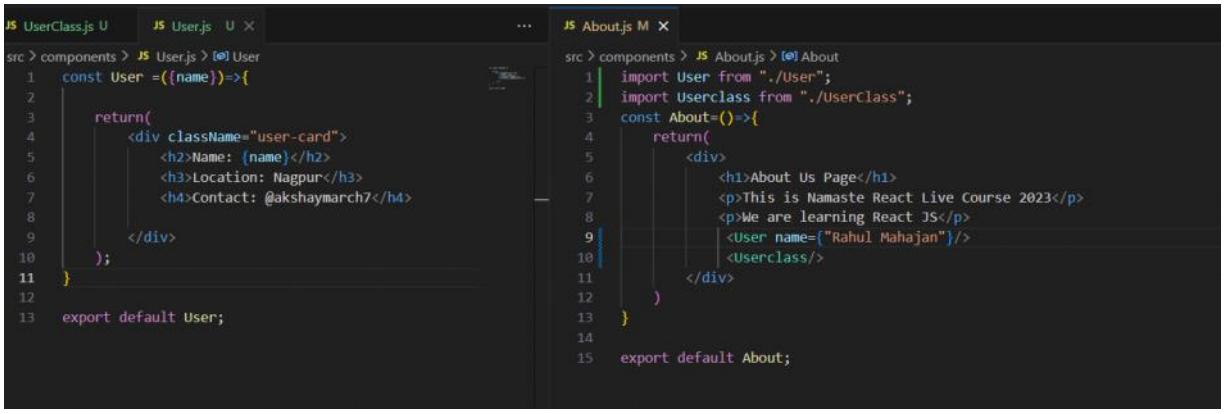
About.js

```
import User from "./User";
import Userclass from "./UserClass";
const About=()=>{
    return(
        <div>
            <h1>About Us Page</h1>
            <p>This is Namaste React Live Course 2023</p>
            <p>We are learning React JS</p>
            <User name={"Rahul Mahajan"}/>
            <Userclass/>
        </div>
    )
}

export default About;
```

On UI=>

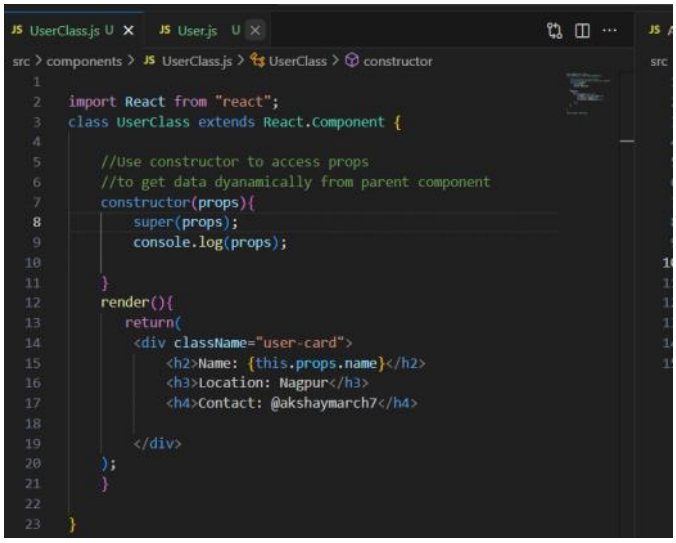


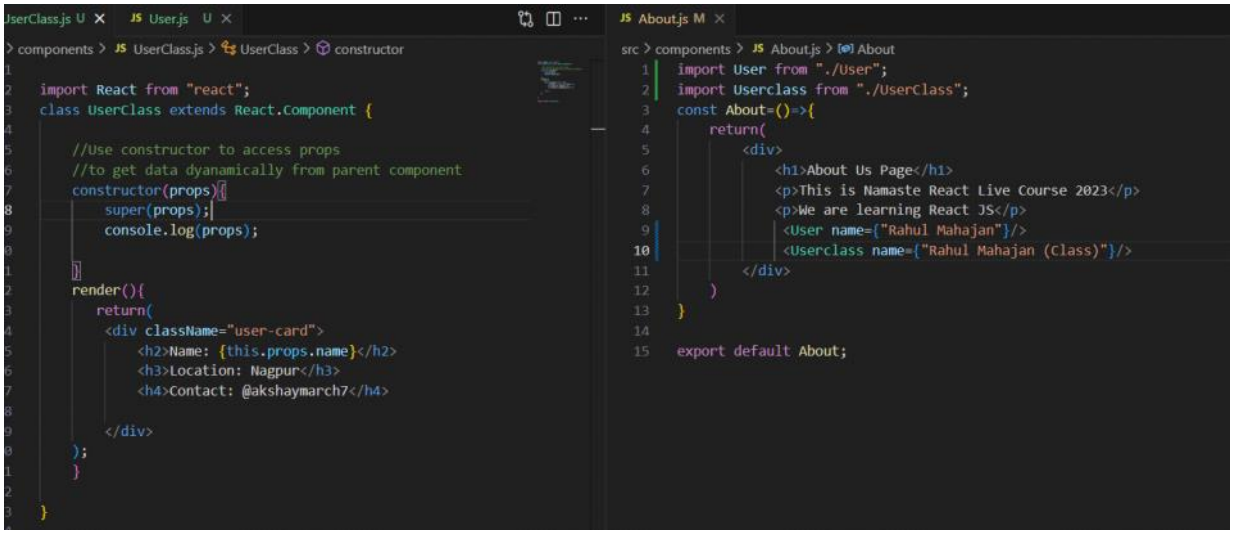
We can write code using destructing way:-
Like

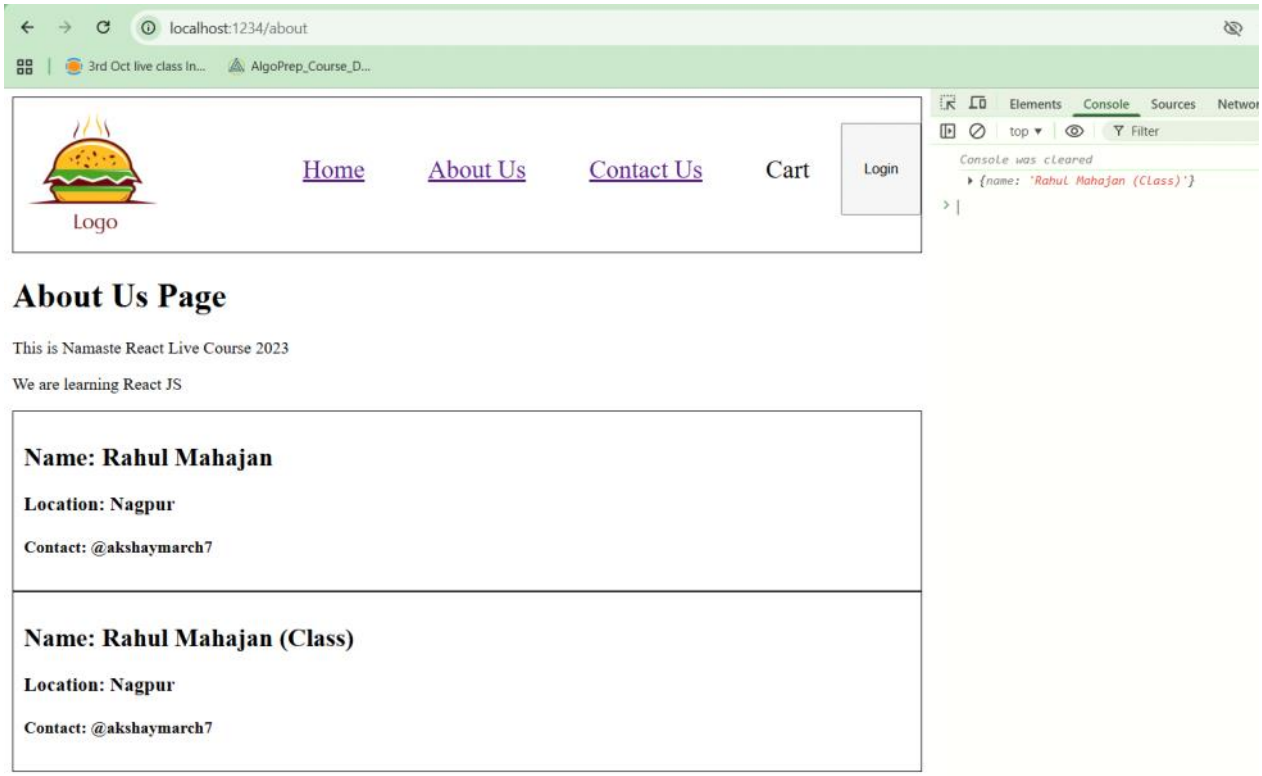


Now, to get Data dynamically from the Parent Compoent in Class-based component as below:-
1)we use param Constructor. Aslo we need to write super() ,method as well.
//Use constructor to access props
//to get data dyanamically from parent component

```
constructor(props){
super(props);
console.log(props);
}
```

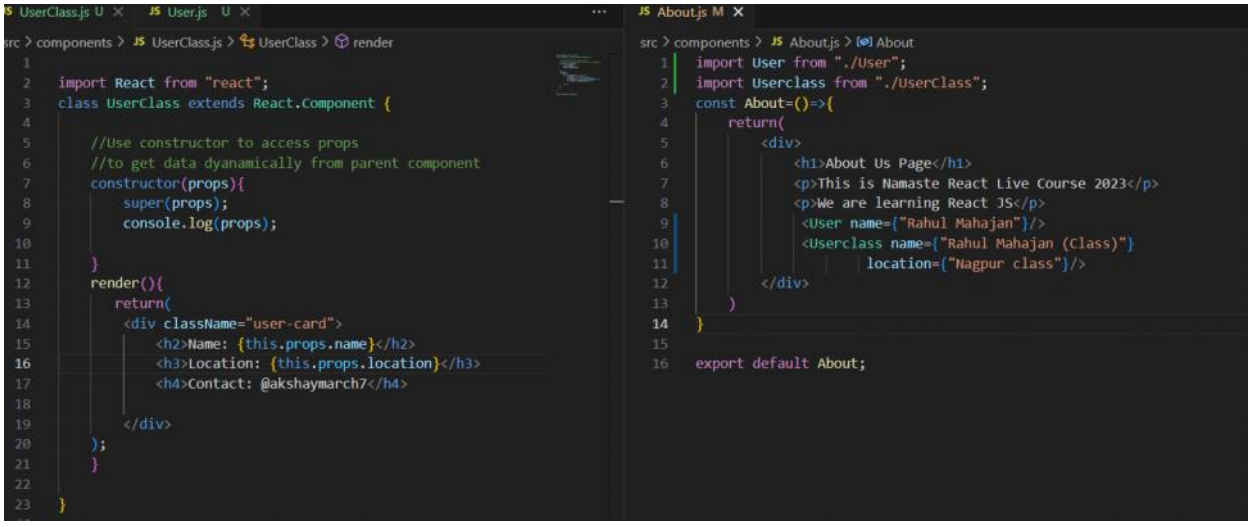




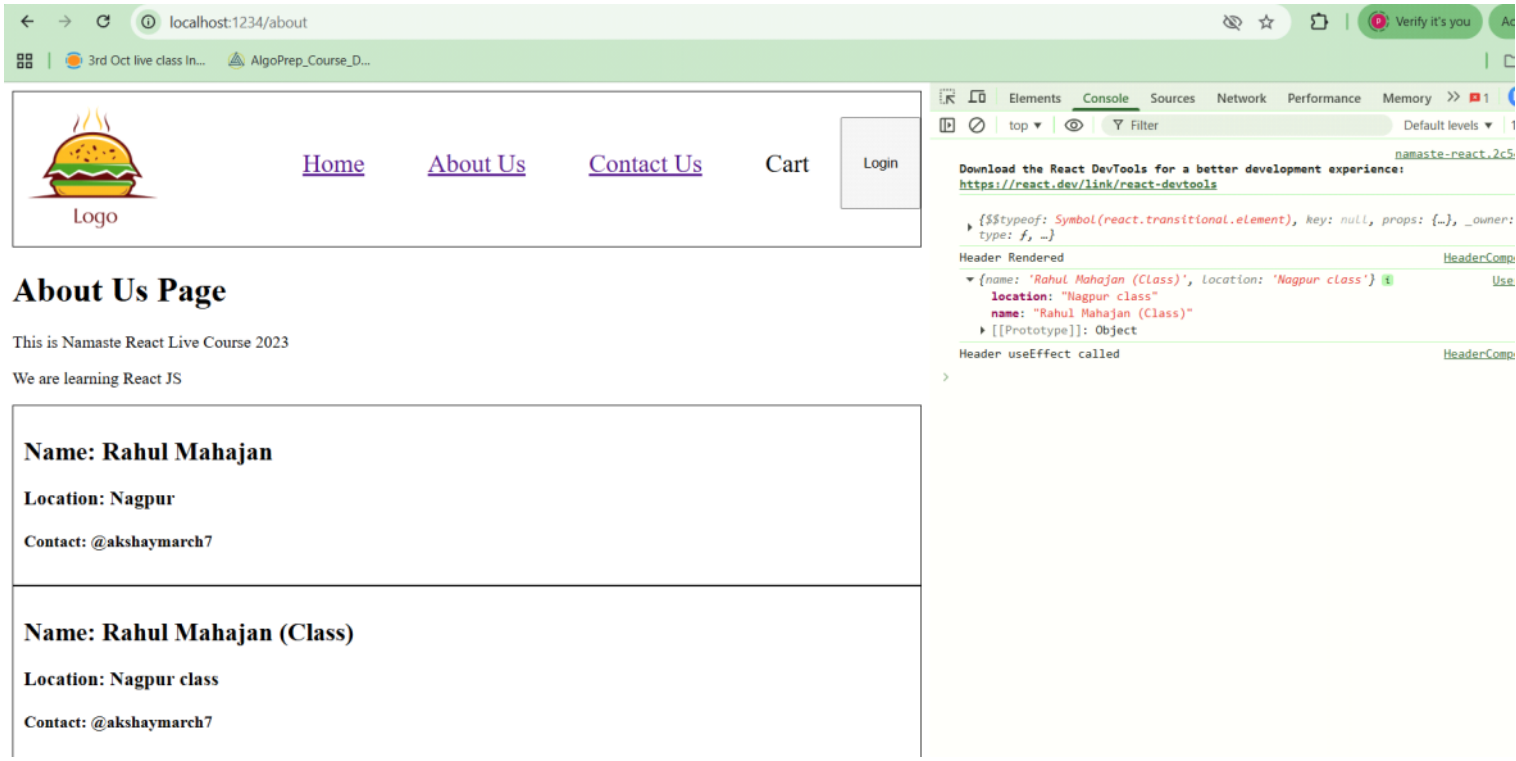


->Whenever new instance of class is create, constructor gets called & props extracted which we pass in ParentComponent while calling that classBased Component.

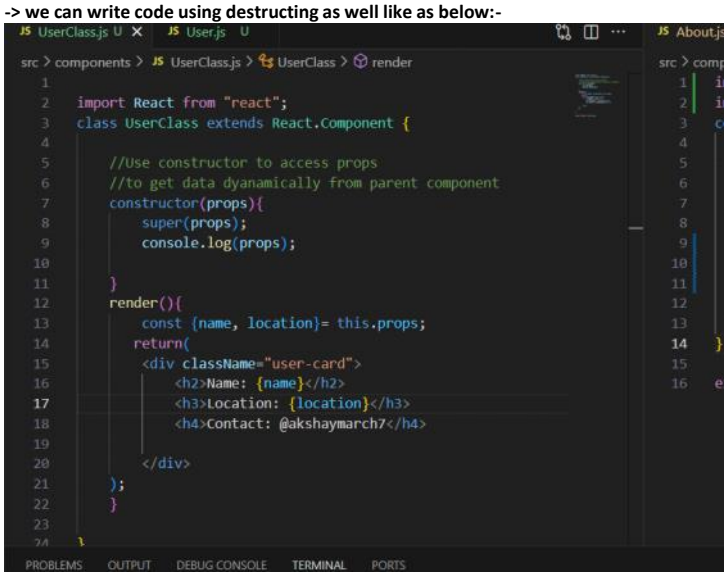
->We can pass n no.of props as well.



o/p=>



-> we can write code using destructing as well like as below:-



Episode 08 Part 02

==========

Now, we will see how to create State variables, local variables inside class based component.

->To Create a state variable inside the functional component we use Hook which is known as UseState.

Using functional Component:-

On UI::-

Count = 0

Name: Rahul Mahajan

Location: Nagpur

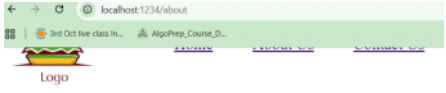
Contact: @akshaymarch7

Using Class-based component:-
Earlier there is no concept of hook is there.
Let's see how they created class-based state variables.
->Whenever we say invoking or calling functional component means we are loading or mounting that function onto that webpage.
->Whenever we say creating an instance of a class-based component means we are loading the class-based component onto that webpage.
->whenever we create an instance of a class, constructor will be called.
This is best place to receive props & this is best place to create state-variables.
We use this.state to create state variables.
state is an big whole object which contains state variables.
State is an reserve word in react.

UserClass.js U

components > UserClass.js > UserClass

```
import React from "react";
class UserClass extends React.Component {

    //Use constructor to access props
    //to get data dyanamically from parent component
    constructor(props){
        super(props);

        this.state={
            count:0,
        };

    }
    render(){
        const {name, location}= this.props;
       return(
        <div className="user-card">
            <h1>Count in class = {this.state.count}</h1>
            <h2>Name: {name}</h2>
            <h3>Location: {location}</h3>
            <h4>Contact: @akshaymarch7</h4>

        </div>
    );
    }

}

export default UserClass;
```



About Us Page

This is Namaste React Live Course 2023

We are learning React JS

Count = 0

Name: Rahul Mahajan

Location: Nagpur

Contact: @akshaymarch7

Count in class = 0

Name: Rahul Mahajan (Class)

Location: Nagpur class

Contact: @akshaymarch7

->we can write in destructing way as well like,

```
components > UserClass.js > UserClass > render
import React from "react";
class UserClass extends React.Component {

    //Use constructor to access props
    //to get data dyanamically from parent component
    constructor(props){
        super(props);

        this.state={
            count:0,
        };

    }
    render(){
        const {name, location}= this.props;
        const {count}= this.state;

       return(
        <div className="user-card">
            <h1>Count in class = {count}</h1>
            <h2>Name: {name}</h2>
            <h3>Location: {location}</h3>
            <h4>Contact: @akshaymarch7</h4>

        </div>
    );
    }

}

export default UserClass;
```

Comparison on how to create state variable in class-based & functional component:-

```
UserClass.js U
src > components > UserClass.js > UserClass > render
  import React from "react";
  class UserClass extends React.Component {

      //Use constructor to access props
      //to get data dyanamically from parent component
      constructor(props){
          super(props);

          this.state={
              count:0,
          };

      }
      render(){
          const {name, location}= this.props;
          const {count}= this.state;

         return(
          <div className="user-card">
              <h1>Count in class = {count}</h1>
              <h2>Name: {name}</h2>
              <h3>Location: {location}</h3>
              <h4>Contact: @akshaymarch7</h4>

          </div>
      );
      }

  }

  export default UserClass;
```

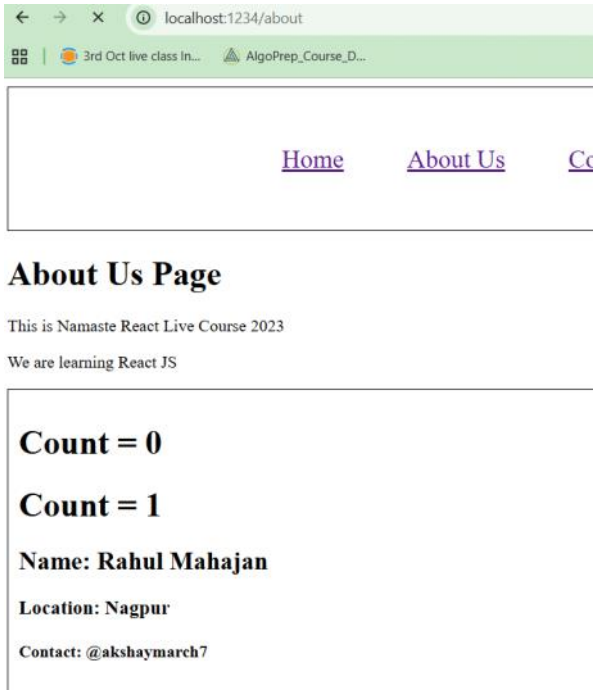
```
User.js U
src > components > User.js > User
  import {useState} from "react";


  const User =({name})=>{

      const [count] =useState(0);

      return(
          <div className="user-card">
              <h1>Count = {count}</h1>
              <h2>Name: {name}</h2>
              <h3>Location: Nagpur</h3>
              <h4>Contact: @akshaymarch7</h4>

          </div>
      );
  }

  export default User;
```

->Let's say I want to create Two state variable.

1)Using react Component:-
->we need to create 2 state variables using 2UseState() hook separately.

```
src > components > User.js > User
  import {useState} from "react";


  const User =({name})=>{

      const [count] =useState(0); //0 is initial value
      const [count2] =useState(1); //1 is initial value

      return(
          <div className="user-card">
              <h1>Count = {count}</h1>
              <h1>Count = {count2}</h1>
              <h2>Name: {name}</h2>
              <h3>Location: Nagpur</h3>
              <h4>Contact: @akshaymarch7</h4>

          </div>
      );
  }

  export default User;
```

UI=>

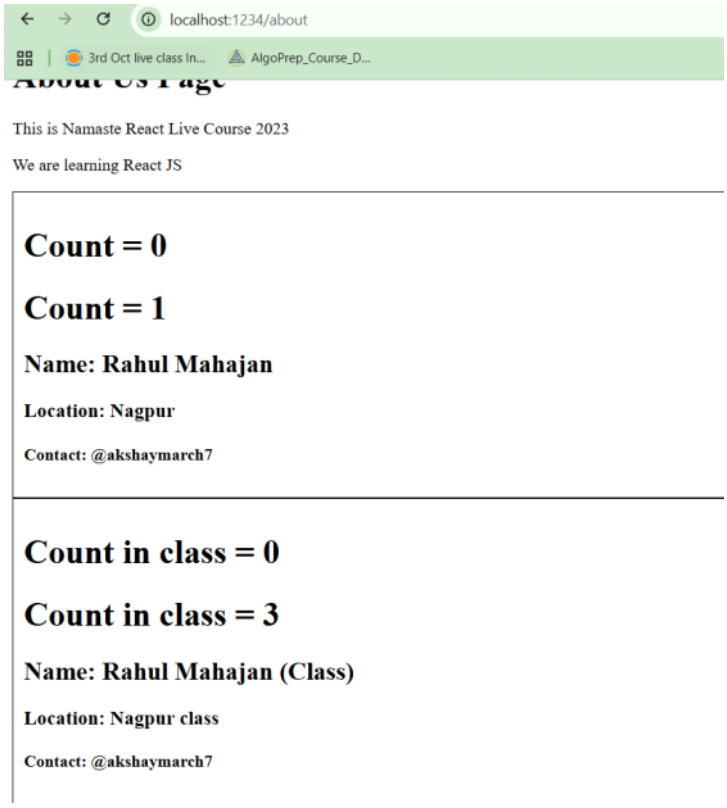


2)Using class-based Component:-

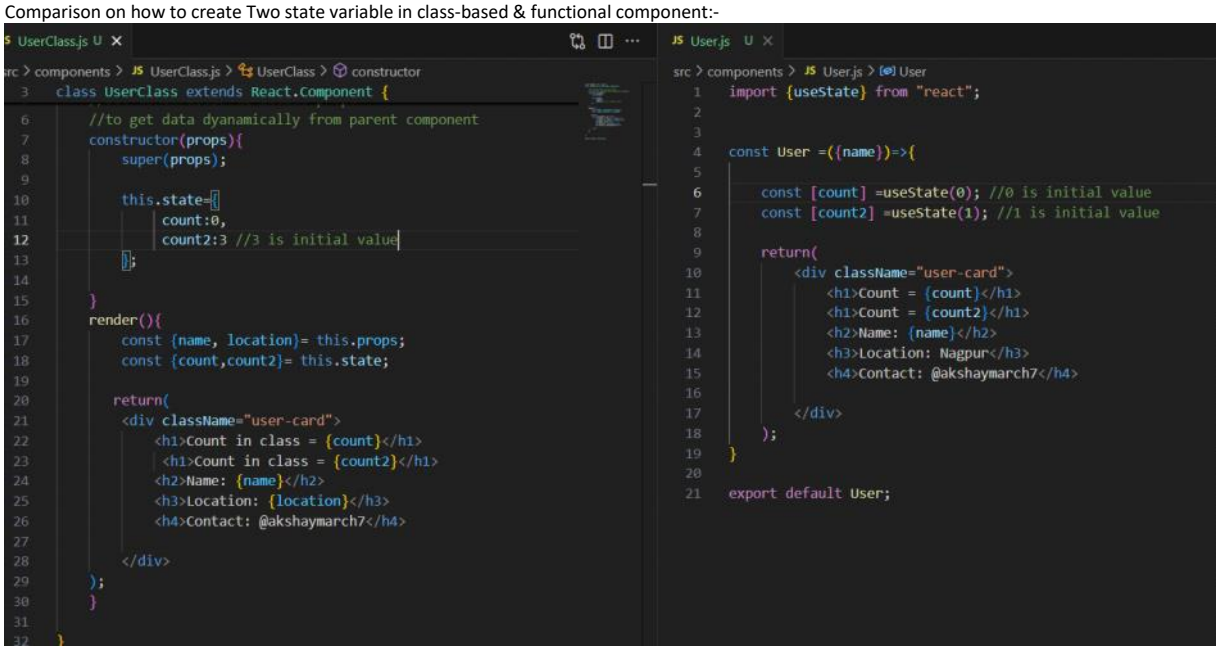
->state variable of react which is whole object which contain all state variables.

```
import React from "react";
class UserClass extends React.Component {

    //Use constructor to access props
    //to get data dyanamically from parent component
    constructor(props){
        super(props);

        this.state={
            count:0,
            count2:3 //3 is initial value
        };

    }
    render(){
        const {name, location}= this.props;
        const {count,count2}= this.state;

       return(
        <div className="user-card">
            <h1>Count in class = {count}</h1>
            <h1>Count in class = {count2}</h1>
            <h2>Name: {name}</h2>
            <h3>Location: {location}</h3>
            <h4>Contact: @akshaymarch7</h4>

        </div>
    );
```

On UI=>



Comparison on how to create Two state variable in class-based & functional component:-



```
UserClass.js U
src > components > UserClass.js > UserClass > constructor
class UserClass extends React.Component {

    //to get data dyanamically from parent component
    constructor(props){
        super(props);

        this.state={
            count:0,
            count2:3 //3 is initial value
        };

    }
    render(){
        const {name, location}= this.props;
        const {count,count2}= this.state;

       return(
        <div className="user-card">
            <h1>Count in class = {count}</h1>
            <h1>Count in class = {count2}</h1>
            <h2>Name: {name}</h2>
            <h3>Location: {location}</h3>
            <h4>Contact: @akshaymarch7</h4>

        </div>
    );
    }
}
```

```
User.js U
src > components > User.js > User
import {useState} from "react";


const User =({name})=>{

    const [count] =useState(0); //0 is initial value
    const [count2] =useState(1); //1 is initial value

    return(
        <div className="user-card">
            <h1>Count = {count}</h1>
            <h1>Count = {count2}</h1>
            <h2>Name: {name}</h2>
            <h3>Location: Nagpur</h3>
            <h4>Contact: @akshaymarch7</h4>

        </div>
    );
}

export default User;
```

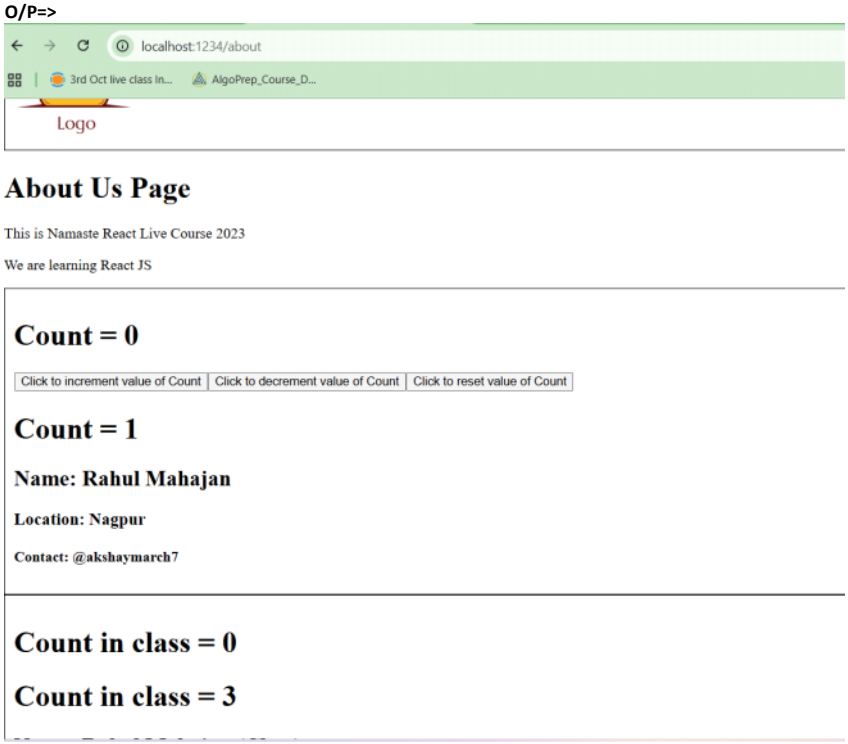
Now, we will see how to update the state variables.

Episode 08 Part 03
==========
Now, we will see how to update the state variables.

Way 1:- Using Functional component to update state variables.
->we use one setFunction with useState variable.

```jsx
const User =({name})=>{

    const [count, setCount] =useState(0); //0 is initial value
    const [count2] =useState(1); //1 is initial value

    const incrementCount =()=>{
        setCount(count+1);
    }
     const decrementCount =()=>{
        setCount(count-1);
    }
      const resetCount =()=>{
        setCount(0);
    }

    return(
        <div className="user-card">
            <h1>Count = {count}</h1>
            <button onClick={incrementCount}>Click to increment value of Count</button>
             <button onClick={decrementCount}>Click to decrement value of Count</button>
              <button onClick={resetCount}>Click to reset value of Count</button>
            <h1>Count = {count2}</h1>
            <h2>Name: {name}</h2>
            <h3>Location: Nagpur</h3>
            <h4>Contact: @akshaymarch7</h4>

        </div>
    );
```

O/P=>



Way 2:- Using class-based components to update state variable values.
NOTE:: ->we can not update state variables directly.
this.state.count= this.state.count+1; //this is wrong way to update state, it will not update value.

```jsx
this.state={
    count:0,
    count2:2,
    count3:3
};
```

```jsx
<button onClick={()=>{
//   this.state.count= this.state.count+1; //this is wrong way to update state
   this.setState({
       count: this.state.count +1,
       count2: this.state.count2 +1
   }) //correct way to update state
}}>Click to increment value of Count</button>
```

->If we have n no.of state variables inside the state variables of React object.
& I want to update only some of them state variables, we can do that using this.setState() method as well.
As soon as we update the value, our page will rerendered again & again.
So, it will not update the count of count3 state variable.

```jsx
import React from "react";
class UserClass extends React.Component {

    //Use constructor to access props
    //to get data dyanamically from parent component
    constructor(props){
        super(props);

        this.state={
            count:0,
            count2:2,
            count3:3
        };

    }
    render(){
        const {name, location}= this.props;
        const {count}= this.state;

       return(
        <div className="user-card">
            <h1>Count in class = {count}</h1>
              <button onClick={()=>{
            //   this.state.count= this.state.count+1; //this is wrong way to update state
               this.setState({
                   count: this.state.count +1,
                   count2: this.state.count2 +1
               }) //correct way to update state
              }}>Click to increment value of Count</button>
            <h2>Name: {name}</h2>
            <h3>Location: {location}</h3>
            <h4>Contact: @akshaymarch7</h4>

        </div>
    );
    }
```

O/P=>

Count in class = 9

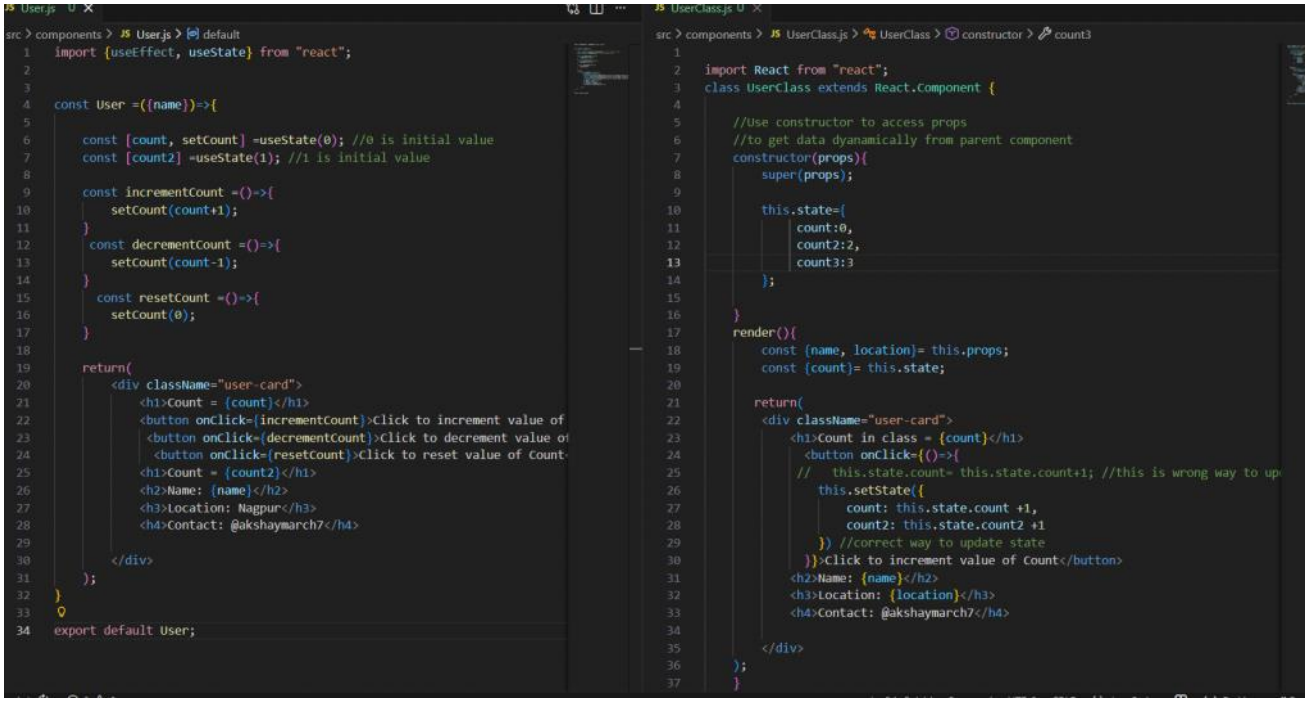
Click to increment value of Count

Name: Rahul Mahajan (Class)

Location: Nagpur class

Contact: @akshaymarch7

Comparsion of React functional & class based component to update state variables.



Episode 08 Part 04
==========

LifeCycle of React Class-Based Component

Loading the component on webpage means mounting the component on web page.

Let say,
Our parent Component is About which is an functional component.
Its child UserClass is class based component.
When About Component is loaded on web page,

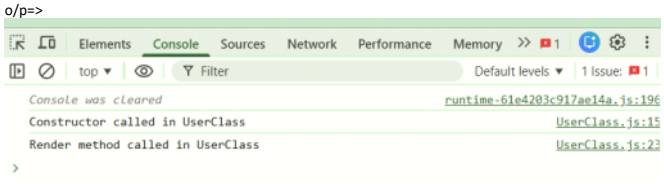
About component start executing the JSX code,
It see UserClass component which is class based component, as soon as UserClass loads now a new instance of this UserClass is created.
Once the class is instantiated, then constructor is called.
Once is constructor is called then the render() method is called.

About.js(functional component)

```
import User from "./User";
import Userclass from "./UserClass";
const About=()=>{
    return(
        <div>
            <h1>About Us Page</h1>
            <p>This is Namaste React Live Course 2023</p>
            <p>We are learning React JS</p>
             <User name={"Rahul Mahajan"}/>
             <Userclass name={"Rahul Mahajan (Class)"}
                    location={"Nagpur class"}/>
        </div>
    )
}

export default About;
```

```
class UserClass extends React.Component {
    //Use constructor to access props
    //to get data dyanamically from parent component
    constructor(props){
        super(props);

        this.state={
            count:0,
            count2:2,
        };

        console.log("Constructor called in UserClass");


    }
    render(){
        const {name, location}= this.props;
        const {count}= this.state;

        console.log("Render method called in UserClass");

       return(
        <div className="user-card">
            <h1>Count in class = {count}</h1>
              <button onClick={()=>{
            //   this.state.count= this.state.count+1; //this is wrong
                this.setState({
                    count: this.state.count +1,
                    count2: this.state.count2 +1
                }) //correct way to update state
              }}>Click to increment value of Count</button>
            <h2>Name: {name}</h2>
            <h3>Location: {location}</h3>
            <h4>Contact: @akshaymarch7</h4>
```

o/p=>



About.js(Class based component)

->Now lets make our About.js is a class-based component.

Some people like this way to import React.Component,

```
import React from "react";

class About extends React.Component {
```

Some people like this way by destructing.

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

class About extends Component {
```

->so, if parent is also class-based component & child is also class based component,
Then 1)parent constructor called
2)render() method of parent class called.
3)Child class constructor called.
4)render() method of child class called.

About.js(Class based component)

```
import User from "./User";
import Userclass from "./UserClass";
import {Component} from "react";

class About extends Component {

        constructor(props){
           super(props);
           console.log("About Constructor parent");

        }


    render(){
        console.log("About render method parent ");

        return(
            <div>
            <h1>About Us Page</h1>
            <p>This is Namaste React Live Course 2023</p>
            <p>We are learning React JS</p>
             <Userclass name={"Rahul Mahajan (Class)"}
                        location={"Nagpur class"}/>
        </div>
        );
    }
}
```

UserClass..**js(Class based component)**

```
UserClass.js U ×   About.js M
c > components > JS UserClass.js > UserClass > render
1
2   import React from "react";
3   class UserClass extends React.Component {
4
5       //Use constructor to access props
6       //to get data dyanamically from parent component
7       constructor(props){
8           super(props);
9
10          this.state={
11              count:0,
12              count2:2,
13          };
14
15          console.log("child constructor called in UserClass");
16
17
18      }
19      render(){
20          const {name, location}= this.props;
21          const {count}= this.state;
22
23          console.log("child Render method called in UserClass");
24
25         return(
26          <div className="user-card">
27              <h1>Count in class = {count}</h1>
```

o/p=>

```
About Constructor parent                                  About.js:9
About render method parent                                About.js:15
child constructor called in UserClass                     UserClass.js:15
child Render method called in UserClass                   UserClass.js:23
```

This is the lifecycle of classbased component.

->Class-based component has one more method i.e. componentDidMount() method.
First constructor is called then render() method is called.
Once constructor once this class based component mounted on to the DOM, then componentDidMount() method is called.
Ex:-

```
import React from "react";
class UserClass extends React.Component {

    //Use constructor to access props
    //to get data dyanamically from parent component
    constructor(props){
        super(props);

        this.state={
            count:0,
            count2:2,
        };

        console.log("child constructor called in UserClass");
    }

    componentDidMount(){
        console.log("child componentDidMount method called in UserClass");
    }
    render(){
        const {name, location}= this.props;
        const {count}= this.state;

        console.log("child Render method called in UserClass");
```

o/p=>

```
child constructor called in UserClass                     UserClass.js:15
child Render method called in UserClass                   UserClass.js:25
child componentDidMount method called in UserClass        UserClass.js:19
```

->**if we have componentDidMount() method is available in parent Class based component** also,
Then 1)first parent Constructor gets called,
Then 2)render() method gets called. While rendering it see child class based component,
So 3)child class constructor gets called then 4)render() method of child class called.
As Child Class has no any other subchild so 5) it will called componentDidMount() method of child class.
Once child class component mounted/loaded properly on web page,
Then 6)componentDidMount() method of parent class-based component will be called.

About.js(Parent class based component)

```
import Userclass from "./UserClass";
import {Component} from "react";

class About extends Component {

        constructor(props){
           super(props);
           console.log("About Constructor parent");

        }
      componentDidMount(){
           console.log("About componentDidMount parent ");
        }

   render(){
       console.log("About render method parent ");

       return(
           <div>
           <h1>About Us Page</h1>
           <p>This is Namaste React Live Course 2023</p>
           <p>We are learning React JS</p>
           <Userclass name={"Rahul Mahajan (Class)"}
                   location={"Nagpur class"}/>
       </div>
       );
```

UserClass.js(Childclass based component)

```
import React from "react";
class UserClass extends React.Component {

    //Use constructor to access props
    //to get data dyanamically from parent component
    constructor(props){
        super(props);

        this.state={
            count:0,
            count2:2,
        };

        console.log("child constructor called in UserClass");
    }

    componentDidMount(){
        console.log("child componentDidMount method called in UserClass");
    }
    render(){
        const {name, location}= this.props;
        const {count}= this.state;

        console.log("child Render method called in UserClass");

        return(
```

O/P=>

About Constructor parent	About.js:9
About render method parent	About.js:17
child constructor called in UserClass	UserClass.js:15
child Render method called in UserClass	UserClass.js:25
child componentDidMount method called in UserClass	UserClass.js:19
About componentDidMount parent	About.js:13

Q) Why componentDidMount() is given ?
->we do somethings once our component is mounted successfully on webpage.

Q)What are those things ?
->componentDidMount() is used to make an API calls in classbased component.

In React functional Component,
We use useEffect() method to make an api calls.
Ex:-
useEffect(()=>{
//API calls
}, []); //[] i.e. empty array which means it makes an api call once.

In React, first we load our component then we make api calls & fill that component with data.
We did not wait for api to return the data then only we will load the page, otherwise our component will not load & we need to wait untill we get data fully from api.

Episode 08 Part 05
==========
Q)What if our parent class based component has multiple child class component ?
->

About.js

```
class About extends Component {

        constructor(props){
           super(props);
           console.log("About Constructor parent");

        }
        componentDidMount(){
           console.log("About componentDidMount parent ");
        }

    render(){
        console.log("About render method parent ");

        return(
            <div>
            <h1>About Us Page</h1>
            <p>This is Namaste React Live Course 2023</p>
            <p>We are learning React JS</p>
            <Userclass name={"First (Class)"}  location={"Nagpur class"}/>
             <Userclass name={"Second (Class)"}  location={"Pune class"}/>
        </div>
```

UserClass.js

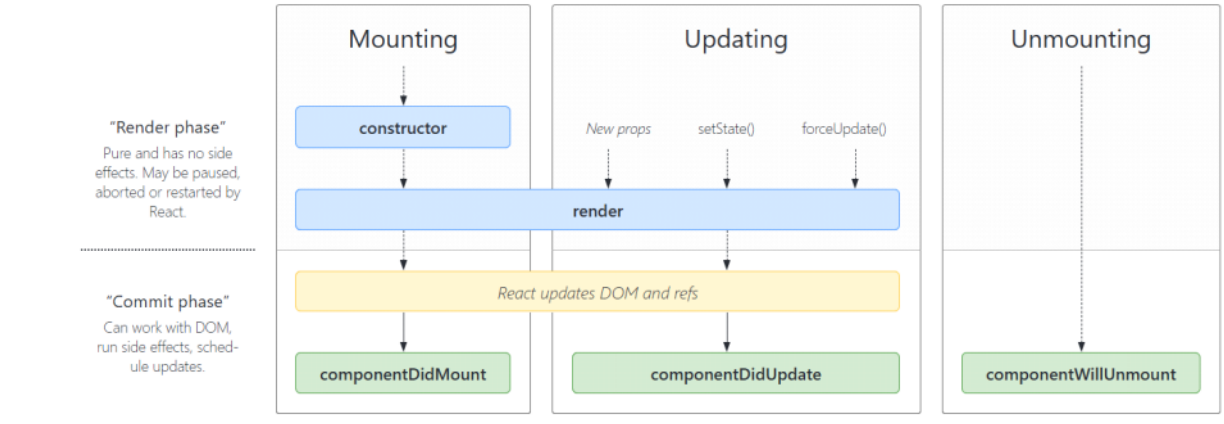
```
import React from "react";
class UserClass extends React.Component {

    //Use constructor to access props
    //to get data dyanamically from parent component
    constructor(props){
        super(props);

        this.state={
            count:0,
            count2:2,
        };
     console.log(this.props.name+" child constructor called in UserClass");
     }

     componentDidMount(){
        console.log(this.props.name+ " child componentDidMount method called in UserClass");

        //API call

     }
     render(){
        const {name, location}= this.props;
        const {count}= this.state;

        console.log(this.props.name+" child Render method called in UserClass");
```

O/P=>

```
About Constructor parent                                          About.js:9
About render method parent                                        About.js:17
First (Class) child constructor called in UserClass               UserClass.js:15
First (Class) child Render method called in UserClass             UserClass.js:28
Second (Class) child constructor called in UserClass              UserClass.js:15
Second (Class) child Render method called in UserClass            UserClass.js:28
First (Class) child componentDidMount method called in UserClass  UserClass.js:19
Second (Class) child componentDidMount method called in UserClass UserClass.js:19
About componentDidMount parent                                    About.js:13
```

Parent Component has two class based component child.
React will batch the render phase for these two child.
So, this child class based components render phase will happen then commit phase will happen together.
This is an optimization of React.
That is why we get output in above order.

Q)Why did the lifecycle method work like this ?
=>
Lets see react lifecycle diagram.
URL:- https://projects.wojtekmaj.pl/react-lifecycle-methods-diagram/

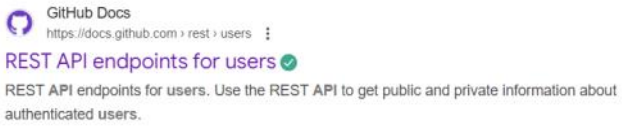


->React is fast because of Two phases.

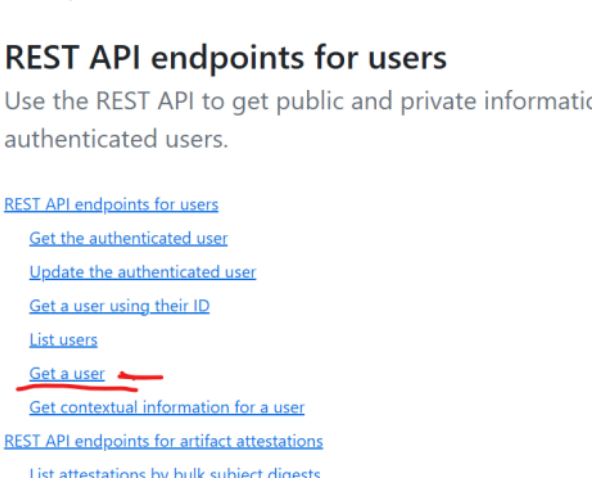
First phase is render phase.
Second phase is commit phase.
When the component is mounting, first constructor is called.
Then render() method is called.
This constructor & render is the Render Phase.
Then React actually updates the DOM & then componentDidMount is called.
That is why we make API calls inside the componentDidMount () method.

->When we are loading the Component, DOM manipulation is very very expensive it takes a lot of time.
So, React want to batch the render.
Render means reconciliation process & finding the diff between these two virtual DOM or Object.
Virtual DOM is fast because virtual DOM is just javascript which deals with objects.

Episode 08 Part 06
==========
Now we can make API calls inside the class-based component.
To get github user api.
Go to Google-> type github user api ->



Click on above link
->click on Get a User link.



->here is API url to get data based on username.

Ex:-
my username is praful-shahane.
URL::-> https://api.github.com/users/praful-shahane
We get following data.

```
{
    "login": "praful-shahane",
    "id": 154780164,
    "node_id": "U_kgDOCTnCBA",
    "avatar_url": "https://avatars.githubusercontent.com/u/154780164?v=4",
    "gravatar_id": "",
    "url": "https://api.github.com/users/praful-shahane",
    "html_url": "https://github.com/praful-shahane",
    "followers_url": "https://api.github.com/users/praful-shahane/followers",
    "following_url": "https://api.github.com/users/praful-shahane/following{/other_user}",
    "gists_url": "https://api.github.com/users/praful-shahane/gists{/gist_id}",
    "starred_url": "https://api.github.com/users/praful-shahane/starred{/owner}{/repo}",
    "subscriptions_url": "https://api.github.com/users/praful-shahane/subscriptions",
    "organizations_url": "https://api.github.com/users/praful-shahane/orgs",
    "repos_url": "https://api.github.com/users/praful-shahane/repos",
    "events_url": "https://api.github.com/users/praful-shahane/events{/privacy}",
    "received_events_url": "https://api.github.com/users/praful-shahane/received_events",
    "type": "User",
    "user_view_type": "public",
    "site_admin": false,
    "name": "Praful Shahane",
    "company": null,
    "blog": "",
    "location": null,
    "email": null,
    "hireable": null,
    "bio": "I am java developer working in TCS. i have 2+ yrs of experience.",
    "twitter_username": null,
    "public_repos": 13,
    "public_gists": 0,
    "followers": 1,
    "following": 0,
    "created_at": "2023-12-25T06:14:49Z",
    "updated_at": "2025-11-14T05:44:59Z"
}
```


->In Functional Component, to make API call we use async function.

```
useEffect(()=>{
    //API call
    getUserInfo();
},[]); //empty dependency array means it will run only once when component is mounted

async function getUserInfo() {
    const data= await fetch("https://api.github.com/users/akshaymarch7");

}
```

->In Class-based component, to make API call we make componentDidMount functions as async.

```
async componentDidMount(){
//   console.log(this.props.name+ " child componentDidMount method called in UserClass");

    //API call
    const data= await fetch("https://api.github.com/users/akshaymarch7");
    const json= await data.json();
    console.log(json);

}
```

Now use github api data in our code to show on UI.
To Update JSON data on web page, we use state variable for this.

Code::

```
import React from "react";
class UserClass extends React.Component {

    //Use constructor to access props
    //to get data dyanamically from parent component
    constructor(props){
        super(props);

        this.state={
            userInfo:{
                name: "Dummy Name",
                location: "Dummy Location",
                avatar_url:"https://dummy.com",
            }

        };
    // console.log(this.props.name+" child constructor called in UserClass");
     }
```

```
async componentDidMount(){
 //   console.log(this.props.name+ " child componentDidMount method called in UserClass");

    //API call
    const data= await fetch("https://api.github.com/users/akshaymarch7");
    const json= await data.json();
    console.log(json);

    //Once data is fetched from API, we will update the latest data into  the state variable
    this.setState({
        userInfo: json,
    });

 }
```

```
    render(){

        const {name, location,avatar_url} = this.state.userInfo;

    //    console.log(this.props.name+" child Render method called in UserClass");

       return(
        <div className="user-card">
            <img src={avatar_url} alt="User Avatar" />
            <h2>Name: {name}</h2>
            <h3>Location: {location}</h3>
            <h4>Contact: @akshaymarch7</h4>

        </div>
    );
    }

}

export default UserClass;
```

O/P on UI=>

We are learning React JS

Name: Akshay Saini

Location: India

Contact: @akshaymarch7

Now we will see how this LifeCycle Works ?

->

As soon as our UserClass is loaded, then constructor is called.

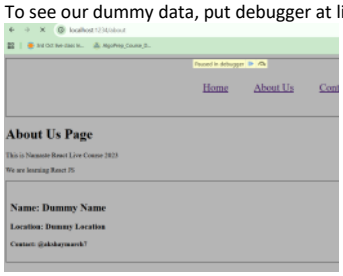
When the constructor was called, this state variables is created with some default value.

Then render() method gets called.

```
    render(){

        const {name, location,avatar_url} = this.state.userInfo;

    //    console.log(this.props.name+" child Render method called in UserClass");

       return(
        <div className="user-card">
            <img src={avatar_url} alt="User Avatar" />
            <h2>Name: {name}</h2>
            <h3>Location: {location}</h3>
            <h4>Contact: @akshaymarch7</h4>

        </div>
    );
    }

}
```

While executing the render method, this state Variables has a default value while first time mounting.

So our whole Component is rendered with default value on web page. This means react will update the DOM with dummy data.
To see our dummy data, put debugger at line no.39




```
22      async componentDidMount(){
23       //   console.log(this.props.name+ " child componentDidMount method called in UserClass");
24
25          //API call
26          const data= await fetch("https://api.github.com/users/akshaymarch7");
27          const json= await data.json();
28          console.log(json);
29
30          //Once data is fetched from API, we will update the latest data into  the state variable
31          this.setState({
32              userInfo: json,
33          });
34
35      }
```

Now our ComponentDidMount() method gets called & api call was made at line no.26,
Then setState will be called at line no.31.
When we call setState() our Updating cycle of React class-based lifecycle begins.
so, Mounting cycle finished when the Component rendered once/first time with dummy data we did not wait for API call to make.
Instead of dummy data, we can show shimmer UI for loading first time.
When we do setState Updating phase will start.
setState() updates the state variables. Then React trigger the rendered once again.
React triggers the render() method once again.
Constructor will call only once at Mounting only.
Constructor will not be call at Updating Phase.
This time state variables has changed with updated value.
Now in Updating phase, react will now update the DOM with new value by calculating diff between two virtual DOM.
Now we able to see data on UI.
Now our updated Cycle diagram said it calls componentDidUpdate() method gets called.

Flow of lifecycle of Class -based React component

```
/*
MOUNTING PHASE (Birth of Component)
==============================
Constructor(Dummy data)
Render(Dummy data)
        <HTML on UI with Dummy data>

 ComponentDidMount called
       make API Call

Updating Phase (Growth of Component)
==============================

 when this.SetState is called(set Latest data from API to state variable)
 Render (render happens with Latest data of state variable)
 <HTML on UI with Latest data>
componentDidUpdate called

*/
```

MOUNTING PHASE (Birth of Component)
==========
Constructor(Dummy data)
Render(Dummy data)
<HTML on UI with Dummy data>

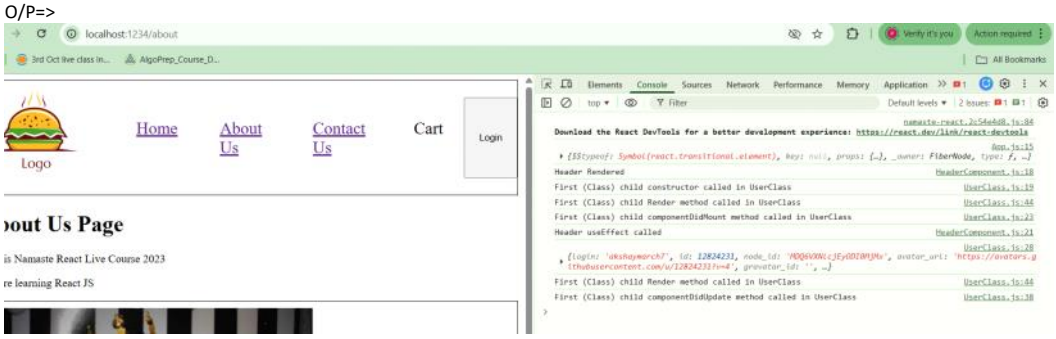
ComponentDidMount called
make API Call

Updating Phase (Growth of Component)
==========

when this.SetState is called(set Latest data from API to state variable)
Render (render happens with Latest data of state variable)
<HTML on UI with Latest data>
componentDidUpdate called

```
class UserClass extends React.Component {
    constructor(props){
    console.log(this.props.name+" child constructor called in UserClass");
     }

    async componentDidMount(){
      console.log(this.props.name+ " child componentDidMount method called in UserClass");

      //API call
      const data= await fetch("https://api.github.com/users/akshaymarch7");
      const json= await data.json();
      console.log(json);

      //Once data is fetched from API, we will update the latest data into  the state variable
      this.setState({
          userInfo: json,
      });

    }

    componentDidUpdate(){
      console.log(this.props.name+ " child componentDidUpdate method called in UserClass");
    }
    render(){

      const {name, location,avatar_url} = this.state.userInfo;

      console.log(this.props.name+" child Render method called in UserClass");

      return(
       <div className="user-card">
          <img src={avatar_url} alt="User Avatar" />
```

O/P=>

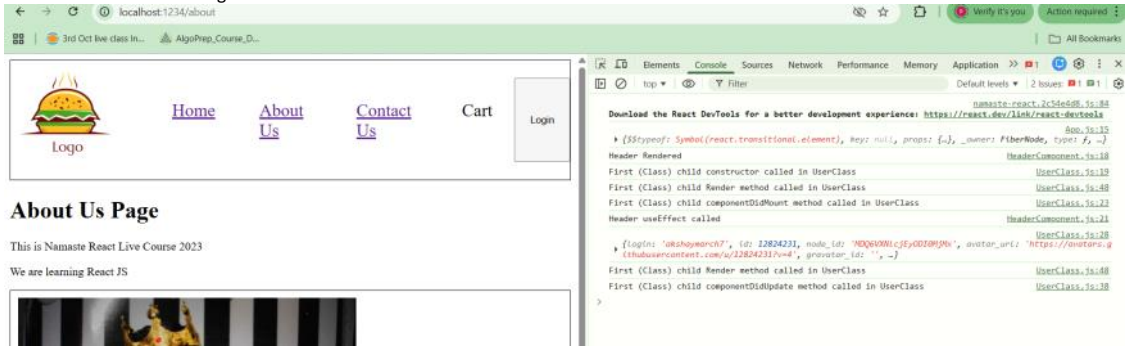


UnMounting Cycle(Removing from UI)

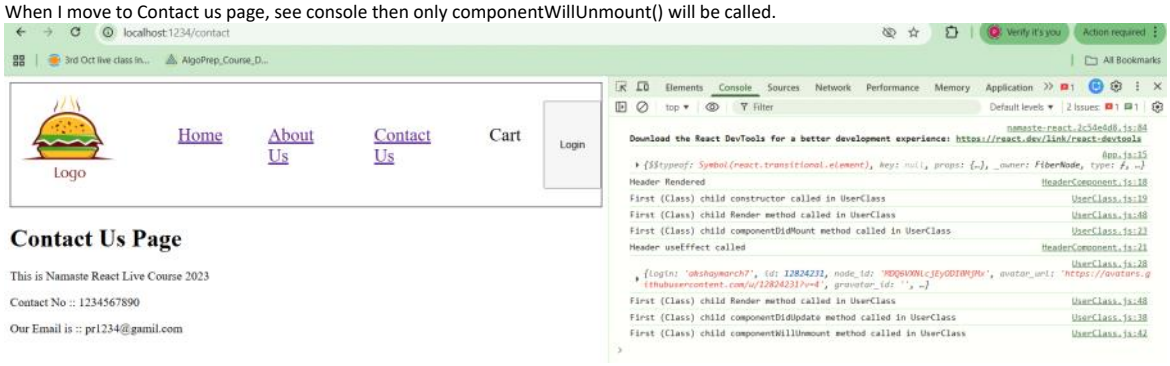
When our Component will be unmounted from web page, then componentWillUnmount() method will be called.
Ex:-
I am in contact Us page,
When I go to the Home page or any other page, this componentWillUnmount() method will be called.

```
class UserClass extends React.Component {
    async componentDidMount(){

      //API call
      const data= await fetch("https://api.github.com/users/akshaymarch7");
      const json= await data.json();
      console.log(json);

      //Once data is fetched from API, we will update the latest data into  the state variable
      this.setState({
          userInfo: json,
      });

    }

    componentDidUpdate(){
      console.log(this.props.name+ " child componentDidUpdate method called in UserClass");
    }

    componentWillUnmount(){
        console.log(this.props.name+ " child componentWillUnmount method called in UserClass");
    }
```

When I was in About us Page on UI. See console.



When I move to Contact us page, see console then only componentWillUnmount() will be called.



Earlier we write code using class-based component.

Episode 08 Part 07
==========

More About Class-based Component
==========

Disclaimer :- Never Ever Compare React Lifecycle methods with functional components.

useEffect in functional Component is not equivalent to componentDidMount() method in class based component.
useEffect is not using componentDidMount() method behind the scenes.
It's completely different.

Q) what happens if we not put []dependency array in useEffect useEffect(()=>{}) ?
-> for every render, our useEffect() function will be called.
Things are different in class based component.
In class based component, After First render componentDidMount() method is called.
& after subsequent render it is updated. It is not mounted.
There is diff between mount & update & unmount.

->if we put [] empty dependency array in useEffect it will be called Once in the initial render.
Ex:- useEffect(()=>{},[]) ; //render once
->When we write the Modern React code using functional component, they remove the concept of these lifecycle methods.
->if instead of empty array[] if we pass count which is an state variable into the useEffect(),
As our count changes, useEffect function will be called.
Ex:-
const [count, setCount] =useState(0); //0 is initial value

```
useEffect(()=>{
    //API call
    getUserInfo();
},[count]);
```

```
const [count, setCount] =useState(0); //0 is initial value
const [count2] =useState(1); //1 is initial value

useEffect(()=>{
      //API call
      getUserInfo();
},[count]); //count is dependency array. Whenver count changes,useEffect will be called
```

How we do something above like this, in class based component whenever count chnages ?
->
After each render, one lifecycle method is called that is componentDidUpdate() method.
It calls after every update.
Then we need to write the code in below way inside componentDidUpdate:-

```
componentDidUpdate(prevProps, prevState){
  console.log(this.props.name+ " child componentDidUpdate method called in UserClass");

  //if location has changed or if name   has changed then only make API call
  if(this.state.userInfo.location !== prevState.userInfo.location || this.props.userInfo.name !== prevState.userInfo.name){
   //API Call
   console.log("Location changed from "+ prevState.userInfo.location + " to "+ this.state.userInfo.location);
  }

}
```

So, we use array in useEffect because there can be multiple state variables where my if else might want to do in componentDidUpdate()

```
useEffect(()=>{
      //API call
      getUserInfo();
},[count, count2]);
```

->let way I want to do something when count changes & do some other thing when count2 changes,
How will we write in functional component ?
=>
We use two useEffect.

```
const [count, setCount] =useState(0); //0 is initial value
const [count2] =useState(1); //1 is initial value

useEffect(()=>{
      //API call
      getUserInfo();
},[count]); //count is dependency array. Whenver count changes,useEffect will be called

useEffect(()=>{
      //API call
      getUserInfo();
},[count2]); //count2 is dependency array. Whenver count2 changes,useEffect will be called
```

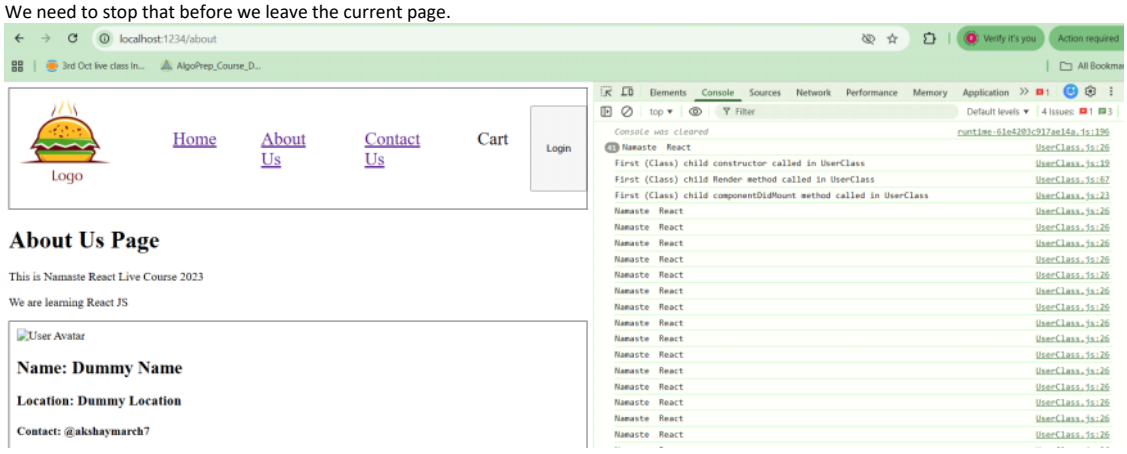
But to do same thing in class-based component, we need to write ifelse condition.

```
componentDidUpdate(prevProps, prevState){
  console.log(this.props.name+ " child componentDidUpdate method called in UserClass");

   //if location has changed or if name   has changed then only make API call
  if(this.state.userInfo.location !== prevState.userInfo.location || this.props.userInfo.name !== prevState.userInfo.name){
   //API Call
   console.log("Location changed from "+ prevState.userInfo.location + " to "+ this.state.userInfo.location);
  }

   if(this.state.count !== prevState.count || this.state.count2 !== prevState.count2){
   //API Call
   console.log("Count/Count2 changed");
  }

 }
```

If-else is not just a one line of code it can be 30-40 lines so much painful this when we write class-based component.
Q)When will we use componentWillUnmount() ?
-> for cleanup.
componentWillUnmount () is called when we leave that page.
When we are leaving the page a lot of things we need to cleanup some things.

```
async componentDidMount(){
  console.log(this.props.name+ " child componentDidMount method called in UserClass");

 //this.timer is an variable created to hold the interval ID
  this.timer=setInterval(()=>{
     console.log("Namaste  React");

  },1000);
```

we have create setInterval function which called after 1 seconds.
Our React is single page, even if we move from one page to another page, our setInterval still executing.
It is happening because we are not reloading the page, it is changing the component.
So it is cons of single page application.
It is leads to performance loss.
We need to stop that before we leave the current page.



So, we need to unmount this issue.

```
componentWillUnmount(){
    console.log(this.props.name+ " child componentWillUnmount method called in UserClass");

   //passing timer variable to stop the interval
   clearInterval(this.timer); //to clear the interval created in componentDidMount
}
```

Q)What if we create setInterval in functional component ?
->

```
useEffect(()=>{
      //API call
      //setINterval function to print Namaste React every second
      setInterval(()=>{
       console.log("Namaste  React in Functional Component");

   },1000);

},[]);
```

If we just write above & not write cleanup code for this.
Our code will be executing even if we leave the page.
useEffect has return some thing which can be used for cleaup activity & called when we leave that component/page.

```
useEffect(()=>{
    console.log("Use Effect logs");

    //API call
    //setINterval function to print Namaste React every second
    const timer=  setInterval(()=>{
      console.log("Namaste  React in Functional Component");

    },1000);
      return ()=>{
        //return will be called when we leave the component/page
        console.log("Cleanup function called in Functional Component");

        //cleanup function to clear the interval
        clearInterval(timer);

      }
},[]);
```

Questions::
1)why we write super(props) & constructor(props) ?
2)why we use async with componentDidMount ?
3) why we not use async with useEffect ?
++
EPISODE 9: Optimizing our App
++

In this, episode we see how we can write code in better way & how we can optimize our app.
& make our app lightweight.

Episode 09 Part 01
==========
->Run our app, the code which we written untill now, can we improve that code ?
Yes.
So, we see how we can write code in much better way.
We will also learn about Custom Hooks as well.

Single Responsibility Principle

If we have function or a class or any single identity code, that should have single responsibility right.
Each component is different functions.
According to the principle, Each component should have single responsibility.
Ex:-
a) RestaurantMenuCard.js component has single responsibility show menu of restaurant.
b)ResaturantCardComponent.js component has single responsibility to show restaurant card.

```
const RestaturantCardComponent =(props)=>{
    const {resData} =props;  //restData is an key which we r

    //Destructuring of our object
    // const cloudinaryImageId = resData.info.cloudinaryImag
      const {
       cloudinaryImageId,
       name,
       cuisines,
       avgRating,
       costForTwo
         } = resData?.info;
      // ? is for optional chaining
```

From parent Component, we pass props to the ResaturantCardComponent, & here we only show the info of restro card.
c)HeaderComponent.js has single responsibility to show only Header on to web page.
So, according to principle, we should make single responsibility for each component & if we see multiple things in component then break it down to each different
component.

Modularity means break down our code into different different small small modules so that our code becomes more maintainable & testable.
How is it become more testable ?
->if there is a big case that we deal with right a single unit of RestaurantCard. So I can write testcases for it. If any bug is there in code, we can catch it easily.

->if we follow single responsibilty principle, we can resuse the code again & again.
So, our code become maintainable, testable, reusable.

Hooks

->It is basically normal javascript function but these are special javascript functions which are given by React.
->we have used some hooks which are given by React like useState, useEffect.
-> useParams which is given by React-router library.
Similarily we can create our own custom hooks.
& how we can use custom Hooks to make our code more modular.
To abstract the responsibility, extra responsibility from these components to a different hook.
Hooks are like utility functions.
So, we will just take out some responsibility from a Component & extract inside a hook so that our hook
& our component becomes more modular.

Episode 09 Part 02
==========

Custom Hooks

->We will create a custom hooks & then we will use inside the ResaturantMenu Component.
->Creating a custom hook is not an mandatory things but it's a very good things bcz that will make our code More readable, modular & reusable.

ResaturantMenu Component Code without Custom Hooks::

src > components > JS RestaurantMenuCard.js > ...

```
import { useState,useEffect, use}from "react";
import ShimmerComponet from "./ShimmerComponent";
import{useParams} from "react-router";
import { MENU_API_URL } from "../utils/constants";


const RestaurantMenuCard =()=>{


/*
we don't have dependency array here, so useEffect will be called every time after the component is
If we want to call useEffect only once after the first render, we have to provide empty dependency
i do not want to call useEffect every time after the component is rendered.
So, we have to provide empty dependency array.
*/
 const [resInfo,setResInfo]=useState(null);
const {resId,resName}= useParams();  //destructuring to get resId,resName directly.
console.log(resId,resName); //resId=nagpur resName=veeraswami-sadar



            useEffect(
             ()=>{
                 console.log("Calling fetch menu");
              fetchMenu();
             },[]);

             const fetchMenu = async ()=>{
                 console.log("fetchin api");

                 const data = await fetch(
                 MENU_API_URL+resId+"/"+resName+"/order&location=&isMobile=0"
                         );
             const jsonData= await data.json();
             console.log(jsonData);

              setResInfo(jsonData.page_data);
```

```
              setResInfo(jsonData.page_data);
             };
             if(resInfo===null)return <ShimmerComponet/>;

             // console.log(resInfo.sections.SECTION_BASIC_INFO);

            const itemCards=resInfo?.order?.menuList?.menus[0].menu.categories[0].category.items;

              console.log( resInfo?.order?.menuList?.menus[0].menu.categories[0].category.items);

              console.log(itemCards);




    return (
        <div className="menu">
          <h1>{resInfo.sections.SECTION_BASIC_INFO.name}</h1>
         <h3>{resInfo.sections.SECTION_BASIC_INFO.cuisine_string}</h3>
        <h2>Menu</h2>
        <ul>

            { itemCards.map((res) =>
                (<li key={res.item.id}>
                    {res.item.name} - {"Rs."} {res.item.price || res.item.default_price}
                    </li>))}

        </ul>
        </div>
    );
};

export default RestaurantMenuCard;
```

->as you see in above code there are two responsibilities in the ResaturantMenu Component.
1)First responsibility is fetching the data.
2)Second responsibility is displaying the data on UI.

ResaturantMenu Component must be concerned about displaying the data not worry about fetching the data.
From where we fetching data ? How the data is coming from ? What is the API to fetch data?
This things should not be worry about this for ResaturantMenu Component.

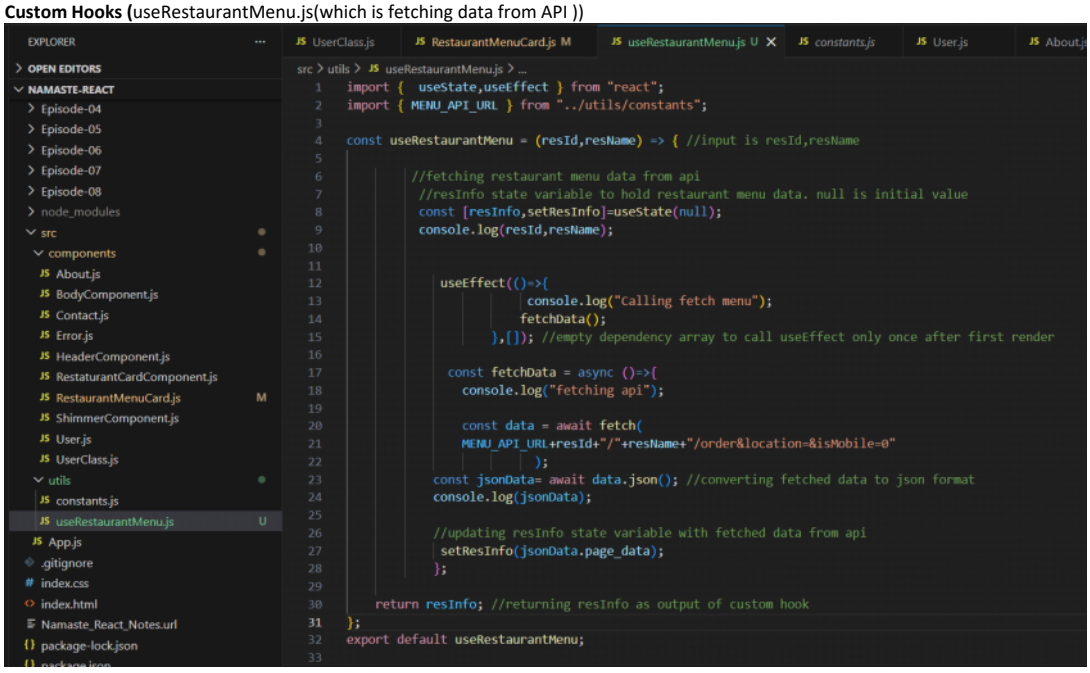
Take example of
`const {resId,resName}= useParams();  //destructuring to get resId,resName directly.`
We had useParam hooks gives us resId. & we don't know what is code written behind the scenes.
useParam hooks is an utility function.
We don't know how useParams getting parameter ? We don't know how the code is written ?
We not bothered about the implementation of hook.
Similarily, can I abstract my fetching data logic also ?

->Hook is an utility function so create hook inside the utility folder.
For separate hook, create separate file.
fileName of hook is always start with use(smallcase)RestaurantMenu.

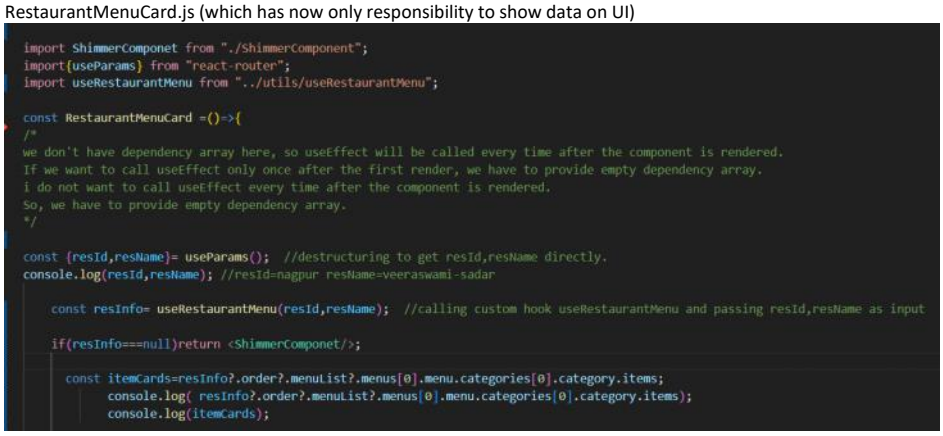
Ex:- useRestaurantMenu.js
So, if we start with use to create custom hooks, React understands it better way.

ResaturantMenu Component Code with Custom Hooks::

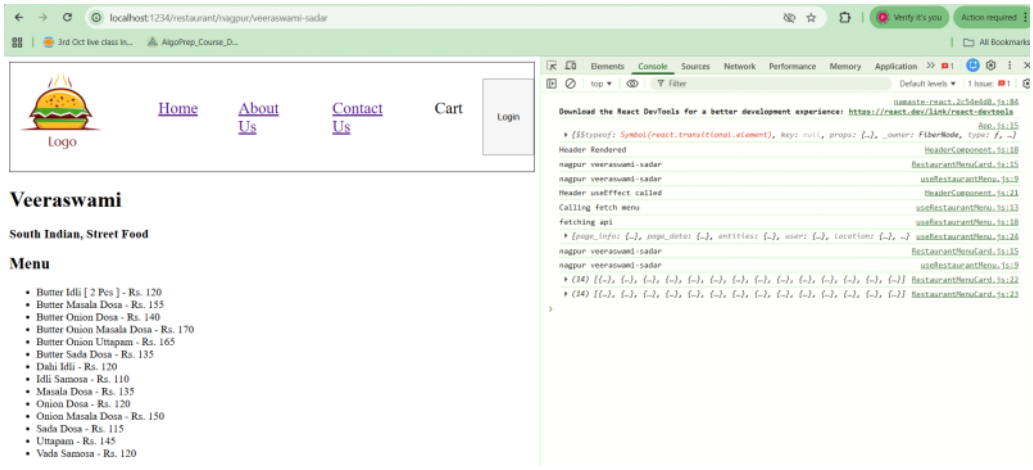
Custom Hooks (useRestaurantMenu.js(which is fetching data from API))



RestaurantMenuCard.js (which has now only responsibility to show data on UI)



O/P=> Code is working as expected.



NOTE:: As we are using zomato api for showing menu & swiggy api to show resaturant info. We need to manually add resId & resName in URL
URL=> http://localhost:1234/restaurant/nagpur/veeraswami-sadar
resId is nagpur. resName is veeraswami-sadar.

So, tomorrow, I need to test my fetching data logic I need to test only useRestaurantMenu.js hook.

Episode 09 Part 03
==========

->Now we are building one feature like user's internet is active or not.
As in FaceBook in chat, if user is active we see green dot. If not, we not see green dot.

So, instead of showing Restaurant Card when user is offline, we show you are offline as no internet or internet not working.

How to write the custom hook ?
->first identify the contract means what is input of that hook.
What is output of that hook.

Do we need any specific info to show online status ?
->No.

We use browser event Listener to keep track of user is online or offline.
https://developer.mozilla.org/en-US/docs/Web/API/Window/online_event

Code::-

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

const useOnlineStatus=()=>{
      //feature try to check if user's internet connection is online or offline

  const [onlineStatus, setOnlineStatus]= useState(true);//initially true

    //we need to add event listeners for online and offline events.
    //for only one time we need to add event listeners, so we will use useEffect with empty dependency array.
   useEffect(()=>{
      //so, i will put the event listeners on to the web page only once when the component is mounted.
      //then it will keep listening for online and offline events.
      window.addEventListener("offline",()=>{
        console.log("You are offline");
        setOnlineStatus(false); //set onlineStatus to false
      });
      window.addEventListener("online",()=>{
        console.log("You are online");
        setOnlineStatus(true); //set onlineStatus to true
      });
   },[]); //empty dependency array to call useEffect only once after first render

   //return true/false
   return onlineStatus;

};

export default useOnlineStatus;
```

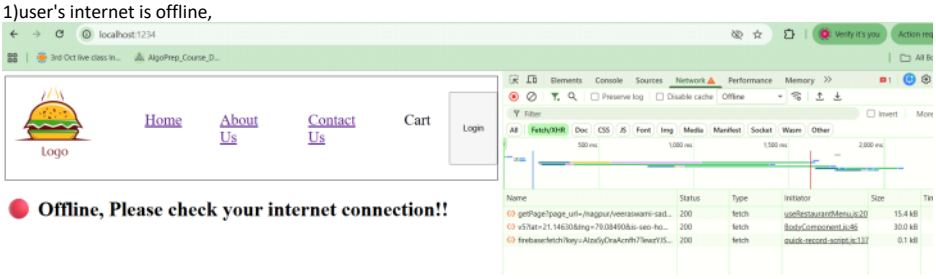
userOnlineStatus.js (Custom Hook) (it checks user's internet is offline or online)

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

const useOnlineStatus=()=>{

    //try to check if user's internet connection is online or offline

    const [onlineStatus, setOnlineStatus]= useState(true);//initially true

    //we need to add event listeners for online and offline events.
    //for only one time we need to add event listeners, so we will use useEffect with empty dependency array.
    useEffect(()=>{
        //so, i will put the event listeners on to the web page only once when the component is mounted.
        //then it will keep listening for online and offline events.
        window.addEventListener("offline",()=>{
           console.log("You are offline");
           setOnlineStatus(false); //set onlineStatus to false
        });
        window.addEventListener("online",()=>{
           console.log("You are online");
           setOnlineStatus(true); //set onlineStatus to true
        });
    },[]); //empty dependency array to call useEffect only once after first render

    //return true/false
    return onlineStatus;

};

export default useOnlineStatus;
```

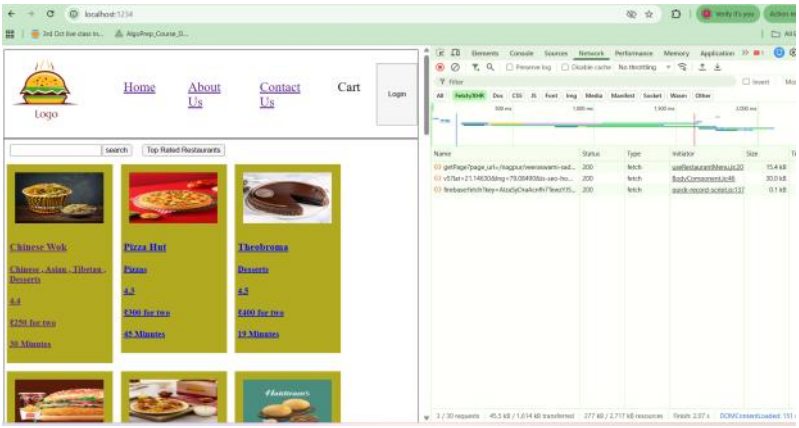
Use this inside the BodyComponent.js

```
    const onlineStatus= useOnlineStatus();
    if(onlineStatus===false){
      return( <h1>
        🔴 Offline, Please check your internet connection!!
        </h1>);
    }

  return     listOfRestaurants.length===0 ? <ShimmerComponet/> : (
    <div className="body">
     <div className="filter">
      <div  className="search">
          <input  type="text"  className="search-box"  value={searchText}
           onChange={ (e)=>{
                setSearchText(e.target.value);
           }} />
          <button
          onClick={()=>{ //Filter the restaurant cards & update the UI.  //we need searc
const filterRestaurantList=listOfRestaurants.filter(res=>
 res.info.name.toLowerCase().includes(searchText.toLowerCase())
);
//update the restaurant list by filtered restaurant list.
 setFilteredRestaurants(filterRestaurantList); }}
```

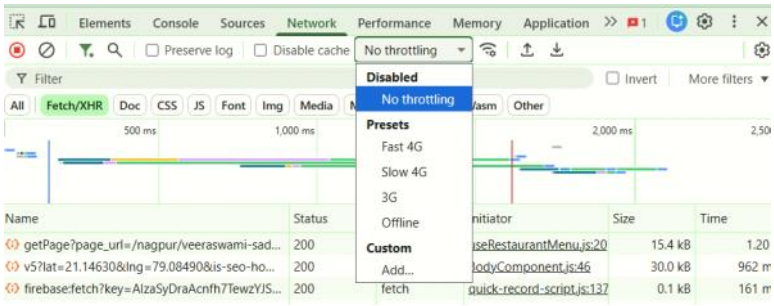
o/p on UI=>
1)user's internet is offline,



2)user's internet is Online.



Note::- Without stopping main internet connection of laptop, we can stop internet for that particular web page,
By disabling internet in Network tab.



->
Now I want to show online status in header whether user's internet is online or offline.
We reuse that userOnlineStatus custom hook.

HeaderComponent.js

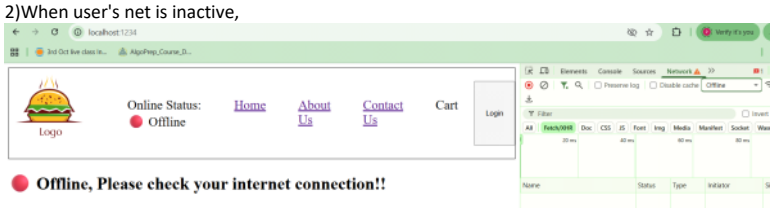
```
import useOnlineStatus from "../utils/useOnlineStatus";
```

```
const onlineStatus = useOnlineStatus();
```

```
<li>
  Online Status: { onlineStatus? "🟢 Online":"🔴 Offline"}
</li>
```

o/p=>
1)When user's net is active,

2)When user's net is inactive,



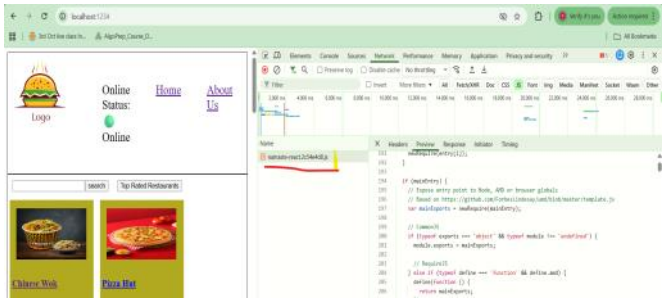
Episode 09 Part 04
==========

Q)when we create our custom hooks so, is it mandatory to use the word use before we write name ?
->it is not mandatory. If we see React Documentation, the recommended us to use the word use.
Why ?
->we create component & its first letter is always capital.
Similarly for hooks we use word as use.
->a lot of times most people set up linting process & our linter will throwing error if we don't put it use before custom hooks, etc.
If not follow convention linter will throw error.
So, always do that what library recommends.
So, use words before fileName to create custom hooks.
Also it is easy for easy to read for other developer also which is hook, component.
Ex:- useOnlineStatus is a custom hook.
If we rename this to getOnlineStatus.
getOnlineStatus is an custom hook but due to naming differently it looks like a function call.

Episode 09 Part 05
==========

How to optimize our app

In Large scale application, there are so many components are there & there performance are not good.
So, when we are building large scale applications & wants a performance to be good.
For this we study now,
->parcel is a bundler which it all our file & make it into one files.
So, it will combine all js file & generate one .js file in dist folder.

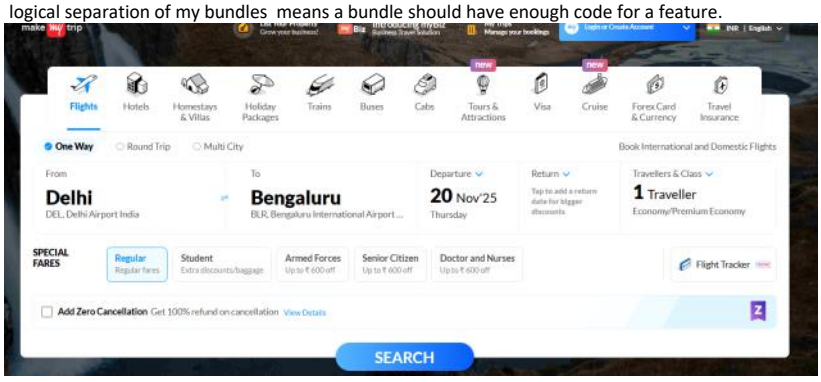


Q) is it good or bad to have one js file ?
->
Lets take makeMyTrip website,
This website has so many components.
So, if we make one js file from a thousands of component.
What is the problem here ?
The size of js file will increase a lot.

->We can't build large scale production ready application if we not take care of this.
Our app will become very slow & it will take a lot of time to load.
So, whenever we building large scale application, break our app down into smaller pieces.
This is possible solution.
Can I break down my app into smaller pieces ?
Can I do something so that my application will not have one javascript file but smaller javascript file. ?

->1)if we have thousands of component, we don't need to load thousand of file onto web page.
It will make more load on browser to do by making thousand calls.
Also 2)we do not want to put like thousand of file into one js file.
Both of these solution is not true.
So we try to make smaller bundles of this files.
This process is known as Chunking or codeSplitting or dynamic Bundling or lazy laoding, on demand loading, dynamic import,etc.
i.e. we have to make our application into smaller chunks. i.e. we have to bundling in dynamic way.

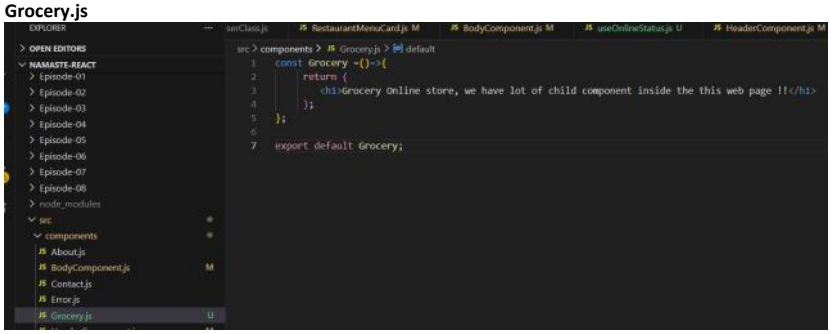
Q)How to make smaller bundles ? When we make smaller bundles ? What should be there in smaller bundles ?
->let's say I want to do system design of make my trip.
For me I want to do logical separation of my bundles.
logical separation of my bundles means a bundle should have enough code for a feature.



->Flight is one chunk of bundle. It contains 100 component & make it to one bundle.
->Hotel is another bundle.

How can we build this logical bundle in our food ordering app?
->
Let's assume situation where our app is not just delivering food it also start delivering groceries.
So our app has food delivery & delivery groceries.
So food delivery has 10 components.
& grocery delivery has 12 components.
So, we will make one bundle for groceries & diff bundle for food delivery.

So, make grocery component in our food delivery app.
Grocery.js



Add one link in **HeaderComponent.js** to move to grocery web page.

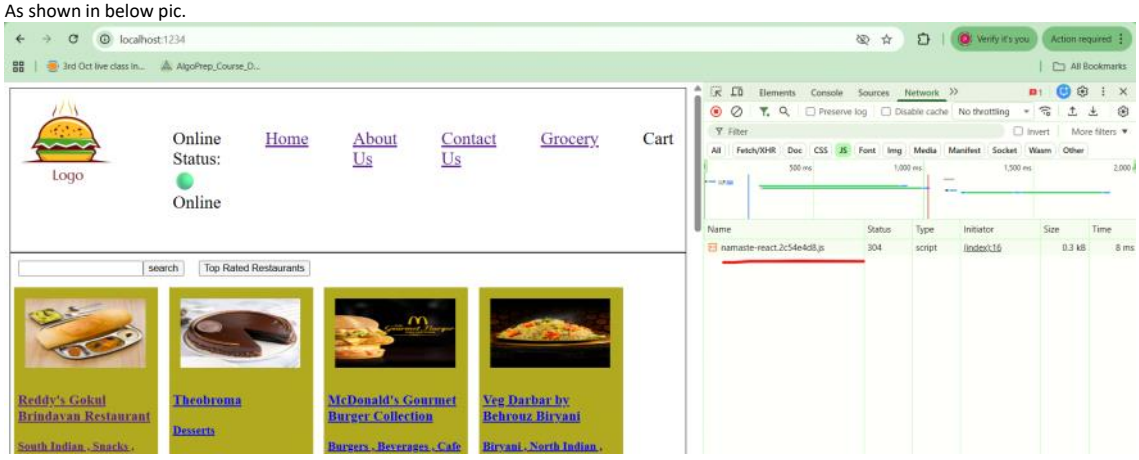
```jsx
const HeaderComponent =()=>{
    <div className="header">
      <div className="logo-container">
        <img className="logo" src={LOGO_URL} />
      </div>
      <div className="nav-items">
          <ul>
          <li>
            Online Status: { onlineStatus? "🟢 Online":"🔴 Offline"}
          </li>
          <li> <Link to="/">Home</Link></li>
          <li> <Link to="/about">About Us</Link></li>
          <li>
            <Link to="/contact">Contact Us</Link>
          </li>
          <li>
            <Link to="/grocery">Grocery</Link>
          </li>
```

Specify router url in **App.js** also to move to that Grocery component when user click on Giocery link.

```jsx
import Grocery from "./components/Grocery";
```

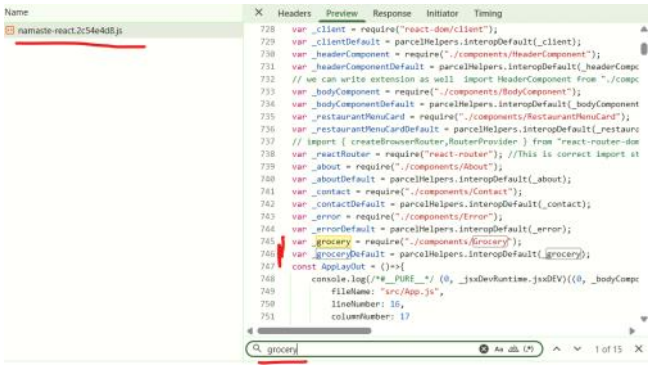
```jsx
const appRouter=createBrowserRouter([
  {
    path:"/",
    element:<AppLayOut/>,
    children:[
      {
      path:"/",
       element:<BodyComponent/>
    },

    {
      path:"/about",
       element:<About/>
    },{
      path:"/contact",
      element:<Contact/>
    },{
      path:"/grocery",
      element:<Grocery/>
    },{
```

o/p=>
As soon as we load web page only one js file is created for Grocery component as well.
As shown in below pic.



When user click on Grocery link, it will routed to /grocery web page. & js file is still the same.

See in below js code our grocery component in same js file.



->Now I want to logically distribute my application that all grocery related components should come from a different bundle.
So, I will not import my Grocery.js component in App.js file directly like

```
import Grocery from "./components/Grocery";
```

But I will import my Grocery.js component in App.js using **lazy loading**.
We can also call these as **Chunking/code splitting/dynamic bundling/on-Demand-Loading /Dynamic Import**.
Means when our app will load initially our app Home page will load but it will not load the code for Grocery.
Only when I go to my Grocery page by clicking on the link of Grocery then there will be Grocery code will be load in our app.
So, using lazy() function which is given by React to load code using lazy loading style.
It is Named export, we will import it using {} parenthesis.
Import {lazy} from "react";

//lazy is function which takes function as an argument which returns a dynamic import.
//import() is a function which is used to dynamically import a module.
//inside the import() we have to provide the path of the module we want to import.
const Grocery = lazy(()=>import("./components/Grocery"));

App.js

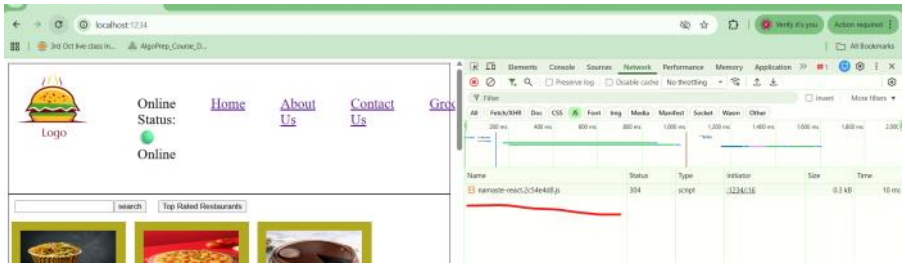
```
import React, { lazy } from "react";
```

```
//lazy is function which takes function as an argument which returns a dynamic import.
//import() is a function which is used to dynamically import a module.
//inside the import() we have to provide the path of the module we want to import.
const Grocery = lazy(()=>import("./components/Grocery"));
```

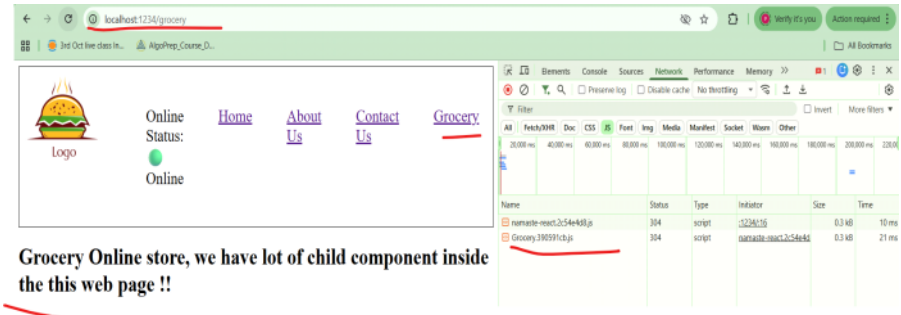
```
src > JS App.js > ...
49   const appRouter=createBrowserRouter([
53       children:[
56           element:<BodyComponent/>
57       },
58
59       {
60         path:"/about",
61          element:<About/>
62       },{
63         path:"/contact",
64         element:<Contact/>
65       },{
66         path:"/grocery",
67         element:<Grocery/>
68       },{
```

Now, our app contain separate file for food delivery Bundle & Grocery bundle.

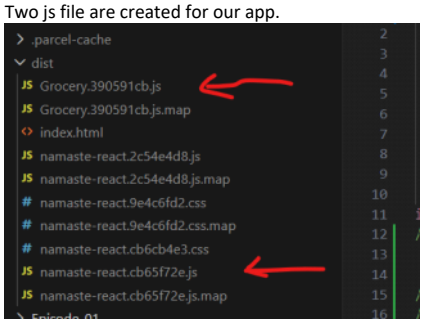
As soon as I load home page, our food delivery bundle js file will load which does not have grocery code.



& when we click on Grocery link, our Grocery bundle JS code will be load in separate bundle.



See in our VS-code .dist folder,
Two js file are created for our app.

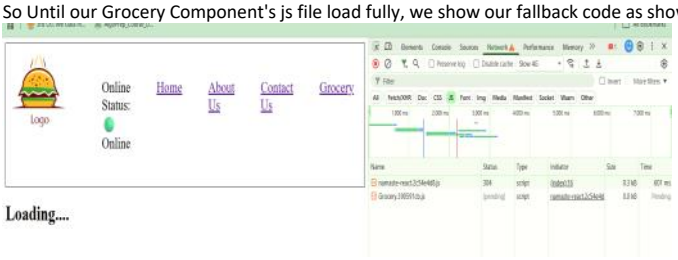


Sometimes, when we click on Grocery link, our Grocery bundle JS code will not be load instantly, React throw error,
To handle this we have Suspense which is an Component given by React.
We need to use this & wrap our Grocery Component into the Suspense component.

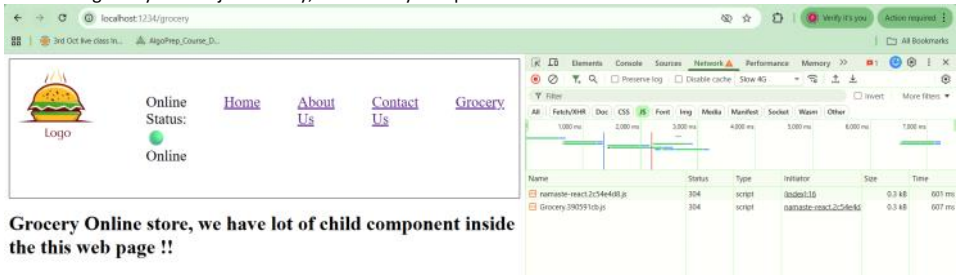
```
import React, { lazy, Suspense } from "react";
```

```
},{
  path:"/grocery",
  element: (<Suspense fallback={<h1>Loading....</h1>}>
  <Grocery/>
  </Suspense> )
},{
```

Fallback is function we need to provide code in JSX. Until our Grocery component is not loaded what we need to show We write inside the fallback function.
So Until our Grocery Component's js file load fully, we show our fallback code as shown in below.



Then once grocery bundle.js load fully, our Grocery Component will load.



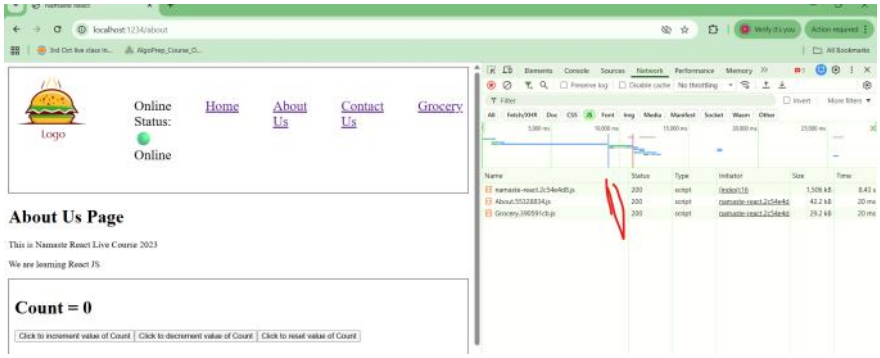
->So, if our App has large Number of Component & its our app size is very big then we need to do chunking of our code.
->we do chunking of About.js component also by using lazy loading.

Code:-

```
const About = lazy(()=>import( "./components/About"));
```

```
{
  path:"/about",
   element: (<Suspense fallback={<h1>Loading....</h1>}>
<About/>
  </Suspense> )
},{
```

O/P=> so now our three bundles are created. 1)Food app bundle.js 2)about.js bundle 3)Grocery bundle.js



This is how we distribute our application into smaller smaller chunks.

++
EPISODE 10: Jo Dikhta Hai, Vo Bikta hai.
++

In this, episode we see how we can make our app beautiful.
Means how we can add styles or CSS to our applications.

Episode 10 Part 01
==========

First Way of Writing CSS(**Normal way using index.css**)

We have index.css file & we put all these CSS into this file.
This is very easier & basic way of putting CSS.

Second Way of Writing CSS (**SASS,SCSS**)

SASS(Systematically Awesome Style Sheets) mean writing CSS with some super power.
So it makes CSS to write in easier way.

SCSS:
but 2nd way is not recommended way to write CSS because as our application scales this writing styles does not scale well.
This is not used industry production ready applications.

3rd Way of Writing CSS(**Styled Components**)

Styled Components is another way to add CSS.
https://styled-components.com/
For React, most companies uses Styled Components to write CSS.

We only learn ***Tailwind CSS*** framework in this Series.

Material UI, Bootstrap, Chakra UI, ant Design are CSS library which helps to write CSS.
A lot of companies library or framework to write CSS.
We get pre-built components in library or framework.

In This episode, we see how to style of Components using Tailwind CSS framework.

Episode 10 Part 02
==========
TailWind CSS

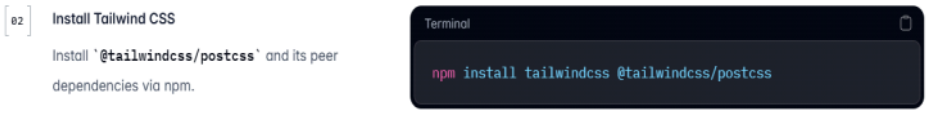
->To use tailWind CSS, we need to configure it in our project.
->TailWind CSS can work with other framework as well.

Go to website:- https://tailwindcss.com/docs/installation/framework-guides
Go to Framework, choose parcel as we are using parcel.

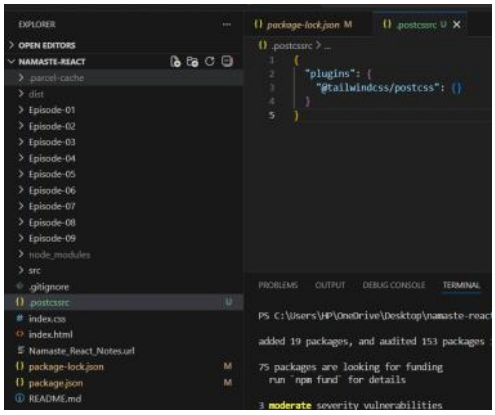
As I am using tailwind css 4 version my steps to install it diff but follow the steps properly.

1) As we have project so don't need to crate project again.

2) Install tailwindcss & postcss.



postCSS is an tool for transforming CSS with javascript.
Basically If I want to transfer our CSS inside javascript, we use postcss tool.
So, tailwindCSS uses postcss behind the scenes.
3)

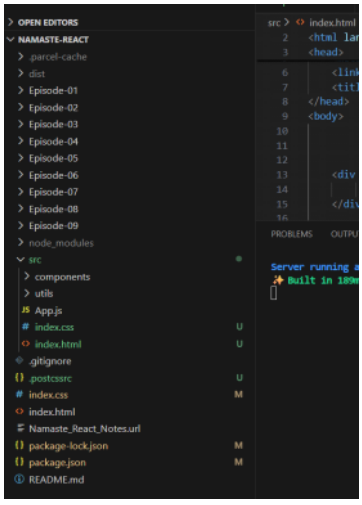


Our parcel uses postcssrc file is used to understand tailwind.
So, in this file we just telling we are using tailwind in our project, please parcel understand tailwind.
.postcssrc is an configuration file.
4)

5)

6)Now we are ready to use tailwind in our project.

Now our all index.html, index.css & .js file inside the src folder only.
The file which we see html & css outside src folder that we use in our previous episodes.



Episode 10 Part 03
==========

->Our Configuration for tailwind CSS is done.

How Tailwind CSS works ?

->it works diff than normal CSS.
->In Normal CSS, we just write className or idName in JSX or javascript file then we add css for this classes or id in index.css file
->Using tailwind, we put CSS using className automatically for every CSS that you want to write in your app.
Ex:- I want to make background color of app to red.
So there is a separate different class which tailwind css gives you that we can use it directly.
Suppose we want to increase width of our image, there a separate class we need to use for it.

Thought process to write tailwind css is same as we write normal CSS.

->Install tailwind vs code extension.

->If this extension is not there, I never use tailwind.
Once this extension is installed, it will give us suggestions when we are writing.

Mt ->margin on the top.
Mb->margin on the bottom.
px -> padding on x axis on both left & right side.
py -> padding on y axis on both up & bottom side.

->tailwind suggest all value in rem but I want to give 200px value, we can give dynamically inside the square bracket.
w-[200px]

->

```
<div className="flex justify-between bg-pink-100 shadow-lg m-2 sm:bg-amber-300 lg:bg-green-200">
```

Here we changing color based on size of devices.
If device is small, header color will be yellow.
If very large, color will be green.

Pros of tailwind:-
1)we don't need to shift from current file to index.css to write css details again & again.
2)it is very lightweight bcz we apply css only. Parcel will bundles that used css only.
Unused css of tailwind will not bundles.
Ex:- if we m-4 10 times in diff diff parts of code.
So, m-4 just import one time in code not every time.

Cons:-
1)when I want to apply a lot of css for each className that makes looks ugly.
& make our code not good readable.

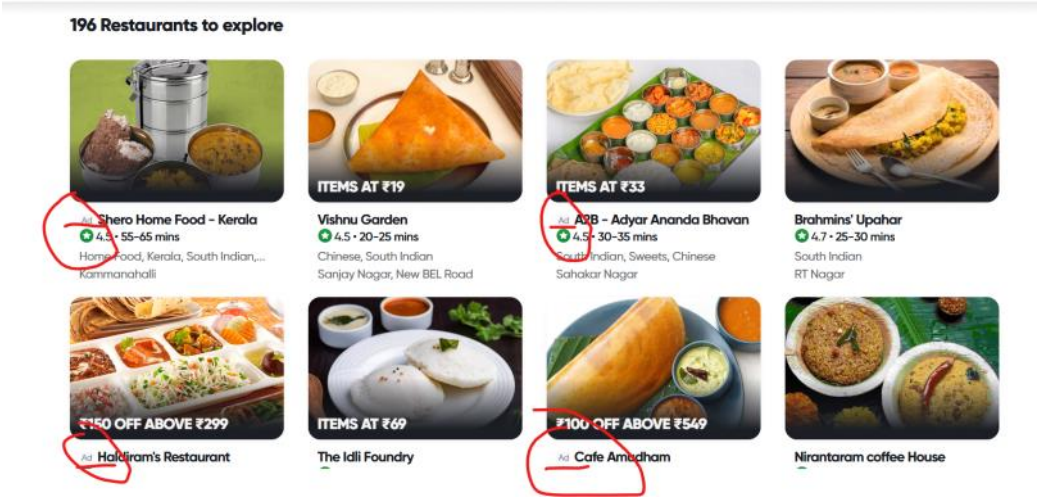
++
EPISODE 11: Data is New oil.
++

In this, episode we learn how to manage data inside the our react application.

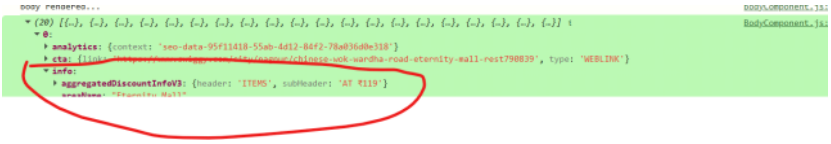
Episode 11 Part 01
==========
Higher Order Component in React

->Higher Order Component is a function that takes a component & returns a component.
->Basically Higher Order Component takes a component as input & the enhances that component.
It add some extra features to that input component & return back.
->Higher Order Component acts like a enhancer kind of function.

->We will develop feature using Higher Order Component in our Application.



->As we see in above swiggy website, we see a list of restaurant, but some of them restaurant are promoted by swiggy by giving ad/promoted on their card.
So, we will develop that feature how we can apply promoted/ad feature on some restaurant card using higher order component.
So, basically all restaurant card are same but some cards having some additional feature like ad/promoted so we will built using Higher order component.
similary above feature, we will implement functionality such that from a list of resrtaurant if some of them has discount available we show the discount label similar to ad/promoted.



So, in our json data, if this aggregateDiscount is empty, means we have no discount for that restaurant.

//Create one Higher order function for RestaurantCardComponent.
We create higher orderfunction inside that component itself.

RestaurantCardComponent.js

```
  //Higher order component

//input - RestaurantMenuCard
//output - RestaurantMenuCardDiscountOffer

export const withDiscountRestaurantMenuCard =(RestaurantMenuCard)=>{

        //props are passed from BodyComponent to RestaurantMenuCardWithDiscount
        //then from RestaurantMenuCardWithDiscount to RestaurantMenuCard
        //...props is used to pass all the props at once.
          return (props)=>{
                  return (
                   <div className="border-[1px] ">
                     <label  className=" absolute ■text-white □bg-black" >Discount</label>
                     <RestaurantMenuCard {...props}/>
                   </div>
                  );
          };
};
```

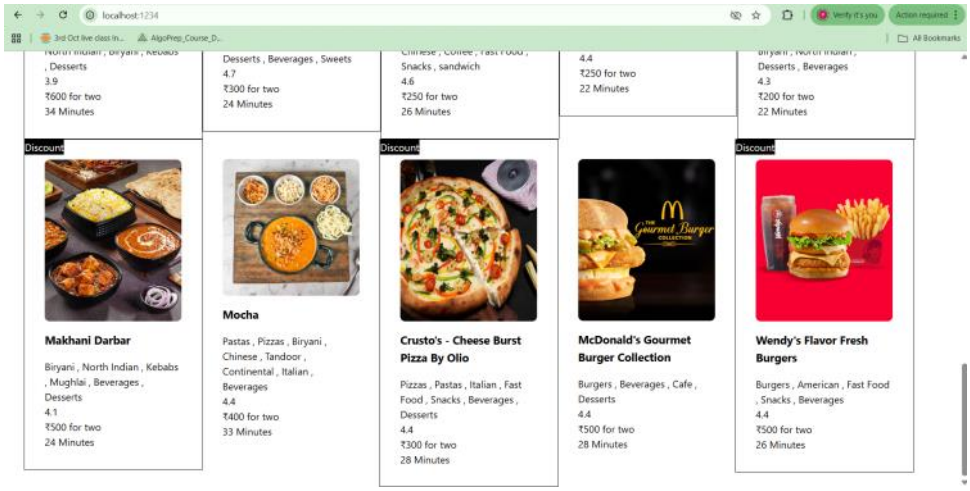
BodyComponents.js

```
import RestaturantCardComponent, {withDiscountRestaurantMenuCard} from "./RestaturantCardComponent" ;
```

```
//created a new component RestaurantMenuCardWithDiscount using Higher Order Component
const RestaurantMenuCardWithDiscount= withDiscountRestaurantMenuCard(RestaturantCardComponent);
```

```
        <div className="mx-4 px-4 flex flex-wrap">
       {filteredRestaurants.map((restaurant)=>
       (
       <Link
          key={restaurant.info.id}
        to={"/restaurant/"+restaurant.info.id+"/"+restaurant.info.name}   >

        {/* {if Restaurant have discount then add a discount  offer on the scard.} */}
           restaurant.info.aggregatedDiscountInfoV3 !==undefined ?
            ( <RestaurantMenuCardWithDiscount  resData={restaurant} />)
             :
             (
              <RestaturantCardComponent  resData={restaurant} />
            )
         }

       </Link>

       ))}
       </div>
```

O/P ON UI=>

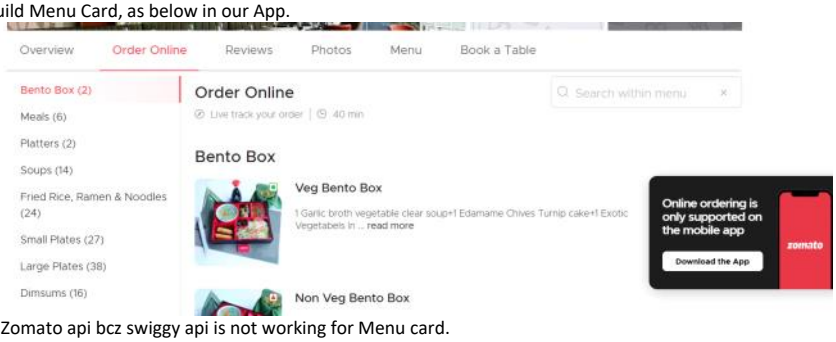


So for some of card, it will display Discount as label if discount available on that restaurant.

Episode 11 Part 02
==========
Controller & UnControlled Component in React , Lifting the state up.

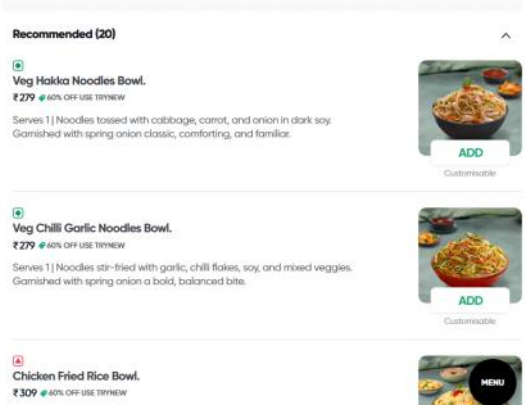
->Important of part of React application is to manage the data.
->All React applications have two important layers.
1)UI Layer 2)Data Layer.
UI layer is powered by Data Layer.
Data Layer consist of our state, props, local variables.
So, Data layer is very important.
If we manager our data, we make our applications super fast.
->UI Layer mostly consist of JSX.
->Data Layer is our state, props, local variables, code we write under {}.

Now,
We build Menu Card, as below in our App.



->it is Zomato api bcz swiggy api is not working for Menu card.
https://www.zomato.com/nagpur/tokii-gayatri-nagar/order
https://jsoneditoronline.org/#left=local.ponate&right=local.qepasu

So, Now we are building this type of Menu Card functionality as avaibale in Swiggy. It means that we need to show our Menu categories wise.



1)
We will create one RestaurantCategoryComponent.js

```jsx
import ItemListComponent from "./ItemListComponent";


const RestaurantCategoryComponent =({data})=>{

    console.log(data);
console.log(data.categories[0].category.items);



    return(
        <div>
            <div className="w-6/12 mx-auto my-4 bg-gray-50 shadow-lg p-4" >
            <div className="flex justify-between">
              <span  className="font-bold text-lg" >{data.name} ({data.categories[0].category.items.length})</span>
               <span>+</span>
            </div>
                <ItemListComponent items={data.categories[0].category.items}/>
           </div>
        </div>
    );

};

export default RestaurantCategoryComponent;
```

2)To Show List of Dishes for Each Category we create one another component i.e. ItemListComponent.js

```jsx
src > components > JS ItemListComponent.js > [@] ItemListComponent
const ItemListComponent = ({items})=>{
    console.log(items);
    return(
          <div>
          {items.map((e)=>{

              <div key={e.item.id}  className="p-2 m-2  border-gray-200 border-b-2 text-left flex justify-between" >
                  <div  className="w-9/12" >
                <div className="py-2">
                  {/* show item name */}
                     <span >{e.item.name}</span>
                     {/* to show item price, if price is not there show default price */}
                     <span> - ₹ {e.item.price!==0 ? e.item.price : e.item.default_price }</span>

                     </div>
                     {/* to show item description */}
                  <p className="text-xs" >{e.item.desc}</p>
                  </div>

                  {/* if image not there for particular item we will not show image for that,
                   so we write condition here  so that if image not available we will display add button for that.*/}
                  {e.item.item_image_url==undefined ?
                    <div className="w-3/12 p-4">
                  <div className="absolute" >  <button className="p-2 mx-16 bg-black  text-white shadow-lg rounded-lg" > Add +</button>
                  <div>  <img src={e.item.item_image_url} className="w-full"/></div>
                  </div>
                  : <div className="w-3/12 p-4"> </div>
                 } </div>

    ))}
      </div>
    );
};

export default ItemListComponent;
```

3)As we are showing this all list of item category wise in Menu section then add our RestaurantCategoryComponent into RestaurantMenuComponent.

```jsx
import ShimmerComponet from "./ShimmerComponent";
import{useParams} from "react-router";
import useRestaurantMenu from "../utils/useRestaurantMenu";
import RestaurantCategoryComponent from "./RestaurantCategoryComponent";

const RestaurantMenuCard =()=>{
/*
we don't have dependency array here, so useEffect will be called every time after the component is rendered.
If we want to call useeffect only once after the first render, we have to provide empty dependency array.
i do not want to call useEffect every time after the component is rendered.
So, we have to provide empty dependency array.
*/
const {resId,resName}= useParams();  //destructuring to get resId,resName directly.
console.log(resId,resName); //resId=nagpur resName=veeraswami-sadar

   const resInfo= useRestaurantMenu(resId,resName);  //calling custom hook useRestaurantMenu and passing resId,resName as input

   if(resInfo===null)return <ShimmerComponet/>;

     const itemCards=resInfo?.order?.menuList?.menus[0].menu.categories[0].category.items;
          console.log( resInfo?.order?.menuList?.menus[0].menu.categories[0].category.items);
          console.log(itemCards);
```

```jsx
   const resInfo= useRestaurantMenu(resId,resName);  //calling custom hook useRestaurantMenu and passing resId,resName as input

   if(resInfo===null)return <ShimmerComponet/>;

     const itemCards=resInfo?.order?.menuList?.menus[0].menu.categories[0].category.items;
          console.log( resInfo?.order?.menuList?.menus[0].menu.categories[0].category.items);
          console.log(itemCards);

            //getting categories list from resInfo object.
            //it contains array of categories with items.
          const categoriesItemsList= resInfo.order.menuList.menus;
          console.log(categoriesItemsList);


   return (

      <div className="text-center">
        <h1 className="font-bold my-6 text-2xl" >{resInfo.sections.SECTION_BASIC_INFO.name}</h1>
        <h3 className="font-bold text-lg" >{resInfo.sections.SECTION_BASIC_INFO.cuisine_string}</h3>

                  {
                    // iterating over categoriesItemsList array to render RestaurantCategoryComponent for each category.
            categoriesItemsList.map((category)=>{
                    // passing category data as prop to RestaurantCategoryComponent
                    // key is unique id for each category to help react identify which items have changed, are added, or are removed.
                  <RestaurantCategoryComponent  key={category.menu.id} data={category.menu} />
          ))}
      </div>
    );
};
export default RestaurantMenuCard;
```

4)Run app O/P on UI=>

->Now we are building as soon we click on + button, we list of item for that category will close/open i.e. carousal.
As our Header of categories like,

Bento Box (2) +

As make our cursor to pointer & & apply event here. As soon as click on Header of category, it must be toggle.
Toggle means if open then closed & if closed then it will be open.

```
    return(
        <div>
            <div className="w-6/12 mx-auto my-4 bg-gray-50 shadow-lg p-4" >
            <div className="flex justify-between cursor-pointer" onClick={handleClick} >
              <span  className="font-bold text-lg" >{data.name} ({data.categories[0].category.items.length})</span>
               <span>+</span>
            </div>
            {   //If showItems is true then only render ItemListComponent
               //if showItems is false then do not render ItemListComponent
                showItems &&  <ItemListComponent items={data.categories[0].category.items}/>
            }

           </div>
        </div>
    );

};
```

handleClick function::-

```
const handleClick=()=>{
    console.log("Category clicked");
    //Write logic to toggle showItems value
    //if showItems is true, set it to false
    //if showItems is false, set it to true
    //this is toggle functionality
    setShowItems(!showItems);
};
```

-> we will create one state variable as soon as click, it must be true so that it will show items,
If false, it will close all items.

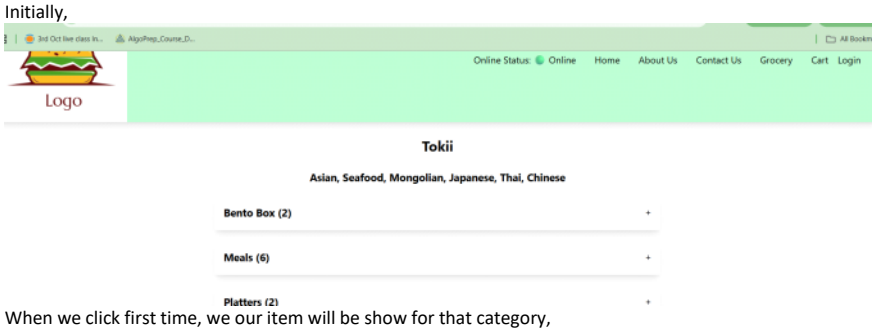
```
import ItemListComponent from "./ItemListComponent";
import {useState} from "react";

const RestaurantCategoryComponent =({data})=>{

    console.log(data);
console.log(data.categories[0].category.items);

//State variable to track whether to show items or not
const [showItems, setShowItems] = useState(false);
```

Now, whole complete code,

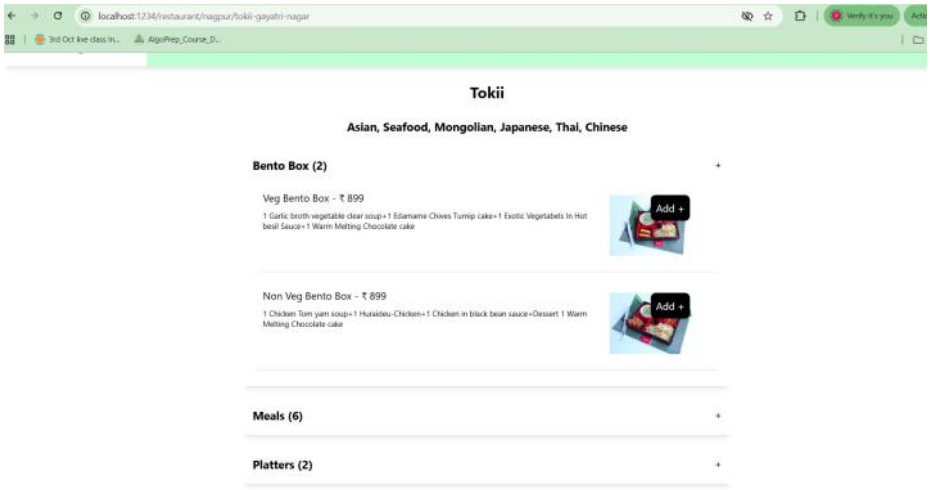
```
import ItemListComponent from "./ItemListComponent";
import {useState} from "react";

const RestaurantCategoryComponent =({data})=>{

    console.log(data);
console.log(data.categories[0].category.items);

//State variable to track whether to show items or not
const [showItems, setShowItems] = useState(false);


const handleClick=()=>{
    console.log("Category clicked");
    //Write logic to toggle showItems value
    //if showItems is true, set it to false
    //if showItems is false, set it to true
    //this is toggle functionality
    setShowItems(!showItems);
};
```

```
    return(
        <div>
            <div className="w-6/12 mx-auto my-4 bg-gray-50 shadow-lg p-4" >
            <div className="flex justify-between cursor-pointer" onClick={handleClick} >
              <span  className="font-bold text-lg" >{data.name} ({data.categories[0].category.items.length})</span>
                <span>+</span>
            </div>
            {   //If showItems is true then only render ItemListComponent
              //If showItems is false then do not render ItemListComponent
               showItems &&  <ItemListComponent items={data.categories[0].category.items}/>
            }

            </div>
        </div>
    );

};

export default RestaurantCategoryComponent;
```

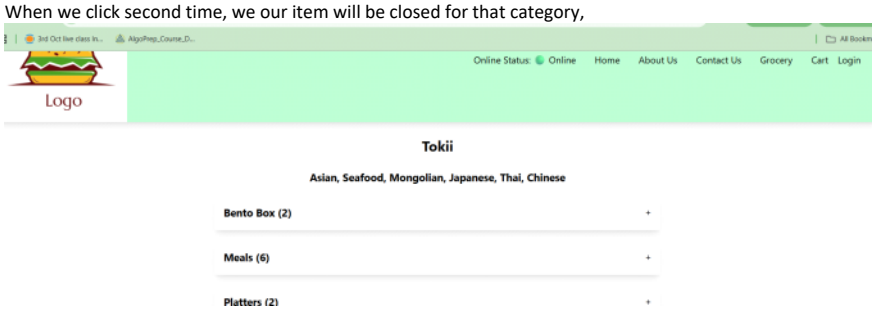
O/P=>
Initially,



When we click first time, we our item will be show for that category,



When we click second time, we our item will be closed for that category,



Episode 11 Part 03
==========

Now, we want to build a one feature that as we have 5 category.
As soon we click on one category/carousel it show all item.
Then I click on seconds category it also shows all items but it will not closed other open carousel.
We build this type of functionality.

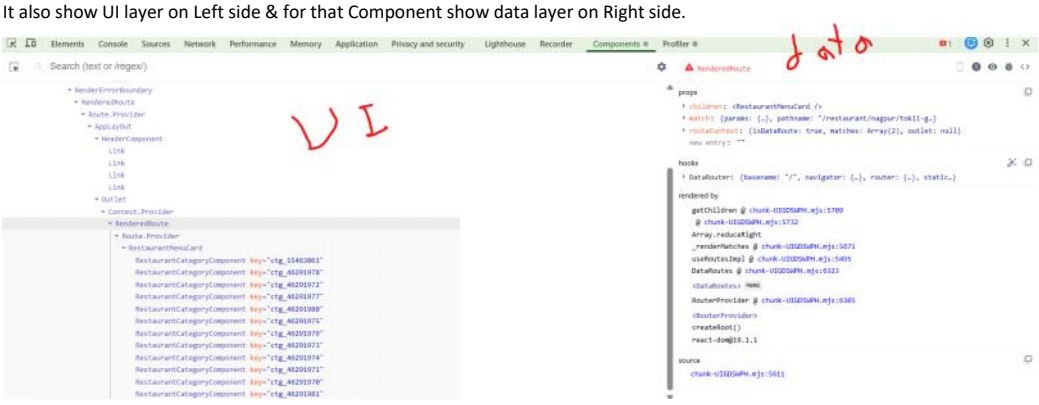
To Build this kind of feature using existing code as see part 11-02 is very tricky.

->Install React Developer tool for debugging.

->Once we installed that tool, our developer console will get superpower.

We get Components & Profiler.

In **Components** section, it will show us all our components for that app.
It also show UI layer on Left side & for that Component show data layer on Right side.

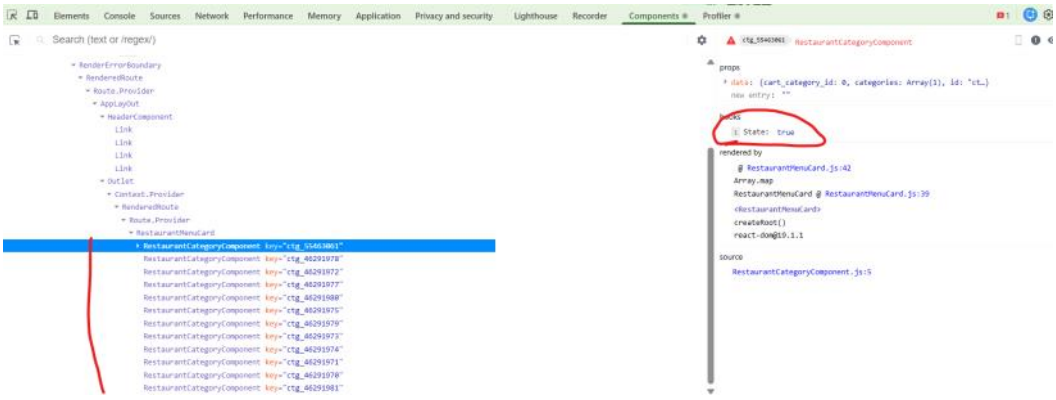


->This above UI is also called Virtual DOM of UI.
->Data layer shows all props which we pass into that component.

Profiler:-
Profiler records our React application.
From here we can record our activity we do in our React app.
Once we closed recording, it will show is flamegraph like how much time it will take to load the page.

->From here we can find our which component is taking time & then we can fix that.

Now, we want to build a one feature that as we have 5 category.
As soon we click on one category/carousel it show all item.
Then I click on seconds category it also shows all items but it will not closed other open carousel.
We build this type of functionality.



->Each RestaurantCategoryComponent has its own state, if we open any carousal, state will be true.
If closed state, will be false.
->Each Category is independent, they does not know each other what are they doing.
But if I want to build the feature if any category is open & I am trying to open other category,
So other category that open previously must be closed means I want this state to be lifted up.
So I do not want to give the power of show item list to Each Category component.
I will give the power of show/expand/close the item list of Each category to the parent of Category Child.
i.e. I want to give this power to RestaurantMenu Component instead of RestaurantCategory Component.
->So, we want RestaurantMenu Component will control the power of show/expand/close the item list of RestaurantCategory Component.
So when Parent Component like RestaurantMenu taking control of showing item list for that RestaurantCategory component or not, then it is Controlled Component.
Because RestaurantCategory is not controlling the showing list of items.

->Earlier Each RestaurantCategory component controlling our own state to show list of Item or not.so Parent Component does not have any control On the Child Component, so we called it as "UnControlled Component".

->RestaurantCategoryComponent.js

```
const RestaurantCategoryComponent =({data, showItems})=>{

    console.log(data);
console.log(data.categories[0].category.items);

//I want to show items by giving control to parent component RestaurantMenuCard

    return(
        <div>
            <div className="w-6/12 mx-auto my-4 bg-gray-50 shadow-lg p-4" >
            <div className="flex justify-between cursor-pointer" >
              <span  className="font-bold text-lg" >{data.name} ({data.categories[0].category.items.length})</span>
               <span>+</span>
            </div>
            {   //if showItems is true then only render ItemListComponent
              //if showItems is false then do not render ItemListComponent
               showItems &&  <ItemListComponent items={data.categories[0].category.items}/>
            }

            </div>
        </div>
    );

};

export default RestaurantCategoryComponent;
```

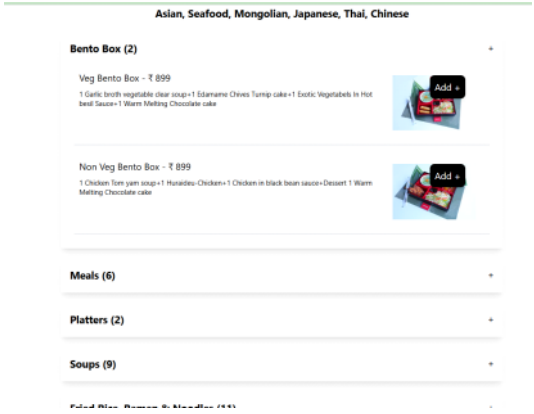
->ResturantMenuCard.js (from parent we passing showItems)

```
    return (

      <div className="text-center">
        <h1 className="font-bold my-6 text-2xl" >{resInfo.sections.SECTION_BASIC_INFO.name}</h1>
        <h3 className="font-bold text-lg" >{resInfo.sections.SECTION_BASIC_INFO.cuisine_string}</h3>

                    {
                      // iterating over categoriesItemsList array to render RestaurantCategoryComponent for each category.
          categoriesItemsList.map((category)=>{
                      // passing category data as prop to RestaurantCategoryComponent
                      // key is unique id for each category to help react identify which items have changed, are added, or are removed.
                    <RestaurantCategoryComponent
                    key={category.menu.id}
                    data={category.menu}
                    showItems={true}
                     />
          })}
      </div>
    );
  };
  export default RestaurantMenuCard;
```

->if showItem is true, all Category will be collapsed to show list of Items.

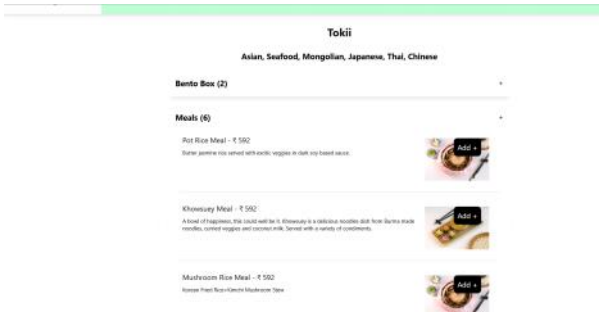
->I want to show list of items for only Oth index, others category must be closed.

```
    return (

      <div className="text-center">
        <h1 className="font-bold my-6 text-2xl" >{resInfo.sections.SECTION_BASIC_INFO.name}</h1>
        <h3 className="font-bold text-lg" >{resInfo.sections.SECTION_BASIC_INFO.cuisine_string}</h3>

                    {
                      // iterating over categoriesItemsList array to render RestaurantCategoryComponent for each category.
          categoriesItemsList.map((category, index)=>{
                      // passing category data as prop to RestaurantCategoryComponent
                      // key is unique id for each category to help react identify which items have changed, are added, or are removed.
                    <RestaurantCategoryComponent
                    key={category.menu.id}
                    data={category.menu}
                    showItems={index===0 && true}
                     />
          })}
      </div>
    );
  };
export default RestaurantMenuCard;
```



->->I want to show list of item for only first index, others category must be closed.

```
    return (

      <div className="text-center">
        <h1 className="font-bold my-6 text-2xl" >{resInfo.sections.SECTION_BASIC_INFO.name}</h1>
        <h3 className="font-bold text-lg" >{resInfo.sections.SECTION_BASIC_INFO.cuisine_string}</h3>

                    {
                      // iterating over categoriesItemsList array to render RestaurantCategoryComponent for each category.
          categoriesItemsList.map((category, index)=>{
                      // passing category data as prop to RestaurantCategoryComponent
                      // key is unique id for each category to help react identify which items have changed, are added, or are removed.
                    <RestaurantCategoryComponent
                    key={category.menu.id}
                    data={category.menu}
                    showItems={index===1 ? true:false}
                     />
          })}
      </div>
    );
  };
  export default RestaurantMenuCard;
```



->We Want at a time only accordion should be expanded when I click on that header.

Q)Can I modify localState variable of parent from children component ?
->it is not possible directly.
But I can be possible indirectly.
SO,we will create state variable at Parent Component.
Then we setVariable value, when our child component click on particular category.

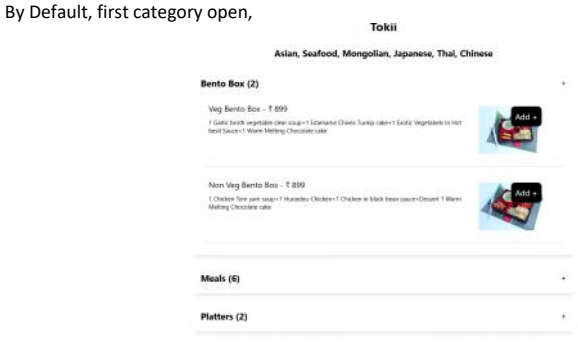
RestaurantMenuCard.js

```
const [showIndex,setShowIndex]= useState(0);//to control which category to show items for bydefault showing items for first category
//If we not provide any index, it will be collapsed by default for all categories.
```

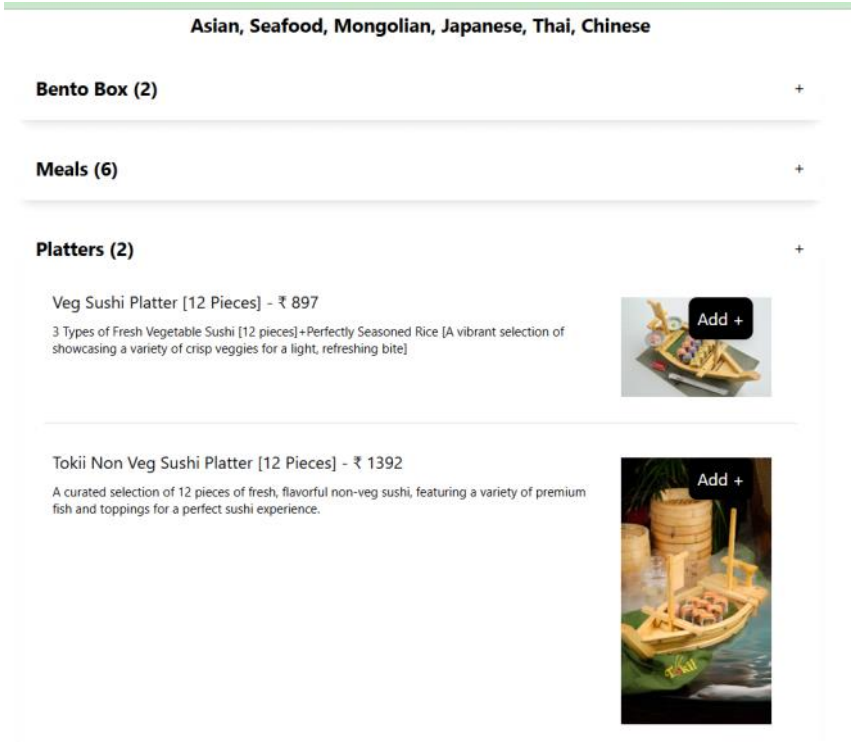
```
return (

    <div className="text-center">
      <h1 className="font-bold my-6 text-2xl" >{resInfo.sections.SECTION_BASIC_INFO.name}</h1>
      <h3 className="font-bold text-lg" >{resInfo.sections.SECTION_BASIC_INFO.cuisine_string}</h3>

                {
                    // iterating over categoriesItemsList array to render RestaurantCategoryComponent for each category.
          categoriesItemsList.map((category, index)=>{
              //Controlled component behavior
                    // passing category data as prop to RestaurantCategoryComponent
                    // key is unique id for each category to help react identify which items have changed, are added, or are removed.
                  <RestaurantCategoryComponent
                  key={category.menu.id}
                  data={category.menu}
                  showItems={index=== showIndex ? true:false}
                  // passing function to set showIndex in parent component
                  // when category is clicked in child component, it will call this function to set showIndex in parent component
                  // this will help to show items for the clicked category only
                  //example: if we click on 2nd category, it will set showIndex to 1 in parent component
                  // so, only items for 2nd category will be shown
                  setShowIndexToShowCategory={()=>setShowIndex(index)}
                  />
          ))
      </div>
    );
};
export default RestaurantMenuCard;
```

```
src > components > RestaurantCategoryComponent.js > RestaurantCategoryComponent

import ItemListComponent from "./ItemListComponent";

//passing data,showItems,setShowIndexToShowCategory as props from parent component
const RestaurantCategoryComponent =({data, showItems, setShowIndexToShowCategory})=>{

  console.log(data);
console.log(data.categories[0].category.items);
//I want to show items by giving control to parent component RestaurantMenuCard

const handleClick=()=>{
    console.log("Category clicked");
    //Call setShowIndex function passed from parent component
     setShowIndexToShowCategory();
};

    return(
        <div>
            <div className="w-6/12 mx-auto my-4 bg-gray-50 shadow-lg p-4" >
            <div  className="flex justify-between cursor-pointer" onClick={handleClick} >
              <span  className="font-bold text-lg" >{data.name} ({data.categories[0].category.items.length})</span>
               <span>+</span>
            </div>
            {   //If showItems is true then only render ItemListComponent
                //if showItems is false then do not render ItemListComponent
                 showItems &&  <ItemListComponent items={data.categories[0].category.items}/>
            }
            </div>
        </div>
    );
};
export default RestaurantCategoryComponent;
```

O/P=>
By Default, first category open,



Then if we click on any other category previous open category will be closed.



Read about Lifting state up from docs.
https://react.dev/learn/sharing-state-between-components#lifting-state-up-by-example

We have bydefault first category is open, but I am not able to close that category to show list of items.
How to fix that ?

Episode 11 Part 04
==========

Props Drilling

->As our Application is big, we have a lot of components.
Passing data between components is a very challenging.
A React is a one-way data stream means Data is passed from parents to Children.
So Data flow in One-direction from top to bottom.
Ex:-
RestaurantMenuCard Component can pass data to its child RestaurantCategoryComponent.

Ex:-
Let say I have dummy data in RestaurantMenuCard.
I want to pass data into ItemListComponent.

First we need to pass dummy data to child component of RestaurantMenuCard i.e. RestaurantCategoryComponent.
Then we need to pass that dummy data to ItemListComponent.

RestaurantMenuCard.js

```
const dummy= "Dummy Data";
```

```
const RestaurantMenuCard =()=>{

    return (

        <div className="text-center">
          <h1 className="font-bold my-6 text-2xl" >{resInfo.sections.SECTION_BASIC_INFO.name}</h1>
         <h3 className="font-bold text-lg" >{resInfo.sections.SECTION_BASIC_INFO.cuisine_string}</h3>

                    {
                      // iterating over categoriesItemsList array to render RestaurantCategoryComponent for each category.
                categoriesItemsList.map((category, index)=>{
                    //Controlled component behavior
                          // passing category data as prop to RestaurantCategoryComponent
                          // key is unique id for each category to help react identify which items have changed, are added, or are removed.
                        <RestaurantCategoryComponent
                        key={category.menu.id}
                        data={category.menu}
                        showItems={index=== showIndex ? true:false}
                        // passing function to set showIndex in parent component
                        // when category is clicked in child component, it will call this function to set showIndex in parent component
                        // this will help to show items for the clicked category only
                        //example: if we click on 2nd category, it will set showIndex to 1 in parent component
                        // so, only items for 2nd category will be shown
                        setShowIndexToShowCategory={()=>setShowIndex(index)}
                        dummy={dummy}
                        />
                ))
        </div>
    );
};
export default RestaurantMenuCard;
```

RestaurantCategoryComponent.js

```jsx
//passing data,showItems,setShowIndexToShowCategory as props from parent component
const RestaurantCategoryComponent =({data, showItems, setShowIndexToShowCategory,dummy})=>{

    console.log(data);
console.log(data.categories[0].category.items);
//I want to show items by giving control to parent component RestaurantMenuCard

const handleClick=()=>{
    console.log("Category clicked");
    //Call setShowIndex function passed from parent component
     setShowIndexToShowCategory();
};

    return(
        <div>
            <div className="w-6/12 mx-auto my-4 bg-gray-50 shadow-lg p-4" >
            <div className="flex justify-between cursor-pointer" onClick={handleClick} >
              <span  className="font-bold text-lg" >{data.name} ({data.categories[0].category.items.length})</span>
               <span>+</span>
            </div>
            {   //If showItems is true then only render ItemListComponent
               //If showItems is false then do not render ItemListComponent
                showItems &&  <ItemListComponent items={data.categories[0].category.items} dummy={dummy}/>
            }
            </div>
        </div>
    );
};
export default RestaurantCategoryComponent;
```

ItemListComponent.js

```jsx
src > components > JS ItemListComponent.js > [@] ItemListComponent > items.map() callback
  const ItemListComponent = ({items,dummy})=>{
      console.log(items);
      console.log(dummy);

      return(
            <div>
             {items.map((e)=>(

                 <div key={e.item.id}  className="p-2 m-2  border-gray-200 border-b-2 text-left flex justify-between" >
                     <div  className="w-9/12" >
                  <div className="py-2">
                     {/* show item name */}
                         <span >{e.item.name}</span>
                         {/* to show item price, if price is not there show default price */}
                         <span> - ₹ {e.item.price!==0 ? e.item.price : e.item.default_price }</span>

                         </div>
                         {/* to show item description */}
                     <p className="text-xs" >{e.item.desc}</p>
                     </div>

                     {/* if Image not there for particular item we will not show image for that,
                      so we write condition here  so that if image not available we will display add Button for that.*/}
                      {e.item.item_image_url!==undefined ?
                         <div className="w-3/12 p-4">
                      <div className="absolute" >   <button className="p-2 mx-16 bg-black text-white shadow-lg rounded-lg" > Add +</button></div>
                      <div>  <img src={e.item.item_image_url} className="w-full"/></div>
                      </div>
                      : <div className="w-3/12 p-4"> </div>
                     } </div>

      )))
```

But Think of Big Level applications which has so many level of nesting.
If we need to pass data from parent to subchild.
Then first we need to pass data to child then child to its child.
It is not good way.
Middle child just serving data not using it like in **RestaurantCategoryComponent**.
Just passing props like this is called props drilling.

State & Props is a very important essential of react.
Every Component have props & state.
But we should avoid props drilling.

If we are passing data from parent to child upto 2-3 level is ok.
But what if my data is at somewhere & how can I access that data at somewhere ?
In Large application, it is very common scenario where sometimes we need to have some kind of global data that I can access anywhere
Whatever nested level I am, whether I am in Header or in any component.
For Thar React gives us a superpower & that SuperPower is called as **React Context**.

So, React gives us access to something known as React context.
While we can use context, we can just avoid props drilling.

We use context which is like a global place where our data is kept & anybody can access it. This is React Context.

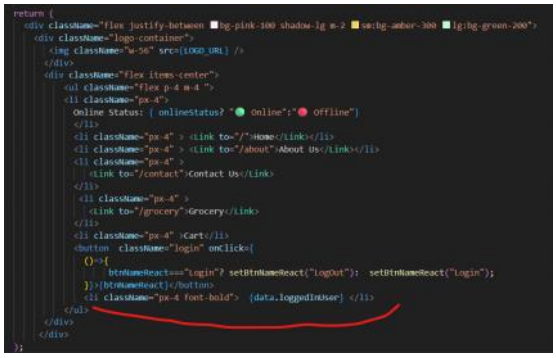
->
EX:-1) we need logged in user data in anywhere in our application.
EX:-2) To change theme of our app from dark to white.
So we store info in context For this.

Context is like Global Things.
Create Context inside the Utils.
->React gives access to utility function to create context.
->We can create any no. of context.

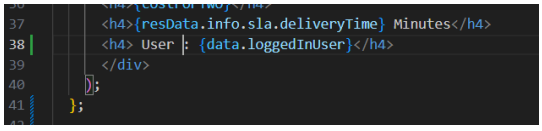
```
src > utils > JS UserContext.js > default
import { createContext } from "react";


const UserContext= createContext({

    loggedInUser:"Defauult  User",

});

export default UserContext;
```

Now, we can access loggedInUser context anywhere in our app.

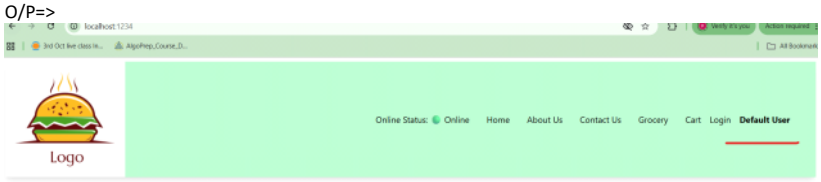
1)Use UserContext in Header Component.

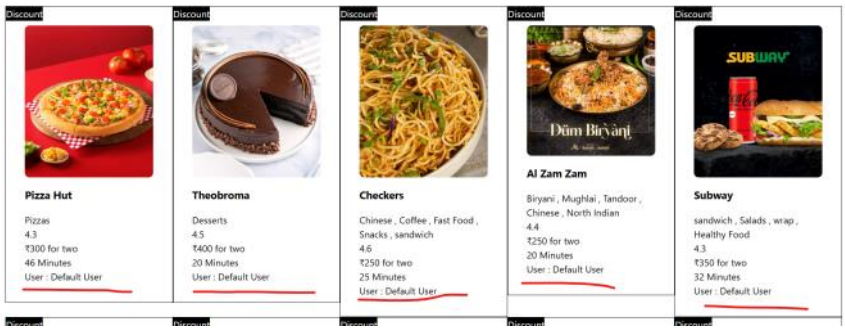


2)Use Context at RestauRantCard Component.



O/P=>





Q)Can I Use React Context in Class based Component ?
=>
As we know in class based component we do not have hooks.
So, we can not use UseContext hook.

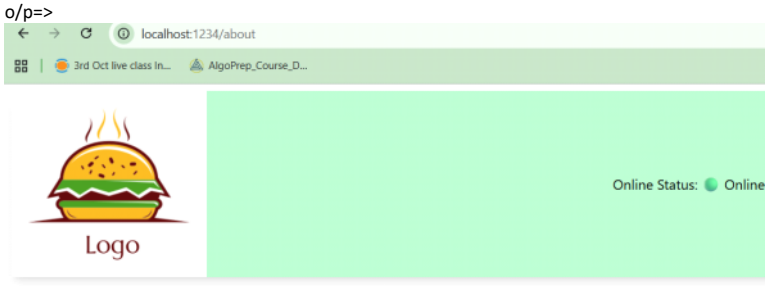
How to access that ?
=>
1)Import the Context.

```
import UserContext from "../utils/UserContext";
```

```
return(
    <div>
    <h1>About Us Page</h1>
    <div>
        Logged in User is : <UserContext.Consumer>
            {
            // to access context value use function which returns data
            //it is callback function which receives context value as argument
                ({loggedInUser})=>  <h1 className="text-xl font-bold" >{loggedInUser}</h1>

            }
        </UserContext.Consumer>
    </div>
    <p>This is Namaste React Live Course 2023</p>
    <p>We are learning React JS</p>
      <User name={"First"}/>
    <Userclass name={"First (class)"}  location={"Nagpur class"}/>
</div>
);
```

Where <UserContext.Consumer> </UserContext.Consumer> is an Component.
We need to pass callBack function inside the JSX to access context data.

o/p=>



About Us Page
Logged in User is :
Default User
This is Namaste React Live Course 2023
We are learning React JS

How to Modify the Context ?
=>
Lets say we do Authentication for identify user & if valid user, we show that username on web page,
We do similar.
In Root js page i.e. App.js we make API call to identify the user & store tha user in stateVariable
& then assign that value to Global Context variable using Provider which is given by react.

APP.js

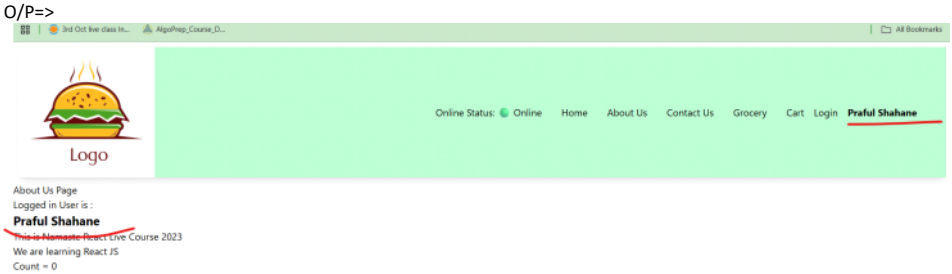
```
import  UserContext from "./utils/UserContext";
```

```
    const [userName,setUserName]= useState();
  //Authetication code written to identify the user
   useEffect(()=>{
     //Make API call to get the user info by sending userName & password
     const data={
      name:"Praful Shahane"
     }

     setUserName(data.name);

   },[]);
```

Wrap our whole app code inside the UserContext.Provider & give updated value to context variable.

```
    return (
      <UserContext.Provider value={{loggedInUser : userName}} >
    <div className="app">
      <HeaderComponent />
      <Outlet />
    </div>
    </UserContext.Provider>);
  };
```

O/P=>



=>If we wrap only Header Component, the it show updated value in HeaderComponent only.

```
    return (

    <div className="app">
      <UserContext.Provider value={{loggedInUser : userName}} >
      <HeaderComponent />
        </UserContext.Provider>
      <Outlet />
    </div>
  );
  };
```

O/P=>

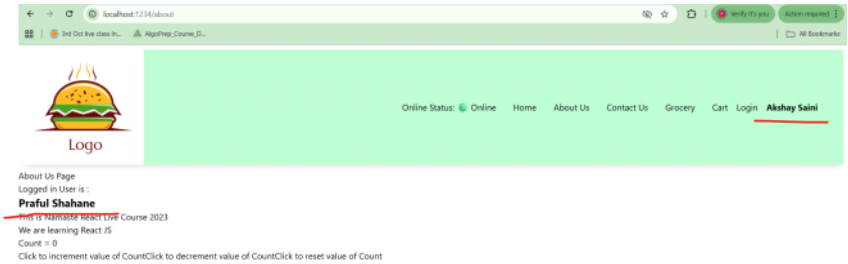
=>Generally we wrap UserContext.Provider inside the whole app.

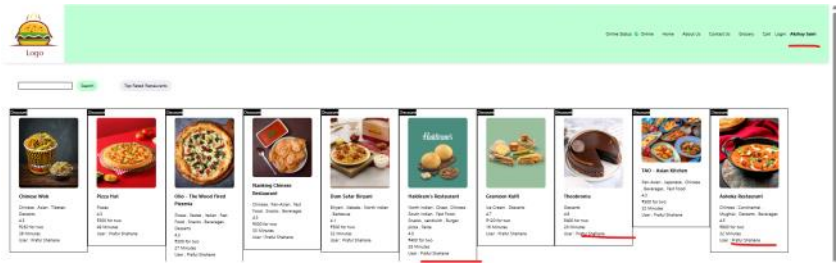
Q)Can we wrap same UserContext with diff value?
Is this valid code ?

```
    return (
      <UserContext.Provider value={{loggedInUser : userName}} >
    <div className="app">
      <UserContext.Provider value={{loggedInUser : "Akshay Saini"}} >
      <HeaderComponent />
  </UserContext.Provider>
      <Outlet />
    </div>
          </UserContext.Provider>
  );
```

=> yes,

```
  return (
    //Outer has "Default User".
    <UserContext.Provider value={{loggedInUser : userName}} >
    {/* Here value is Praful Shahane */}
  <div className="app">
    <UserContext.Provider value={{loggedInUser : "Akshay Saini"}} >
    {/* Here only Header component will have value "Akshay saini" */}
    <HeaderComponent />
  </UserContext.Provider>
    <Outlet />
  </div>
          </UserContext.Provider>
  );
```





So, it is depend on we provide the UserContext.

=>
Create one input box in body. Whatever I write in input box, it update my context with same data.

1)Create one Input box

```
<div className="flex m-4 p-4">
 <label>UserName : </label>
 <input className="border border-black p-2" value={loggedInUser} onChange={(e)=> setUserName(e.target.value)} />
</div>
```

Initial value is taking from loggedInUser variable of Context.
As soon as we type, our Onchange event is called & then we setting setUserName value also.

App.js

```
const [userName,setUserName]= useState();
```

```
42
43    return (
44      <UserContext.Provider value={{loggedInUser : userName,setUserName}} >
45      <div className="app">
46        <HeaderComponent />
47        <Outlet />
48      </div>
49          </UserContext.Provider>
50    );
```

As we know, userName is local state variable & setUserName is function to setValue of UserName.
We can pass that setUserName function into UserContext.provider also, so that I can set UserName's new value from anywhere.
So we pass setUserName function as well in UserContext.

Now, in Body.js Component,
We import UserContext.

```
8         const data= useContext(UserContext);
9         console.log(data);
0
1         const { loggedInUser,setUserName } = useContext(UserContext);
2         console.log(loggedInUser,setUserName);
3
```

o/p of data :

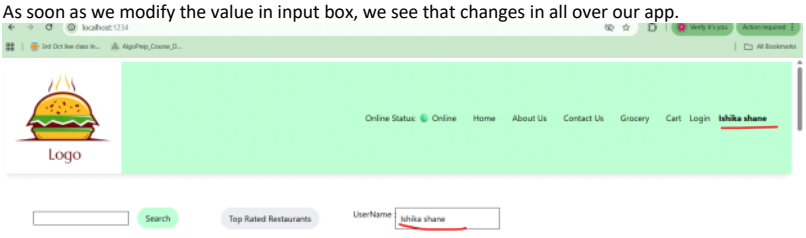
```
▸ {loggedInUser: 'Praful Shahane', setUserName: ƒ}
```

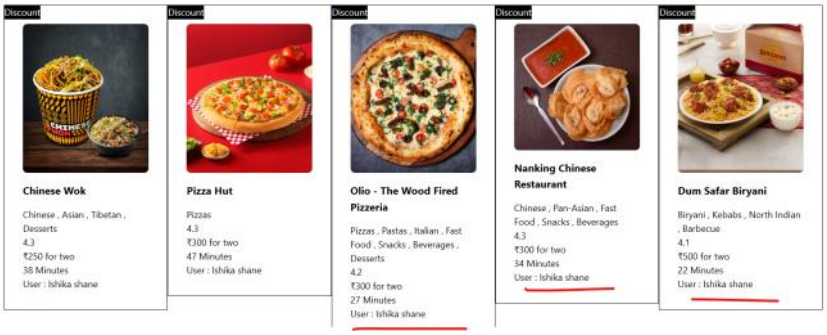
Use above value into input tag,

```
<div className="flex m-4 p-4">
 <label>UserName : </label>
 <input className="border border-black p-2" value={loggedInUser} onChange={(e)=> setUserName(e.target.value)} />
</div>
```

o/p=>
Intially we get Username from API is Praful Shahane.

As soon as we modify the value in input box, we see that changes in all over our app.





=>

UserContext.js

```
const UserContext= createContext({

    loggedInUser:"Default  User",

});

export default UserContext;
```

If we pass this value into App.js

```
const AppLayout =()=>{

    console.log(<BodyComponent />);

        const [userName,setUserName]= useState();
      //Authetication code written to identify the user
       useEffect(()=>{
         //Make API call to get the user info by sending userName & password
         const data={
          name:"Praful Shahane"
         }

         setUserName(data.name);

       },[]);


const datanew="Data User";
```

```
    const datanew="Data User";

    return (
      <UserContext.Provider value={{loggedInUser : userName,setUserName , datanew}} >
    <div className="app">
      <HeaderComponent />
      <Outlet />
    </div>
          </UserContext.Provider>
  );
```

BodyComponent.js

```
    const data= useContext(UserContext);
    console.log(data);
```

o/p=>

```
▼ {loggedInUser: 'Praful Shahane', datanew: 'Data User', setUserName: ƒ} ℹ
    datanew: "Data User"
    loggedInUser: "Praful Shahane"
  ▶ setUserName: ƒ ()
  ▶ [[Prototype]]: Object
```

We can build large scale app using react Context also.
But redux is also do the same activity for scalable applications.
Redux is diff from react.
We learn redux, in next lecture.

++

EPISODE 12: Let's Build Our Store

++

In this, episode we learn about Redux.
We are going to see how we can manage state of application using Redux.
how we can manage data of application using Redux.

Redux works in the data layer.
UI layer & date layer works in sync & build our own react app.

Episode 12 Part 01
==========
Many Companies using Redux from day one.
But Redux is not mandatory in our application.
A lot of companies build projects & they use Redux along with React no matter what the state is
They don't think about whether do we actually need redux or not.
->When you are building small scale & large scale application, we don't need redux,
But if you are building large scale application where the data is heavily used
Where a a lot of read & write operations are happening in UI layer in our react application.
There are a lot of components & a lot of data transfer between components & application is growing huge,
Then using redux makes sense.
->If we build small app using react with few components we don't need redux.
->Any application which is build using redux can be built without using Redux.
So, Redux is not mandatory used it only when It is required.

->Redux & React are different libraries.
->We install Redux library to use in our React application.
->Redux is not the only library to manage the state. There are other library also we can used with React like
Zustand which is used for state managing.

Advantage of Redux:-
1)When we building large scale applications, redux offers a great solution for handling the data, managing store.
Primarily used for handling state of the application.
2)When we use redux our application is easier to debug.
we have redux dev tools. It helps us to debug our applications.

Episode 12 Part 02
==========
->Redux documentation link :- https://redux.js.org/
->A JS library for predictable and maintainable global state management.
->Redux offers two libraries.
1)React-redux (it is bridge between React & redux)
2)Redux toolkit(it is newer way of writing redux).

In Older days, we used to have diff way of writing redux but with modern web development redux simplifies itself a lot.
So, we will using Redux toolkit with React-Redux.
We will not be using traditional vanilla redux.

So, if we are building project today use Redux-toolkit.

Redux ToolKit(RTK)

->The **Redux Toolkit** package is intended to be the standard way to write Redux logic.
->It was originally created to help address three common concerns about Redux:
a)"Configuring a Redux store is too complicated"
b)"I have to add a lot of packages to get Redux to do anything useful"
c)"Redux requires too much boilerplate code".

Episode 12 Part 03
==========
IN our Food application, we build cart flow.
When I click on particular restaurant, we get list of menu, we click on add button.
As soon as we click on add button, our data must be shown in cart.
We built cart page also.
To Store cart information, we will be using our redux store.
This feature we are going to build using redux.

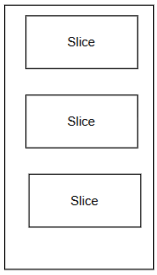
Before we code, first we learn about Redux.
If I say redux now, means Redux-toolkit.

Redux Architecture

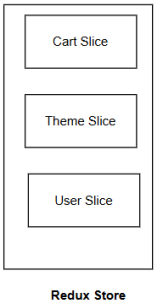
Redux store is an big JavaScript object which is kept in central global place.
Any Components can access that object inside the React application.
It can write or Read data from store.
We keep major data of our application into this redux store.

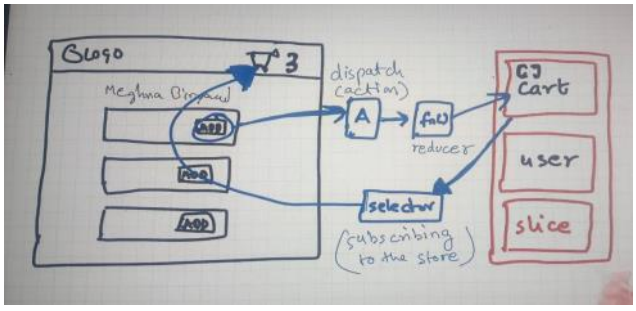
-> But is it good practice to store all data inside the one whole object ?
Yes. It is absolutely fine. So that our redux store does not becomes very big, very clumsy.
We have something known as slices inside the redux store.

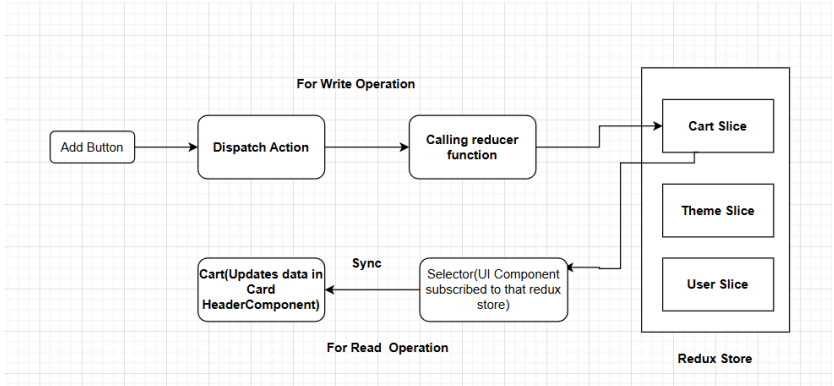
Assume slices are a small portion of redux store.
We can create multiple slices inside the redux store.



Q)What is need of slices & what are these slices ?
->
Let make example of our Food ordering app,
To keep data separate, we make logical partitions & these logical partitions are slices.
Assume, if I want to add cart data into redux store, we will create a separate slice for cart data.
Suppose if I want to keep logged in userInfo inside our redux store, so we will create a user slice kind of thing.







->Intially this cart slice can be empty array. But later we can put data inside it can just modify the card slice data inside it.
Q)When I click on add Button, how does the data goes inside the card slice in redux store ?
->It is not simple. We can't directly add data to the card slice.
Redux says that we cannot modify our cart slice data directly.
There is a way we can do that.
->If I click on add button, It will have to dispatch an action.
After dispatch an action, it calls the function.
Then that function modifies the cart slice in redux store.
That function is known as reducer.
so, **When we click on add button, it dispatches an action which calls the reducer function & this reducer function updates the cart slice of redux store.**
Now cart slice will have some data inside it.
This is how we write data into cart slice of redux store.

To Read data from cart slice to show on UI Cart

For this we use **selector**.
->We will use selector to read data from redux store.
Then selector will modify our react component.
->When we use selector so this phenomena is known as **Subscribing to the store**.
So, our Header Component is subscribed to the Store it means both are sync.

So, if data inside store changes it will update the Header component automatically.
Using selector, we subscribe.

This is how whole flow works.

Q)Why this is so complicated ?
->It is complicated to update the cart slice of redux store.
It makes our life easier.

We study theory.
Now, we will write code for above requirement.

Episode 12 Part 04
==========
We study theory.
Now, we will write code for above requirement.
We will do following steps.

```
# Redux Toolkit
  -STEPS
  ---------------
  - Install @reduxjs/toolkit +  react-redux
  -Build our own store.
  -Connect our store to our app.
  -Create slice(Cart slice)
  -dispatch(action)
  -Selector
```

1) Install redux/toolkit & react-redux.
npm install @reduxjs/toolkit

npm i react-redux

Package.json

```
"dependencies": {
  "@reduxjs/toolkit": "^2.11.0",
  "@tailwindcss/postcss": "^4.1.17",
  "react": "^19.1.1",
  "react-dom": "^19.1.1",
  "react-redux": "^9.2.0",
  "react-router-dom": "^7.9.5",
  "tailwindcss": "^4.1.17"
},
```

2)Now build our own store.
Create one **appStore.js** js file under until folder.
appStore.js

```
src > utils > JS appStore.js > [⊙] appStore
1   import { configureStore } from "@reduxjs/toolkit";
2
3   const appStore = configureStore(
4       {
5
6       }
7   );
8
9   export default appStore;
```

Now provide this store to the application.
Go to Main file which is App.js
This is the root of our application as below.

```
return (
  <UserContext.Provider value={{loggedInUser : userName,setUserName , datanew}} >
<div className="app">
  <HeaderComponent />
  <Outlet />
</div>
      </UserContext.Provider>
);
};
```

So, we need to use provider which is given by react-redux.

```
import { Provider } from "react-redux";
```

Now, wrap our root component inside the Provider which is given by react-redux.
& store is our appStore config.

```
15  import { Provider } from "react-redux";
16  import appStore from "./utils/appStore";
```

```
    return (
      <Provider store={appStore}>

      <UserContext.Provider value={{loggedInUser : userName,setUserName , datanew}} >
    <div className="app">
      <HeaderComponent />
      <Outlet />
    </div>
          </UserContext.Provider>
          </Provider>
  );
  };
```

->we can specify Provider to our specific component also.
->Now create cartSlice.

```
import { createSlice } from "@reduxjs/toolkit";


const cartSlice= createSlice({
    name: 'cart', // Name of the slice
    initialState:{
        items:[] // Initial state of the cart
    },
    reducers:{
        //reducers object contains multiple actions.
        //for Each action we define a  reducer function.

        //addItem is action creator
        //so, addItem is an reducer function whose name is addItem action.
        addItem: (state , action)=>{
            state.items.push(action.payload); // Add item to cart
            /*
            */
        },
        // removeItem is action creator
        removeItem: (state)=>{
            // Logic to remove item from cart
            state.items.pop(); // Remove last item from cart
        },
        // clearCart is action creator
        clearCart: (state)=>{
                    state.items.length=0; // Clear all items from cart so no need of action.payload
                    //state=[];  not working
        }
    }
});
```

```
/*
O/P=>
    cartSlice is an object with following properties:
{
  actions: {},
  reducer:
  }
*/

// Exporting action creators
export const {addItem, removeItem, clearCart} = cartSlice.actions;

// Exporting  reducer.
export default cartSlice.reducer;
```

Now, we need to add cartSlice into our Redux store i.e appStore.js

```
src > utils > JS appStore.js > ...
import { configureStore } from "@reduxjs/toolkit";
import cartReducer from "./cartSlice.js";

const appStore = configureStore(
    {
        reducer:{
            cart : cartReducer,
            //we can add more reducers here like user : userReducer
        }
    }
);

export default appStore;
```

Now let say can we read data from our app.
We need to subscribe the store using selector.

Let we add dummy items intially in cart.
Show on UI.

cartSlice.js (add dummy items)

```
const cartSlice= createSlice({
    name: 'cart', // Name of the slice
    initialState:{
        items:["burger","pizza"] // Initial state of the cart
    },
    reducers:{
        //reducers object contains multiple actions.
```

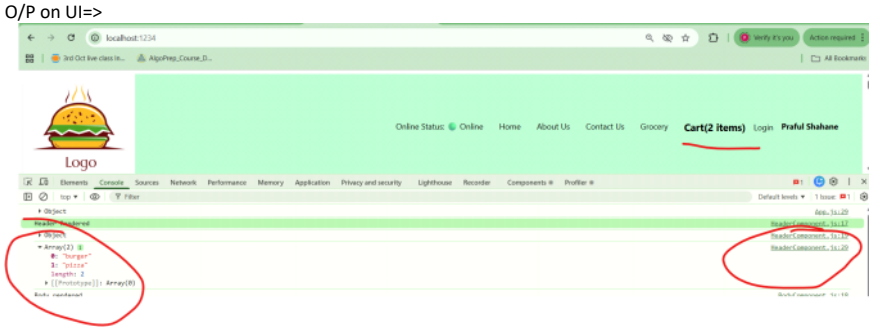
HeaderComponent.js

```
import { useSelector } from "react-redux";
```

```
//Selector is hook inside the react,
//Subscribe to the redux store using useSelector hook.
const cartItems = useSelector((store)=> store.cart.items);  // we can access the redux store state inside the component.
console.log(cartItems); // {items: Array(2)} items: ['burger','pizza']
```

```
<li className="px-4" >
  <Link to="/grocery">Grocery</Link>
</li>
<li className="px-4 font-bold text-xl"  >Cart({cartItems.length} items)</li>
<button  className="login" onClick={
  ()=>{
```

O/P on UI=>



->Now , let make dummy items as empty.
cartSlice.js

```
const cartSlice= createSlice({
    name: 'cart', // Name of the slice
    initialState:{
        items:[] // Initial state of the cart
    },
```

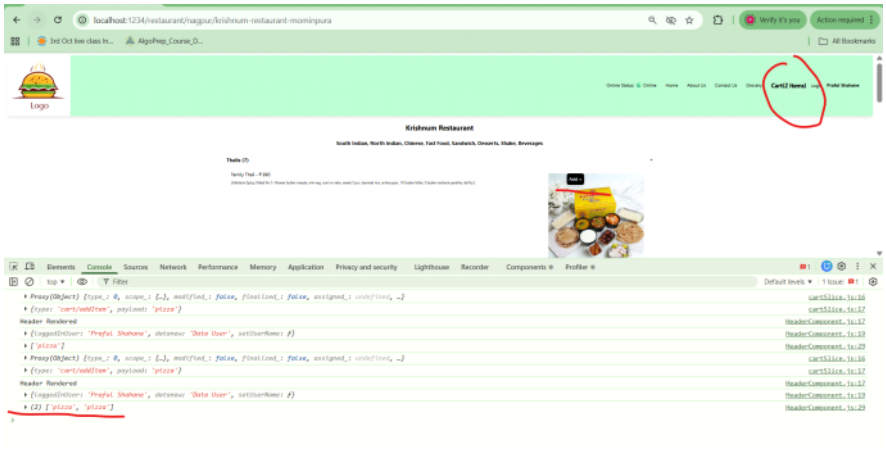
Let us use real world items.
Once we click on add button, we must add items into cart.
1)create onClick event on addButton of menu card.

```
<div className="absolute" >
  <button className="p-2 mx-16 bg-black text-white shadow-lg rounded-lg"
  onClick={
     handleAddItemToCart
  }
  > Add +</button></div>
<div>  <img src={e.item.item_image_url} className="w-full"/></div>
</div>
```

2)write logic once we click on button, it will dispatch the action to cart.
We will use useDispatch hook given by react-redux.
Then import our addItem action from CartSlice.js
Pass our item name as argument to the addItem action.

```
import { useDispatch } from "react-redux";
import { addItem } from "../utils/cartSlice";

const ItemListComponent = ({items,dummy})=>{

    //dispatch function is used to dispatch an action to the redux store.
    //given by react-redux library.
    const dispatch= useDispatch();

const handleAddItemToCart=()=>{
    //dispatch an action to add item to cart
    dispatch(addItem("pizza"));//   payload is pizza string
    /**
     * O/P=>
     * action={
     *    type: 'cart/addItem',
     *    payload: 'pizza'
     * }
     */
};
```

o/p=>
As soon as we click on Add button, our item will be added to the cart.



Q) What is diff when we write function like below
=>

onClick={ handleAddItemToCart }
onClick={ ()=> handleAddItemToCart(item) }
onClick={ handleAddItemToCart(item) }

=>To pass an actual object,just pass actual object.

```
<div className="w-3/12 p-4">
<div className="absolute" >
<button className="p-2 mx-16 bg-black text-white shadow-lg rounded-lg"
onClick={()=> handleAddItemToCart(e) }
> Add +</button></div>
<div> <img src={e.item.item_image_url} className="w-full"/></div>
</div>
```

```
const handleAddItemToCart=(e)=>{
  //dispatch an action to add item to cart
  // dispatch(addItem("pizza"));//   payload is pizza string
  //pass specific item to be added to cart
  dispatch(addItem(e));
  /**
   * O/P=>
   * action={
   *    type: 'cart/addItem',
   *    payload: 'pizza'
   * }
   */
};
```

=>Now, lets build cart page. Once we click on cart link it go to cart page & show details of cart items.
1)Create CartComponent.js

```
const  CartComponent =()=>{
    return(
        <div className=" text-center m-4 p-4" >
            <h1 className=" text-2xl font-bold" >Cart</h1>
        </div>
    );
}

export default CartComponent;
```

2)add url /cart in app.js

```
},{
  path:"/cart",
  element:<CartComponent/>
}
```

3)make our Cart as Link in HeaderComponent.

```
<Link to="/grocery">Grocery</Link>
</li>
<li className="px-4 font-bold text-xl"  >
<Link to="/cart"> Cart({cartItems.length} items)</Link>
</li>
```

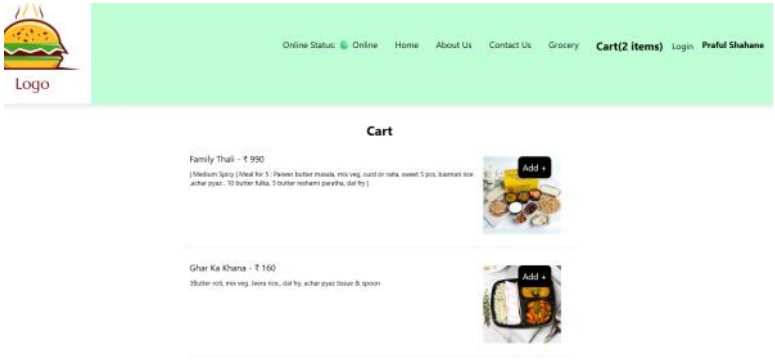
O/P on UI=>

Now, how we will read Cart from store.
First we need to subscribe to the store.
So that we can display cart data into our CartComponent.js

We need to show ItemData here, so we can use ItemListComponent here.
& get cartItems data from cart store using selector.

```
const CartComponent =()=>{
    //to show item present inside the cart in CartComponent.
    const cartItems= useSelector((store)=> store.cart.items);
    console.log(cartItems);



     return(
        <div className=" text-center m-4 p-4" >
            <h1 className=" text-2xl font-bold" >Cart</h1>
              <div className=" w-6/12 m-auto">

                <ItemListComponent items={cartItems} />

              </div>

        </div>
     );
}
```

O/P ON UI=>
As soon as we click on Item list from menu we can see that selected item in Cart.

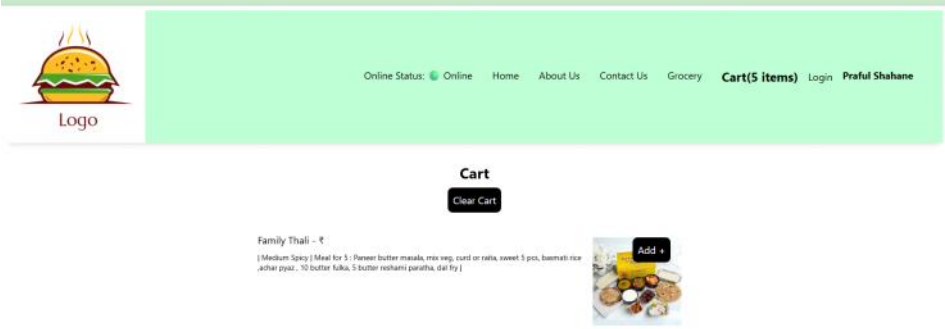


-> we will develop one clear cart button to clear the data from cart.
If cart empty show message.

```
import { useSelector } from "react-redux";
import ItemListComponent from "./ItemListComponent";
import { use } from "react";
import { clearCart } from "../utils/cartSlice";
import { useDispatch } from "react-redux";

const CartComponent =()=>{

   //Subscribe to the redux store using useSelector hook.
   //so that we can access the redux store state inside the component.
   //to show item present inside the cart in CartComponent.
   const cartItems= useSelector((store)=> store.cart.items);
   console.log(cartItems);


   //get the dispatch function from react-redux library.
   const dispatch = useDispatch();


const handleClearCart=()=>{
    //dispatch an action to clear the cart
    dispatch(clearCart());
};
```

```
      <div className=" text-center m-4 p-4" >
          <h1 className=" text-2xl font-bold" >Cart</h1>
            <div className=" w-6/12 m-auto">
            <button className="p-2 m-2 bg-black text-white rounded-lg"
              onClick={
                  handleClearCart
              }
            >Clear Cart</button>
             {/* if cart is empty show message */}
             {cartItems.length===0  && (<h1>Your Cart is Empty add ietms to teh cart</h1>)}
              <ItemListComponent items={cartItems} />

            </div>

      </div>
   );
}

export default CartComponent;
```

O/P on UI=>
Once we click on Clear Cart all item will be removed from cart.



Episode 12 Part 05
==========

->Whenever we are doing selector make sure we are subscribing to the right portion of the store.
Using this we can optimize the performance.
If we not subscribe to the right portion of the store, it will be big performance loss.
why ?
->
Right portion is I need only item from cartStore, I will write code for it is

```
const cartItems= useSelector((store)=> store.cart.items);
console.log(cartItems);
```

But to get same items from cartStore, if I write a code like this,

```
const store= useSelector((store)=> store);
const cartItems= store.cart.items;
console.log(cartItems);
```

But this code make our performance loss.

->
const store= useSelector((store)=> store);
This code means subscribing to whole redux store.
Anything changes inside the store, our CartComponent will get to Know.
i.e. store variable will be updated with anything changes.
But I don't want to subscribe to whole store.
I want updates from Cart Store only.
So, we only subscribe to Cart store only using below code.

```
const cartItems= useSelector((store)=> store.cart.items);
console.log(cartItems);
```

So, never subscribe to whole store.
Subscribe to specific portion of required store.
So, when my cart's Items changes then only our cartItems variable will update.

Episode 12 Part 06
==========

2nd Mistake

->There is a lot of confusion between reducer & reducers.
->Whenever we creating appStore or reduxStore then we use keyword as **reducer**.

```
import { configureStore } from "@reduxjs/toolkit";
import cartReducer from "./cartSlice.js";

const appStore = configureStore(
    {
        reducer:{
            cart : cartReducer,
            //we can add more reducers here like user : userReducer
        }
    }
);

export default appStore;
```

This reducer can have multiple small reducer.
->When we are writing slice, we create multiple reducers.
->When we are exporting, we are exporting just one reducer from it.
Reducer can be a combination of diff small reducers.

```
utils > cartSlice.js > cartSlice > reducers
import { createSlice, current } from "@reduxjs/toolkit";

const cartSlice= createSlice({
    name: 'cart', // Name of the slice
    initialState:{
        items:[] // Initial state of the cart
    },
    reducers:{
        //reducers object contains multiple actions.
        //for Each action we define a  reducer function.
        //addItem is action creator
        //so, addItem is an reducer function whose name is addItem action.
        addItem: (state , action)=>{
            console.log(state); //Proxy(Object) {type_: 0, scope_: {…}, modified_: false, finalized_: false, assigned_: undefined, …}
            console.log(action);  //{type: 'cart/addItem', payload: 'pizza'}
            //mutating the state directly because immer library is used internally by redux toolkit.
            //i.e. modifying original state is allowed in redux toolkit.
            //Redux toolkit used immer behind the state.
           state.items.push(action.payload); // Add item to cart
            /*
            */

            // //Vanilla redux way to update the state immutably
            // const newState=[...state];
            //  newState.items.push(action.payload);
            //  return newState;

        },
        // removeItem is action creator
        removeItem: (state,action)=>{
            // Logic to remove item from cart
            state.items.pop(); // Remove last item from cart
        },
```

```
        // clearCart is action creator
        clearCart: (state,action)=>{
                //    state.items.length=0; // Clear all items from cart so no need of action.payload
                  //state=[];  not working bcz here we are not modifying the state, we are just adding refrence of it.

                //    console.log(state); //we get proxy object of reduc
                //     console.log(current(state)); //to print object of redux
                //    state=[];
                //    console.log(state);
                return {items:[]};

        }
    }
});
/*
O/P=>
    cartSlice is an object with following properties:
{
  actions: {},
  reducer:
  }
*/
// Exporting action creators
export const {addItem, removeItem, clearCart} = cartSlice.actions;

// Exporting  reducer.
export default cartSlice.reducer;
```

Episode 12 Part 07
==========

->when we used to write vanilla redux,there was a big problem with state here.
Redux give us warning that Don't mutate state in older redux.
So, we should not mutate state in vanilla redux in older version or older redux.

state.items.push(action.payload); // this code is prohibited in vanilla redux.
But in new redux, we can directly modify the state.
In Vanilla redux, we used to create a copy of state variable.
Then we used to push.

```
        addItem: (state , action)=>{
            console.log(state); //Proxy(Object) {type_: 0, scope_: {…}, modifi
            console.log(action);  //{type: 'cart/addItem', payload: 'pizza'}
            //mutating the state directly because immer library is used inter
            //i.e. modifying original state is allowed in redux toolkit.
        //    state.items.push(action.payload); // Add item to cart
            /*
            */

            //Vanilla redux way to update the state immutably
            const newState=[...state];
            newState.items.push(action.payload);
            return newState;

        },
```

->**state.items.push(action.payload);** // this is new way of writing redux (muting state).
no need to return anything in new redux. Redux take care of everything in new redux.
But in old redux, we need to return new state.

->If we are mutating state in new redux version, what is redux doing behind the scene ?
=>
Redux is creating an immutable state behind the scene in older redux way.
But it is not asking for developer to write it.
Redux is uses immer library. **Immer** library is like finding difference between original state & mutated state then gives us back new state which us immutable state, new copy of state.
Immer is a tiny package that allows you to work with immutable state in a more convenient way.

```
clearCart: (state,action)=>{
        state.items.length=0; // Clear all items from cart so no need of action.payload
        //state=[];  not working bcz here we are not modifying the state, we are just adding refrence of it.
    }
```

state=[]; not working bcz here we are not modifying the state, we are just adding refrence of it.
So, to make it empty we have mutate the state, we can mutate by using state.items.length===0.

State is an local variable which has original state value.

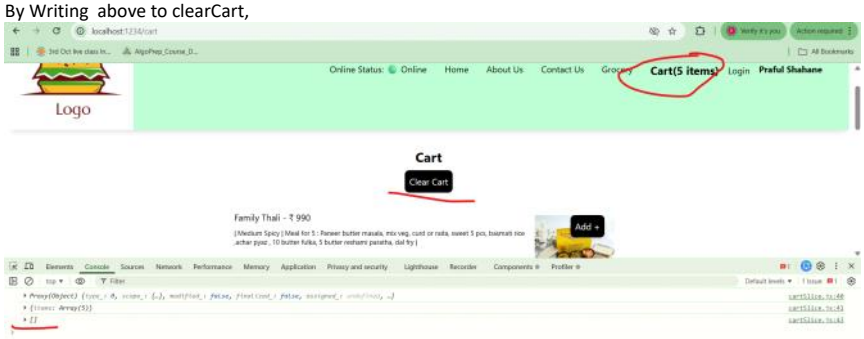
```
//original state =["pizza"]
clearCart: (state,action)=>{
    console.log(state); //[pizza]
    state=[];
     console.log(state); //[]
     //it is making our state local variable empty but oirginal state value remain the same as ["pizza"].
    }
```

->To print the object of redux, we use current which is from reduxtoolkit.

```
import { createSlice, current } from "@reduxjs/toolkit";
```

```
clearCart: (state,action)=>{
        //    state.items.length=0; // Clear all items from cart so no need of action.payload
          //state=[];  not working bcz here we are not modifying the state, we are just adding refrence of it.

          console.log(state); //we get proxy object of reduc
           console.log(current(state)); //to print object of redux
          state=[];
          console.log(state);


}
```

By Writing above to clearCart,



In console we get empty object but our original state object remain as it is due to this, our cart items from UI is not clear.

```
clearCart: (state,action)=>{
        //    state.items.length=0; // Clear all items from cart so no need of action.payload
          //state=[];  not working bcz here we are not modifying the state, we are just adding refrence of it.

          console.log(state); //we get proxy object of reduc
           console.log(current(state)); //to print object of redux
          state=[];
          console.log(state);


}
```

But instead of above code if we write belowe code, it will actually modifies the original state.

```
// clearCart is action creator
clearCart: (state,action)=>{
            state.items.length=0; // Clear all items from cart so no need of action.payload
            //state=[];  not working bcz here we are not modifying the state, we are just adding refrence of it.

          //    console.log(state); //we get proxy object of reduc
          //     console.log(current(state)); //to print object of redux
          //    state=[];
          //    console.log(state);

}
```

ReduxToolKit says if we want to clear something, you have to either mutate the state. (state.items.length=0;)
internally it make original state to empty.
Or return a new state.

```
// clearCart is action creator
clearCart: (state,action)=>{
        //    state.items.length=0; // Clear all items from cart so no need of action.payload
          //state=[];  not working bcz here we are not modifying the state, we are just adding refrence of it.

          //    console.log(state); //we get proxy object of reduc
          //     console.log(current(state)); //to print object of redux
          //    state=[];
          //    console.log(state);
          return {items:[]};

}
}
```

Whatever we return from reducer it will modifies the original state.
Ex:-

```
//original state =["pizza"]
clearCart: (state,action)=>{
  return {items:[]};   // new enpty array we returning which modifies the original state.
```

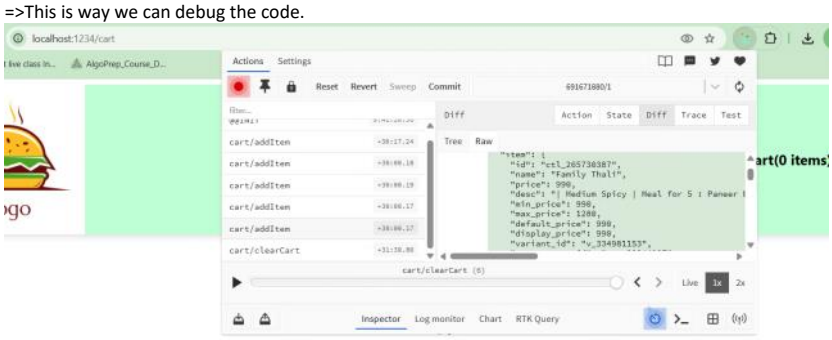
}
So, originalState =["pizza"] is replaced by [] empty array.

Episode 12 Part 08
==========

Redux Dev Tools

Redux help us in debugging a lot.
Install redux dev tools chrome extension.

=>This is way we can debug the code.



Read about RTK query is way of fetching data.
Middleware & thungs
++
EPISODE 13: Time for Test
++

In this, episode we learn how to write testcases for react application.

Episode 13 Part 01
==========
->we only concerned about developer testing.
What diff types of testing developer can do ?
1)Manual testing :-
We will test our feature we developed manually.
But it is not very efficient.
Bcz if we make changes in code are we able test again our app ?
No. very time consuming.

In React, we have a lot of components & all are interdependent on each other.
Even if we make one line of code it may create bug in our application.
So, testing is very important even if we make small changes.

2)Writing testcases:-
By writing testcases we test our code.

Diff types of testing we can do in React :-
Type of Testing (Developer can do)
 -Unit Testing
 -Integration testing
 -End to End Testing(E2E Testing)

  -Unit Testing means we test our React Component in isolation or individual Component.
   only specific Component we can test.

  -Integration testing means testing the integration of components.
   there are multiple components & they are talking to each other & we will develop a flow of an action in ojur react app
   that we will test.
  ex:- my application has 20 cards, i will search pizza in search box, we get only Three cards, this type of testing is integration testing.
   we will write code for this flow.

   -End To Testing means testing a React application as soon as user lands on the website to the user leaves the website.
   we will test all the flows.
   it requires some tools like cypress, etc.

   as a developer we concerned about only first 2 types of testing.
    in some companies testing is part of code.

Episode 13 Part 02
==========
->we will learn how to set application to test our app & what libraries are we going to use to do testing ?

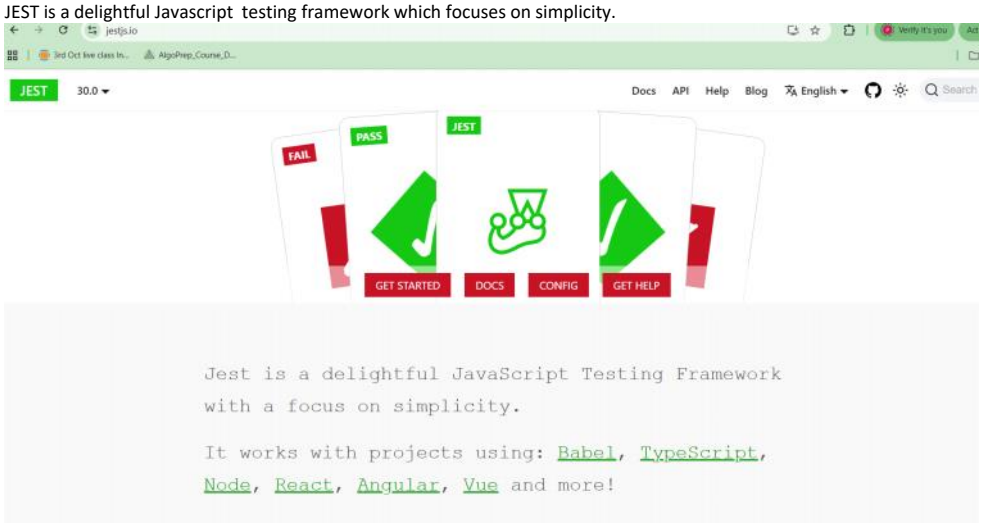
First library is React Testing library

It is most standard library used to write testcases in react.
It is very old testing library.
It is build on top of DOM testing library.
DOM testing library is base of all other Testing library/framework as well.

DOM testing library means without any frameworks.

->if we built the React app using create React app, then react testing library already added in it.
But we have built our application right from scratch using parcel which is bundler.
We are building our react app on top of the bundler.
So, with parcel, we have not have testing library inbuilt.
So, we have to integrate with our app.

->React Testing library uses something known as JEST.
JEST is a delightful Javascript testing framework which focuses on simplicity.

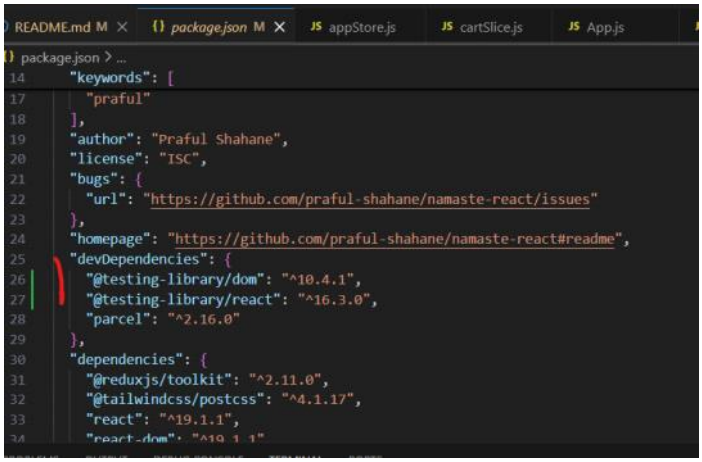


->DOM testing library or React Testing library uses JEST testing framework behind the scenes.
->So, React testing library needs JEST in our project also.
So, we need two things 1)React testing library 2)JEST in our project for setting testing environment.

Step 1) type command to set up react testing library.

```
PS C:\Users\HP\OneDrive\Desktop\namaste-react> npm install --save-dev @testing-library/react @testing-library/dom

added 15 packages, and audited 178 packages in 6s
added 15 packages, and audited 178 packages in 6s


77 packages are looking for funding
  run `npm fund` for details

3 moderate severity vulnerabilities

To address all issues, run:
```

Our testing library is added in package.json



Step 2) install ZEST dependcies
Also as our parcel uses babel so we need to add additional dependencies as well.
Also we need to config babel by create one file at root level.

Getting Started

Install Jest using your favorite package manager:

npm Yarn pnpm

```
npm install --save-dev jest
```

Using Babel

To use Babel, install required dependencies:

npm Yarn pnpm

```
npm install --save-dev babel-jest @babel/core @babel/preset-env
```

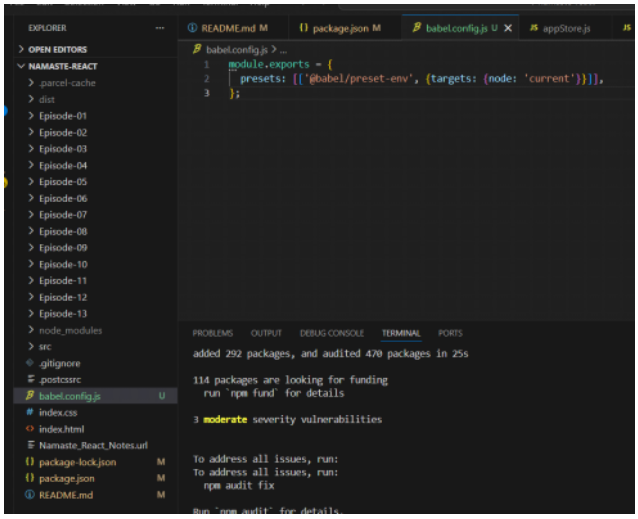
Configure Babel to target your current version of Node by creating a `babel.config.js` file in the root of your project:

babel.config.js

```
module.exports = {
  presets: [['@babel/preset-env', {targets: {node: 'current'}}]],
};
```

```
PS C:\Users\HP\OneDrive\Desktop\namaste-react> npm install --save-dev jest
```

```
PS C:\Users\HP\OneDrive\Desktop\namaste-react> npm install --save-dev babel-jest @babel/core @babel/preset-env
```



```
"devDependencies": {
  "@babel/core": "^7.28.5",
  "@babel/preset-env": "^7.28.5",
  "@testing-library/dom": "^10.4.1",
  "@testing-library/react": "^16.3.0",
  "babel-jest": "^30.2.0",
  "jest": "^30.2.0",
  "parcel": "^2.16.0"
},
```

-> as we see earlier episode, our parcel has automatically babel inside it.
Also Parcel uses babel behind the scenes.
Babel is transpiler & parcel uses babel.

When we use JEST using this below dependencies,

Using Babel

To use Babel, install required dependencies:

npm Yarn pnpm

```
npm install --save-dev babel-jest @babel/core @babel/preset-env
```

Configure Babel to target your current version of Node by creating a `babel.config.js` file in the root of your project:

babel.config.js

```
module.exports = {
  presets: [['@babel/preset-env', {targets: {node: 'current'}}]],
};
```

This dependencies will interfere with parcel's babel.
So, we are trying to configure babel according to us.
So, there will be conflict between parcel & babel.
Parcel says that I have my own internal babel configuration.
& we make babel config file babel.config.js

babel.config.js > ...

```
1  module.exports = {
2    presets: [['@babel/preset-env', {targets: {node: 'current'}}]],
3  };
```

This babel.config.js file will override the configuration of parcel's babel.

Website:: https://parceljs.org/languages/javascript/#usage-with-other-tools

Babel

Babel is a popular transpiler for JavaScript, with a large plugin ecosystem. Using Babel with Parcel works the same way as using it standalone or with other build tools. Create a Babel config file such as `.babelrc` and Parcel will pick it up automatically.

Parcel supports both project wide config files such as `babel.config.json`, as well as file relative configs such as `.babelrc`. See the Babel docs for details on configuration for more details.

> **Note**: JavaScript Babel configs (e.g. `babel.config.js`) should be avoided. These cause Parcel's caching to be less effective, which means all of your JS files will be recompiled each time you restart Parcel. To avoid this, use a JSON-based config format instead (e.g. `babel.config.json`).

Usage with other tools

While Parcel includes transpilation by default, you may still need to use Babel with other tools such as test runners like Jest, and linters like ESLint. If this is the case, you may not be able to completely remove your Babel config. You can make Parcel ignore your Babel config instead, which will have performance benefits and prevent the other issues described above.

To disable Babel transpilation in Parcel, override the default Parcel config for JavaScript to exclude `@parcel/transformer-babel`.

.parcelrc:

```
{
  "extends": "@parcel/config-default",
  "transformers": {
    "*.{js,mjs,jsx,cjs,ts,tsx)": [
      "@parcel/transformer-js",
      "@parcel/transformer-react-refresh-wrap"
    ]
  }
}
```

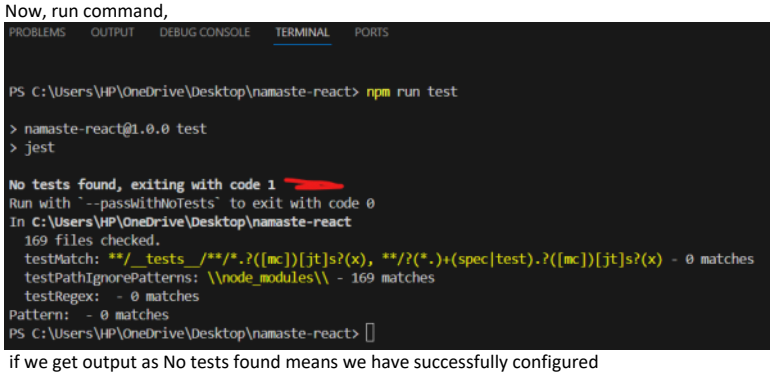
This will allow other tools to continue using your Babel config, but disable Babel transpilation in Parcel.

So, to disable parcel's babel transpilation we create .parcelrc file at root folder.
Then write above code into it.
Then our app will be use babel's configuration from babel.config.js file.
So, it will overcome conflict issue between parcel & babel.

Now, we have done our basic setup to write testcases.
How to run testcases ?

-> to run our app, we use **npm start** command.
->to run test cases in app, **npm run test** command we will use.

Now, run command,

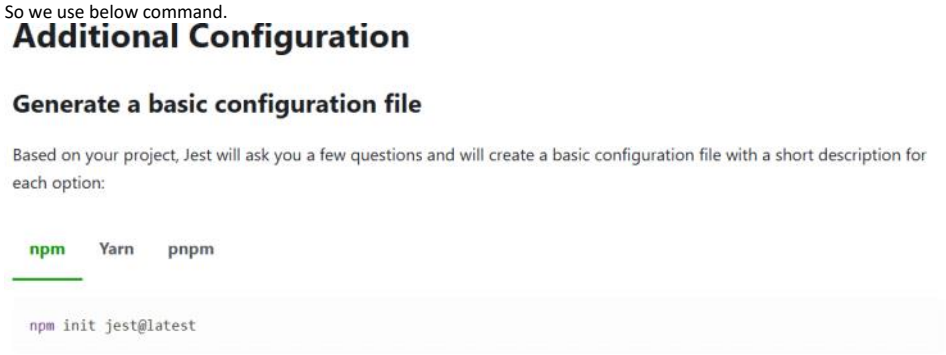


if we get output as No tests found means we have successfully configured
our React testing library, Jest, Babel & parcel.
->it does not throw any error, bcz we configure project for testing correctly.

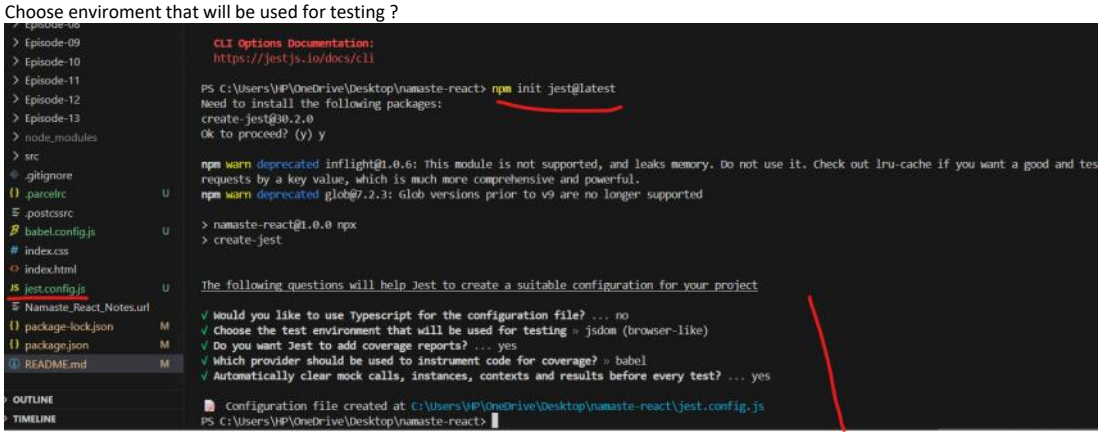
Episode 13 Part 03
==========
->Now we are ready to write testcases.
We make our React testing library, Jest, Babel & parcel are in sync.
->But before we write testcases, we need to make one configuration again which is
JEST configuration.
->Type command for configuration,

```sh
npx jest --init
```

npx means we are just executing jest package.
but above command is deprecated,
So we use below command.



Choose enviroment that will be used for testing ?



-> then new configuration file will be created at root folder i.e. **jest.config.js**

Q)Choose the test environment that will be used for testing
Select jsdom(browser-lik)

What is jsdom ?
->when we run testcases then there no server is running, there is no browser.
This testcases not run on browser.
They will need an enviroment at runtime so that testcases will be executed.

What is JSDOM used for?

Description: JavaScript Document Object Model (JSDOM) is a JavaScript-based headless browser that can be used to create a realistic testing environment.

We need to install one more library for jsdom install jsdom library.

Jest 28

If you're using Jest 28 or later, jest-environment-jsdom package now must be installed separately.

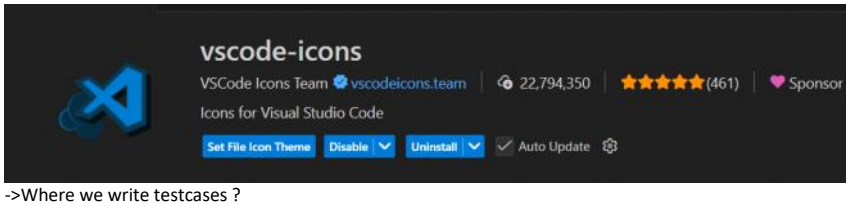
npm Yarn

```
npm install --save-dev jest-environment-jsdom
```

Our jest version in app is 30. so we install separately.

```
PS C:\Users\HP\OneDrive\Desktop\namaste-react> npm install --save-dev jest-environment-jsdom

added 41 packages, and audited 607 packages in 10s

126 packages are looking for funding
  run `npm fund` for details

3 moderate severity vulnerabilities

To address all issues, run:
  npm audit fix

Run `npm audit` for details.
PS C:\Users\HP\OneDrive\Desktop\namaste-react>
```

->Now our testing setup is completed.
-> let's write some javascript code to test without react.
Install VS code icons extension in vs code for see icon for file.



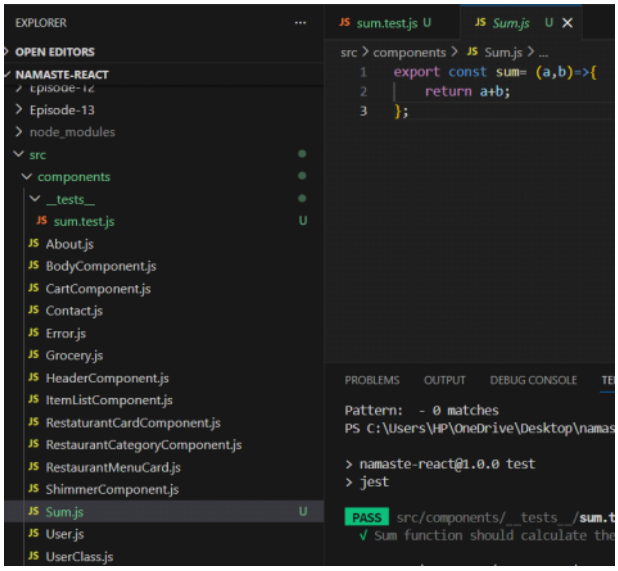
->Where we write testcases ?
As soon as we type command for test, we see that it is searching for folder **__tests__**

```
PS C:\Users\HP\OneDrive\Desktop\namaste-react> npm run test

> namaste-react@1.0.0 test
> jest

No tests found, exiting with code 1
Run with `--passWithNoTests` to exit with code 0
In C:\Users\HP\OneDrive\Desktop\namaste-react
  171 files checked.
  testMatch: **/__tests__/**/*.?([mc])[jt]s?(x), **/?(*.)+(spec|test).?([mc])[jt]s?(x) - 0 matches
  testPathIgnorePatterns: \\node_modules\\ - 171 matches
  testRegex:  - 0 matches
Pattern:  - 0 matches
```

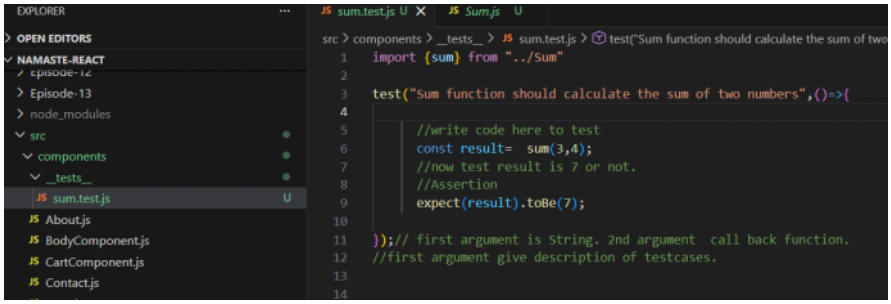
Or
we can create file at any location to write testcase with .test.js or .test.ts or spec.ts or spec.js
Ex:- Header.test.js **or** Header.test.ts **or** Header.spec.js **or** Header.spec.ts all are testing files.

_ _ (2 underscore)=>dunder tests.

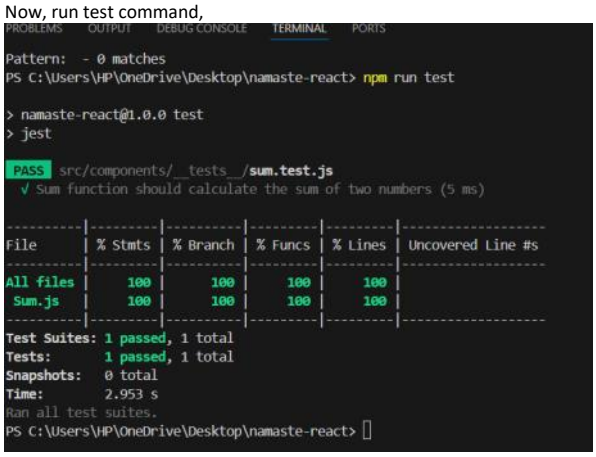
1)Lets create one dummy testcase like check to calculate sum.
Sum.js



Sum.test.js

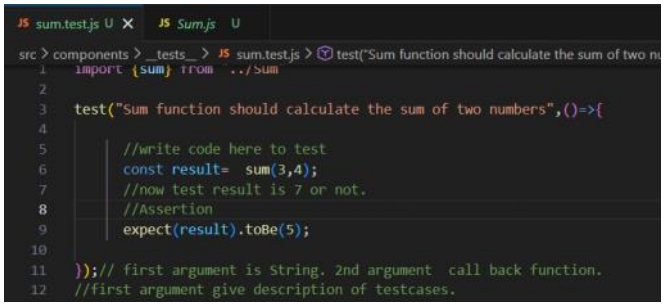


Now, run test command,



->If I make expected result to 5 instead of 7. our testcase will be failed.

Sum.test.js



O/P => when our expected result not correct it throw error =>

->If we comment, Assertion still our run testcase but it is good practice to have Assertion.

Episode 13 Part 04

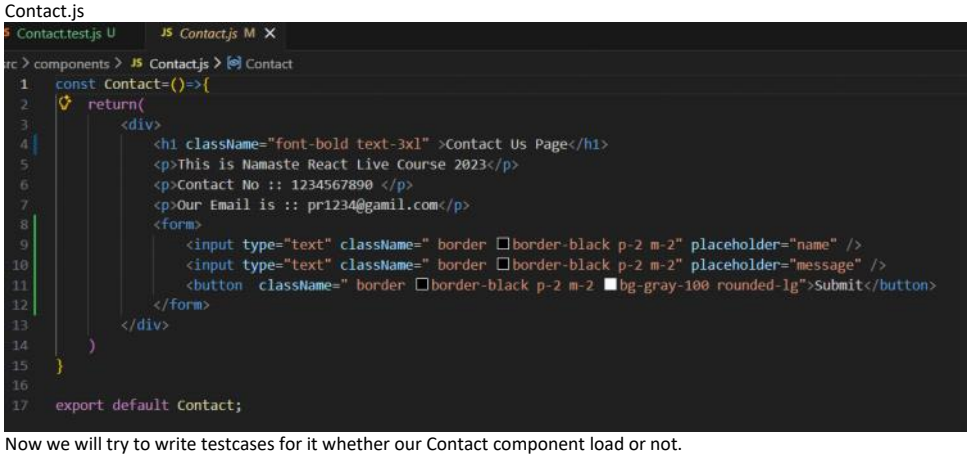
==========

->Now we are ready to write testcases for React.

Now first we will write unit testcases.

First we will test our Contact Us Component page is loaded or not.

Contact.js

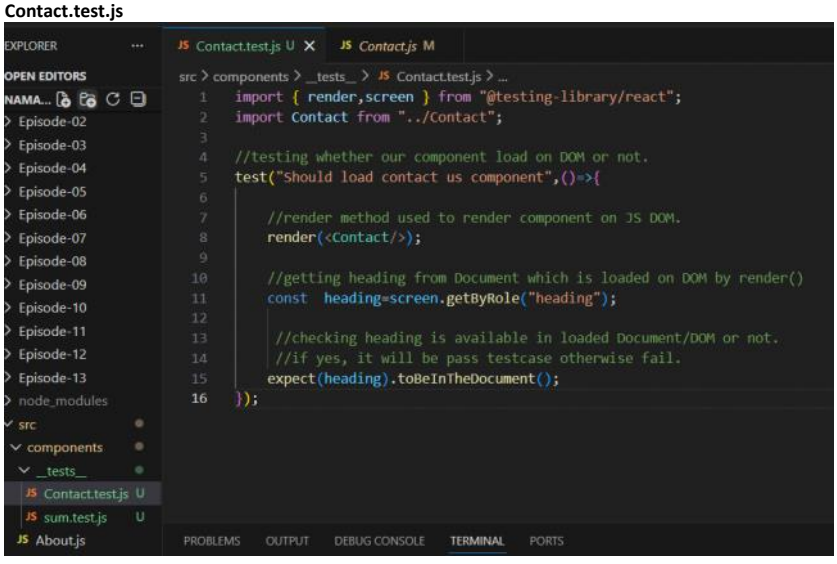


Now we will try to write testcases for it whether our Contact component load or not.

Note::

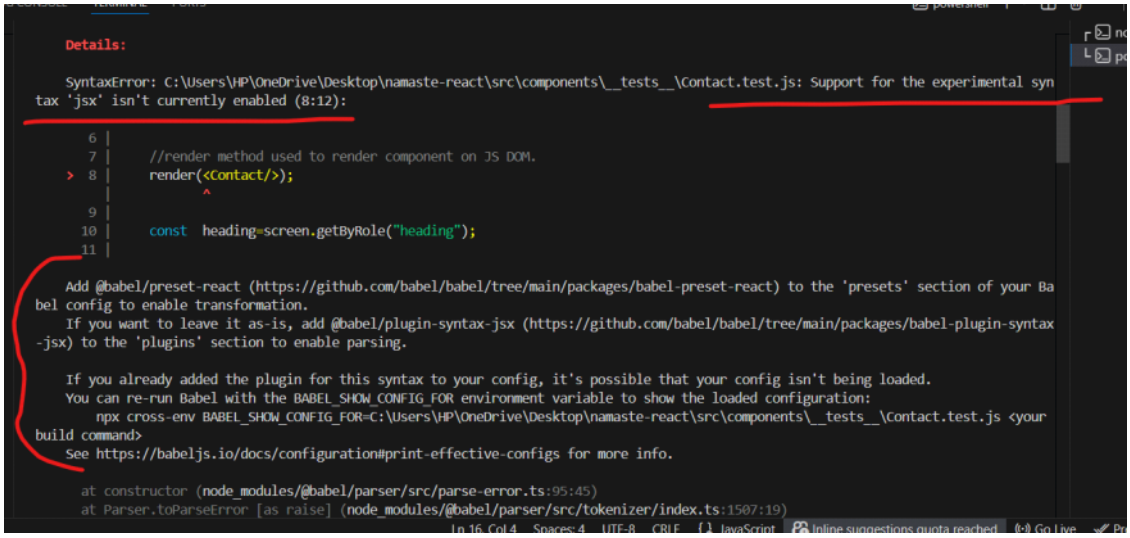
->Whenever we are testing UI Component inside React, we need to render that component onto the JS DOM first of all.

Contact.test.js



Once we run testcase:- we get error as, saying support for JSX is not yet enabled for testcases.

We need to add one library.



Once we install the library,

```
PS C:\Users\HP\OneDrive\Desktop\namaste-react> npm i -d @babel/preset-react
```

or

```
npm install --save-dev @babel/preset-react
```

We need to include, babe/present inside babel.config.js

What is presents ?
=>Babel converts code from one form to another.
Babel/preset-react basically helping our testing library to convert JSX code to HTML.
So that it can read properly.

Now, if we try to run testcases,
We get some other error,

We get error bcz this function is not present in current library, for this we need to install one more library.
Install below library.

jest-dom

jest-dom is a companion library for Testing Library that provides custom DOM element matchers for Jest

npm Yarn

```
npm install --save-dev @testing-library/jest-dom
```

```
PS C:\Users\HP\OneDrive\Desktop\namaste-react> npm install --save-dev @testing-library/jest-dom
```

```
components > __tests__ > JS Contact.test.js > test("Should load contact us component") callback
import { render,screen } from "@testing-library/react";
import Contact from "../Contact";
import "@testing-library/jest-dom";

//testing whether our component load on DOM or not.
test("Should load contact us component",()=>{

    //render method used to render component on JS DOM.
    render(<Contact/>);

    //getting heading from Document using screen  which is loaded on DOM by render()
    const  heading=screen.getByRole("heading");

     //checking heading is available in loaded Document/DOM or not.
     //if yes, it will be pass testcase otherwise fail.
    expect(heading).toBeInTheDocument();


});
```

Now, run our test.

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

Run `npm audit` for details.
PS C:\Users\HP\OneDrive\Desktop\namaste-react> npm run test

> namaste-react@1.0.0 test
> jest

 PASS  src/components/__tests__/sum.test.js
 PASS  src/components/__tests__/Contact.test.js
------------|---------|----------|---------|---------|-------------------
File        | % Stmts | % Branch | % Funcs | % Lines | Uncovered Line #s
------------|---------|----------|---------|---------|-------------------
All files   |     100 |      100 |     100 |     100 |
 Contact.js |     100 |      100 |     100 |     100 |
 Sum.js     |     100 |      100 |     100 |     100 |
------------|---------|----------|---------|---------|-------------------

Test Suites: 2 passed, 2 total
Tests:       2 passed, 2 total
Snapshots:   0 total
Time:        3.277 s
Ran all test suites.
```

Explanation:-

```
components > __tests__ > JS Contact.test.js > test("Should load contact us component") callback
import { render,screen } from "@testing-library/react";
import Contact from "../Contact";
import "@testing-library/jest-dom";

//testing whether our component load on DOM or not.
test("Should load contact us component",()=>{

    //render method used to render component on JS DOM.
    render(<Contact/>);

    //getting heading from Document using screen  which is loaded on DOM by render()
    const  heading=screen.getByRole("heading");

     //checking heading is available in loaded Document/DOM or not.
     //if yes, it will be pass testcase otherwise fail.
    expect(heading).toBeInTheDocument();


});
```

1)we have render our Contact component using render() which is given by testing library.
2)we can access the render DOM, using screen which is given by testing library.

```
const  button=screen.getByRole("button");
```

There are different types of roles is there in HTML.
These roles are defined by JEST & testing library.
Contact.test.js

Contact.test.js U | Contact.js M | README.md M | babel.config.js U | package.json M

src > components > __tests__ > Contact.test.js > ...

```
import { render,screen } from "@testing-library/react";
import Contact from "../Contact";
import "@testing-library/jest-dom";

//testing whether our component load on DOM or not.
test("Should load contact us component",()=>{

    //render method used to render component on JS DOM.
    render(<Contact/>);

    //getting heading from Document using screen  which is loaded on DOM by render()
    const  heading=screen.getByRole("heading");

     //checking heading is available in loaded Document/DOM or not.
     //if yes, it will be pass testcase otherwise fail.
    expect(heading).toBeInTheDocument();
});

test("Should load button inside contact us component",()=>{

    //render method used to render component on JS DOM.
    render(<Contact/>);

    //getting button from Document using screen  which is loaded on DOM by render()
    // const  button=screen.getByRole("button");  or
    const button = screen.getByText("Submit");

     //checking button is available in loaded Document/DOM or not.
     //if yes, it will be pass testcase otherwise fail.
    expect(button).toBeInTheDocument();
});
```

```
test("Should load input name inside contact us component",()=>{

    //render method used to render component on JS DOM.
    render(<Contact/>);

    //getting inputName from Document using screen  which is loaded on DOM by render()
    const inputName = screen.getByPlaceholderText("name");

     //checking button is available in loaded Document/DOM or not.
     //if yes, it will be pass testcase otherwise fail.
    expect(inputName).toBeInTheDocument();
});

test("Should load 2 input boxes inside contact us component",()=>{

    //render method used to render component on JS DOM.
    render(<Contact/>);

    //getting inputName from Document using screen  which is loaded on DOM by render()
    //Querying
    const inputBoxes = screen.getAllByRole("textbox"); //role of input tag is textbox
    //it return JSX Element from screen

    console.log(inputBoxes.length);

    expect(inputBoxes.length).not.toBe(3);
});
```

Episode 13 Part 04

==========

->If our component have 20 testcases, we can group them in diff group, using describe();
It has no impact on execution of test cases. We are using for grouping only.
describe("write name of group",()=>{});
Ex:-

```
describe("Contact us Page test case",()=>{
    //testing whether our component load on DOM or not.
test("Should load contact us component",()=>{

    //render method used to render component on JS DOM.
    render(<Contact/>);

    //getting heading from Document using screen  which is loaded on DOM by render()
    const  heading=screen.getByRole("heading");

     //checking heading is available in loaded Document/DOM or not.
     //if yes, it will be pass testcase otherwise fail.
    expect(heading).toBeInTheDocument();
});

test("Should load button inside contact us component",()=>{

    //render method used to render component on JS DOM.
    render(<Contact/>);

    //getting button from Document using screen  which is loaded on DOM by render()
    // const  button=screen.getByRole("button");  or
    const button = screen.getByText("Submit");

     //checking button is available in loaded Document/DOM or not.
     //if yes, it will be pass testcase otherwise fail.
    expect(button).toBeInTheDocument();
});
});
```

->we can write name of testcase is **test**. We can write this name as **it** also.

Both are same it is alias of test.

```
38
39 > it("Should load input name inside contact us component",()=>{ ···
50   });
51
52 > test("Should load 2 input boxes inside contact us component",()=>{ ···
65   });
```

This is unit testing we can do in isolation for component.

Episode 13 Part 05

==========

->We don't need to push our coverage files.
So add coverage file in gitignore.

Episode 13 Part 06

==========

->We are doing unit testing currently.
Now we will test our HeaderComponent successfully loaded or not.

```
components > __tests__ > JS HeaderComponent.test.js > it("Should load Header Comp
import { render , screen} from "@testing-library/react";
import HeaderComponent from "../HeaderComponent";

it("Should load Header Component with login button",()=>{

    render(<HeaderComponent/>);

});
```

->if we try to run above testcase, we get error like,
->Our HeaderComponent contain react-router library,
So our testcase will not run, we get error as,'
referenceerror textencoder is not defined.

To fixed this error, we install following dependency & setup environment in jest.config.js file.
As below instructions.

referenceerror textencoder is not defined in react testing library

Solution 3: Use `jest-fixed-jsdom`

If the standard solutions don't work, particularly when using MSW (Mock Service Worker), you can use the `jest-fixed-jsdom` environment. This environment avoids potential compatibility issues that can arise with `jest-environment-jsdom`.

1. Install the package as a development dependency.

```sh
npm install -D jest-fixed-jsdom
```

2. Update your Jest configuration (`jest.config.js` or `package.json`) to use the new test environment.

```javascript
// jest.config.js
module.exports = {
  testEnvironment: 'jest-fixed-jsdom',
};
```

->Now, if we run our testcase, we get diff error like,

```
    at log (src/components/HeaderComponent.js:19:14)

● Should load Header Component with login button

  could not find react-redux context value; please ensure the component is wrapped in a <Provider>

   26 |       //Selector is hook inside the react,
   27 |       //Subscribe to the redux store using useSelector hook.
  > 28 |       const cartItems = useSelector((store)=> store.cart.items);  // we can access the redux sto
      |                                                                    ^
   29 |       console.log(cartItems); // {items: Array(2)} items: ['burger','pizza']
   30 |
```

we get above error because, our JSDOM does not understand redux code.
JSDOM only understand JSX, javascript code, React code but it does not understand redux code.
So,cwe need to provide store to them i.e. HeaderComponent. i.e. we need to wrap our HeaderComponent inside the Provider which Is given by redux.

```
components > __tests__ > HeaderComponent.test.js > it("Should load Header C
import { render , screen} from "@testing-library/react";
import HeaderComponent from "../HeaderComponent";
import { Provider } from "react-redux";
import appStore from "../../utils/appStore"


it("Should load Header Component with login button",()=>{

    render(
    <Provider store={appStore}>
    <HeaderComponent/>
    </Provider>


);
```

Now, if we run testcases again, we get error as below,

```
● Should load Header Component with login button

  TypeError: Cannot destructure property 'basename' of 'React.useContext(...)' as it is null.

    8 | it("Should load Header Component with login button",()=>{
    9 |
  > 10 |     render(
      |          ^
   11 |         <Provider store={appStore}>
   12 |          <HeaderComponent/>
   13 |         </Provider>
```

Bcz we need to wrap our code inside the BrowserRouter. Why we need BrowserRouter ?
Bcz we use Link which is comes from react router. So, we need to wrap our HeaderComponent inside the BrowserRouter.
Like as below,

```
components > __tests__ > HeaderComponent.test.js > it("Should load Header
import { render , screen} from "@testing-library/react";
import HeaderComponent from "../HeaderComponent";
import { Provider } from "react-redux";
import appStore from "../../utils/appStore"
import { BrowserRouter } from "react-router-dom";


it("Should load Header Component with login button",()=>{

    render(
        <BrowserRouter>
        <Provider store={appStore}>
         <HeaderComponent/>
        </Provider>
    </BrowserRouter>
);
});
```

Now, if we run our testcases, our testcases will be pass.

```
----------------------|---------|----------|---------|

Test Suites: 3 passed, 3 total
Tests:       6 passed, 6 total
Snapshots:   0 total
Time:        3.81 s
Ran all test suites.
```

->We can write above code as,

```
components > __tests__ > HeaderComponent.test.js > it("Should load Header Compo
import { Provider } from "react-redux";
import appStore from "../../utils/appStore"
import { BrowserRouter } from "react-router-dom";


it("Should load Header Component with login button",()=>{

    render(
        <Provider store={appStore}>
        <BrowserRouter>
        <HeaderComponent/>
        </BrowserRouter>
        </Provider>
);
});
```

->our testcase will be pass.

But here we not add assertion to check it.
Now we adding assertion.
Now we have completed our writing testcases for HeaderComponent.js
HeaderComponent.test.js

```
src > components > __tests__ > JS HeaderComponent.test.js > it("Should chnaged Login button to LogOut on click") callback
1   import { fireEvent, render , screen} from "@testing-library/react";
2   import HeaderComponent from "../HeaderComponent";
3   import { Provider } from "react-redux";
4   import appStore from "../../utils/appStore"
5   import { BrowserRouter } from "react-router-dom";
6   import "@testing-library/jest-dom"
7
8
9   it("Should render Header Component with login button",()=>{ ···
30  });
31
32  it("Should render Header Component with Cart Item 0 ",()=>{ ···
46  });
47
48  it("Should render Header Component with Cart any Item",()=>{ ···
63  });
64
65  it("Should chnaged Login button to LogOut on click",()=>{ ···
86  });
```

O/P=>

```
Test Suites: 3 passed, 3 total
Tests:       9 passed, 9 total
---------------------|---------|----------|---------|

Test Suites: 3 passed, 3 total
Tests:       9 passed, 9 total
Snapshots:   0 total
Time:        4.278 s
Ran all test suites.
```

Episode 13 Part 07

==========

->Now we can write testcases for RestaurantCard Component.

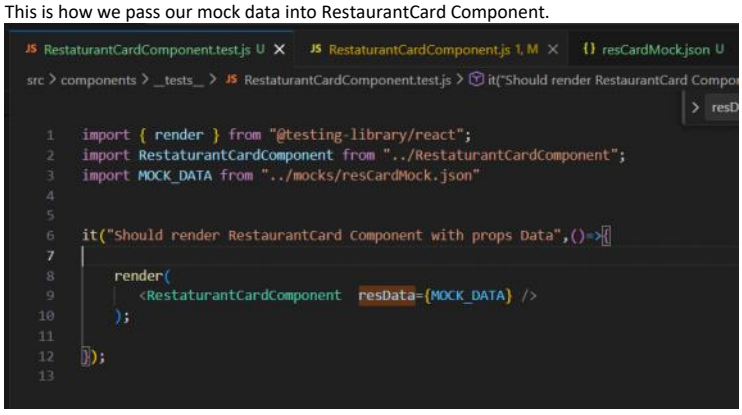
Unique things about RestaurantCard Component is they receives props.

For props, we use mocked data.

So, get mock data from our live api & store that JSON into one file under mock folder.

->

This is how we pass our mock data into RestaurantCard Component.



->

```
src > components > __tests__ > RestaturantCardComponent.test.js > it("Should render RestaurantCard Component with props
```

```
import { render ,screen} from "@testing-library/react";
import RestaturantCardComponent from "../RestaturantCardComponent";
import MOCK_DATA from "../mocks/resCardMock.json";
import "@testing-library/jest-dom";

it("Should render RestaurantCard Component with props Data",()=>{

    render(
      <RestaturantCardComponent  resData={MOCK_DATA} />
    );
    //get name from JSDOM document
   const name= screen.getByText("Olio - The Wood Fired Pizzeria");

    expect(name).toBeInTheDocument();

});
```

```
PROBLEMS 1  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

 Sum.js                    |     100 |     100 |     100 |     100 |
 utils                     |   59.09 |     100 |   28.57 |   59.09 |
  UserContext.js           |     100 |     100 |     100 |     100 |
  appStore.js              |     100 |     100 |     100 |     100 |
  cartSlice.js             |   28.57 |     100 |       0 |   28.57 | 14-43
  constants.js             |     100 |     100 |     100 |     100 |
  useOnlineStatus.js       |      60 |     100 |      50 |      60 | 15-16,19-20
---------------------------|---------|---------|---------|---------|-------------------

Test Suites: 4 passed, 4 total
Tests:       10 passed, 10 total
Snapshots:   0 total
Time:        4.431 s
Ran all test suites.
```

->Now test our RestaurantCardComponent with DiscountLable.

```
src > components > RestaurantCardComponent.js > withDiscountRestaurantMenuCard
```

```
import { CDN_URL } from "../utils/constants";
import { useContext } from "react";
import UserContext from "../utils/UserContext";

const RestaturantCardComponent =(props)=>{ ···
  };


  //Higher order component

//input - RestaurantMenuCard
//output - RestaurantMenuCardDiscountOffer

export const withDiscountRestaurantMenuCard =(RestaurantMenuCard)=>{

        //props are passed from BodyComponent to RestaurantMenuCardWithDiscount
        //then from RestaurantMenuCardWithDiscount to RestaurantMenuCard
        //...props is used to pass all the props at once.
           return (props)=>{
                  return (
                   <div className="border-[1px] ">
                     <label  className=" absolute text-white bg-black" >Discount</label>
                     <RestaurantMenuCard {...props}/>
                   </div>
                  );
           };
};

  export default RestaturantCardComponent;
```

->RestaurantCardComponent.test.js

```
src > components > __tests__ > RestaurantCardComponent.test.js > ...
```

```
import { render ,screen} from "@testing-library/react";
import RestaturantCardComponent, { withDiscountRestaurantMenuCard } from "../RestaturantCardComponent";
import MOCK_DATA from "../mocks/resCardMock.json";
import "@testing-library/jest-dom";

```

```
it("Should render RestaurantCard component with Discount label",()=>{
      const RestaturantCardComponentWithDiscount= withDiscountRestaurantMenuCard(RestaturantCardComponent);

    render(
      <RestaturantCardComponentWithDiscount   resData={MOCK_DATA} />
    )

   //get name from JSDOM document
  const name= screen.getByText("Olio - The Wood Fired Pizzeria");
      expect(name).toBeInTheDocument();
});
```

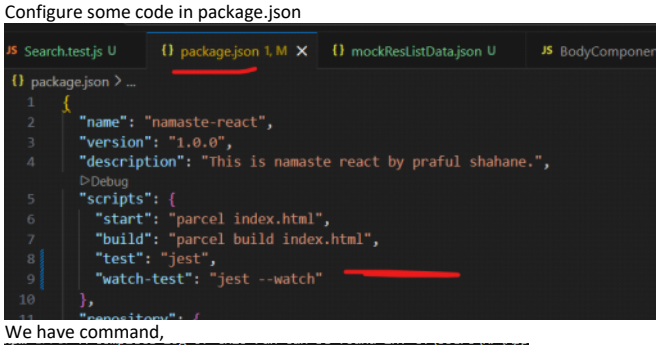
o/p=>

```
Test Suites: 4 passed, 4 total
Tests:       11 passed, 11 total
Snapshots:   0 total
Time:        4.642 s
Ran all test suites.
```

->Now, we will write integration testing for one feature it involve a lot of components.

Note:-
->To make our test run automatically, once we did anychanges,

Configure some code in package.json



We have command,

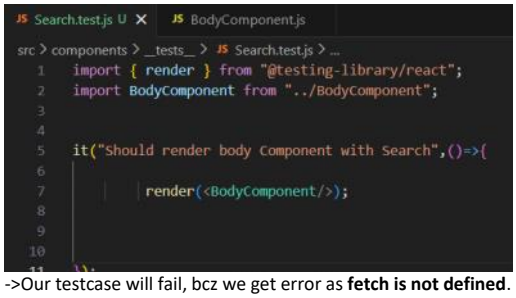
```
PS C:\Users\HP\OneDrive\Desktop\namaste-react> npm run watch-test
```

Now, our testcases will run automatically if we make any changes.

Episode 13 Part 08

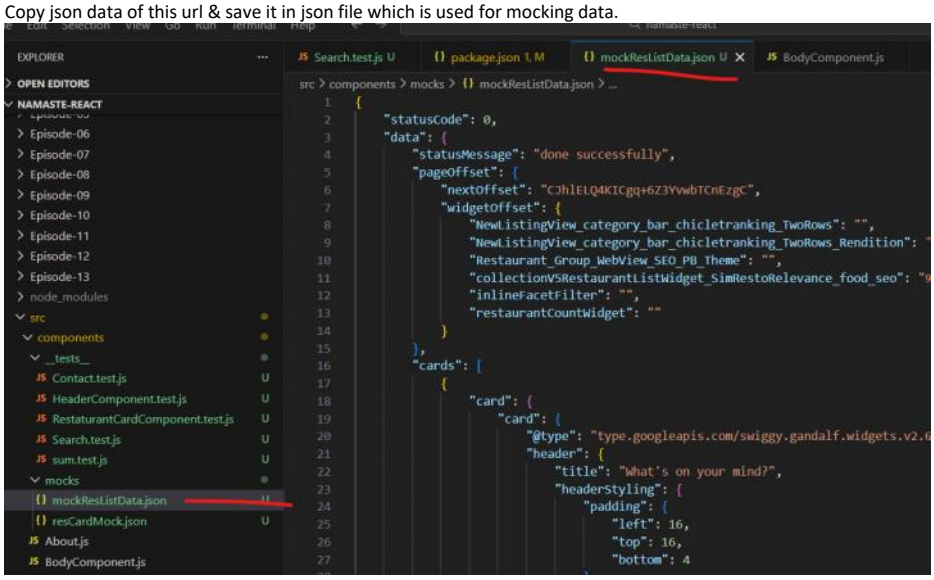
==========

->Now we will write integration testing.
As soon our body loads, it shows a list of cards.
If I write burger in search input & click on search, we get burger restaurant card.
We will test this feature.



->Our testcase will fail, bcz we get error as **fetch is not defined**.
Bcz our Browser understand **fetch**.
But our JSDOM which is browser like it does not have all the superpower of browsers.
Fetch is given us by browser not by javascript.
So, when we render our BodyComponent, it is render inside the JSDOM which is browser like but not browser.
So, we will write mock function for fetch.

https://www.swiggy.com/dapi/restaurants/list/v5?lat=21.14630&lng=79.08490&is-seo-homepage-enabled=true&page_type=DESKTOP_WEB_LISTING
Copy json data of this url & save it in json file which is used for mocking data.



->as soon as we run testcases, we get warning as,

```
    at log (src/components/BodyComponent.js:46:15)

  console.error
    An update to BodyComponent inside a test was not wrapped in act(...).

    When testing, code that causes React state updates should be wrapped into act(...):

    act(() => {
      /* fire events that update state */
    });
    /* assert on the output */

    This ensures that you're testing the behavior the user would see in the browser. Learn more at https://react.dev/link/wrap-tests-with-act

    50 |       //Optional Chaining Operator (?.) is used to avoid runtime errors if any property is undefined or null.
    51 |
  > 52 |       setListOfRestaurant(jsonData?.data?.cards[1]?.card?.card?.gridElements?.infoWithStyle?.restaurants);
       |       ^
    53 |
    54 |
    55 |       //when we fetch the data from API, we have to set the filtered restaurant list also.

    at node_modules/react-dom/cjs/react-dom-client.development.js:16023:19
```

->Original fetch call in BodyComponent.

```
  //Using async await way of writing function
const fetchData =async ()=>{
    const data = await fetch(
     "https://www.swiggy.com/dapi/restaurants/list/v5?lat=21.14630&lng=79.08490&is-seo-homepage-enabled=true&page_type=DESKTOP
    );
     //get JsON from the response object
    const jsonData= await data.json();
    console.log(jsonData);

    console.log(jsonData.data.cards[4].card.card.gridElements.infoWithStyle.restaurants);
```

->

We are trying to make fetch function exactly like our original fetch function.
original fetch function return a promise & that promise return us a json
& we convert into json & then it return a promise once again.
when we resolve that promise then we actually get the data.

```
global.fetch=  jest.fn(()=>{

    //fetch function return a promise
    return Promise.resolve({
        json: ()=>{
            return Promise.resolve(MOCK_DATA);
        }
    })

});
```

->Whenever we are doing state update activity or asyn operation, we have to wrap our function inside the act().
act function return a promise, so we need to add await for act().
we need to add async for it()'s callback function.
act takes callback function this again async function.
then async function render our component.

Note:- Earlier act() comes from react-dom/test-utils
This is deprecated, so now it is given by @testing-library/react.

import { render ,act } from "@testing-library/react";
// import {act } from "react-dom/test-utils";

Search.test.js

```
import { render ,act } from "@testing-library/react";
// import {act } from "react-dom/test-utils";
import BodyComponent from "../BodyComponent";
import { jsx } from "react/jsx-runtime";
import MOCK_DATA from "../mocks/mockResListData.json";
import { BrowserRouter  } from "react-router-dom";

/*
We are trying to make fetch function exactly like our orginal fetch function.
original fetch function return a promise & that promise return us a json
& we convert into json & then it return a promise once again.
when we resolve that promise then we actually get the data.

*/
global.fetch=  jest.fn(()=>{

    //fetch function return a promise
    return Promise.resolve({
        json: ()=>{
            return Promise.resolve(MOCK_DATA);
        }
    })
});
```

```
it("Should render body Component with Search",async ()=>{

    /* act function return a promise, so we need to add await for act().
    we need to add async for it()'s callback function.
    act takes callback function this again async function.
    then async function render our component
    */
    await act( async ()=> render(
        <BrowserRouter>
            <BodyComponent/>
            </BrowserRouter>
));

});
```

->now if we run testcase, we did not get any error or warning.
Now we add assertion in testcases.

```
Search.test.js U   package.json 1, M   mockResListData.json U   BodyComponent.js
src > components > __tests__ > Search.test.js > ...
  it("Should render body Component with Search",async ()=>{
    then async function render our component
   */
    await act( async ()=> render(
        <BrowserRouter>
            <BodyComponent/>
            </BrowserRouter>
  ));

  //search button is there or not.
  const searchBtn=screen.getByRole("button", {name: "Search"});

  console.log(searchBtn);

  expect(searchBtn).toBeInTheDocument();


  });
```

```
PROBLEMS 2   OUTPUT   DEBUG CONSOLE   TERMINAL   PORTS

 RestaturantCardComponent.js |     100 |      100 |     100 |     100 |
 Sum.js                      |     100 |      100 |     100 |     100 |
utils                        |     100 |      100 |     100 |     100 |
 appStore.js                 |     100 |      100 |     100 |     100 |
-----------------------------|---------|----------|---------|---------|-------------------

Test Suites: 5 passed, 5 total
Tests:       12 passed, 12 total
Snapshots:   0 total
Time:        2.906 s
Ran all test suites related to changed files.
```

->Now our testcase is passed by checking Search button is there in BodyComponent.
->Now, we need to pass input to update our input box.
We have onChange event for this.

```
    <div  className="flex m-4 p-4 ">
     <div  className="search m-4 p-4">
        <input  type="text"  data-testid="searchInput" className="border border-solid border-black"  value={searchText
          onChange={ (e)=>{
              setSearchText(e.target.value);
          }} />
        <button  className="px-4 py-2 bg-green-200 mx-4 rounded-3xl"
        onClick={()=>{ //Filter the restaurant cards & update the UI.  //we need searchText here ignore case.
const filterRestaurantList=listOfRestaurants.filter(res=>
 res.info.name.toLowerCase().includes(searchText.toLowerCase())
);
//update the restaurant list by filtered restaurant list.
```

```
return(
  <div  data-testid="resCard" className="m-4 p-4 w-[250px] bg-gray-150 hover:bg-gray-400">
  <img  className="rounded-lg"
  alt="res-logo"
  src={
  CDN_URL
```

Search.test.js

```
src > components > __tests__ > Search.test.js > it("Should Search resList for pizza input") callback
1   import { render ,act,screen, fireEvent } from "@testing-library/react";
2   // import {act } from "react-dom/test-utils"; //Deprecated
3   import BodyComponent from "../BodyComponent";
4   import { jsx } from "react/jsx-runtime";
5   import MOCK_DATA from "../mocks/mockResListData.json";
6   import { BrowserRouter, ScrollRestoration  } from "react-router-dom";
7   import "@testing-library/jest-dom"
8   /*
9   We are trying to make fetch function exactly like our orginal fetch function.
10  original fetch function return a promise & that promise return us a json
11  & we convert into json & then it return a promise once again.
12  when we resolve that promise then we actually get the data.
13
14  */
15  global.fetch=  jest.fn(()=>{
16
17      //fetch function return a promise
18      return Promise.resolve({
19          json: ()=>{
20              return Promise.resolve(MOCK_DATA);
21          }
22      })
23  });
24
```

```
it("Should Search resList for pizza input",async ()=>{

    /* act function return a promise, so we need to add await for act().
    we need to add async for it()'s callback function.
    act takes callback function this again async function.
    then async function render our component
    */
    await act( async ()=> render(
        <BrowserRouter>
            <BodyComponent/>
            </BrowserRouter>
));

    //Before searching, my rescard must be 20.
    const cardsBeforeSearch= screen.getAllByTestId("resCard");
    expect(cardsBeforeSearch.length).toBe(20);

//search button is there or not.
const searchBtn=screen.getByRole("button", {name: "Search"});

 const searchInput = screen.getByTestId("searchInput");

 //Typing inpput into input buttton & passing  value key inside the target object.
 //so that our code can access it as e.target.value
  fireEvent.change(searchInput,{target:
                                {value:"Pizza"}
                                });
//click on search buton.
  fireEvent.click(searchBtn);

  //Now i expect 2 card to be loaded on JSDOM.
/* we have list of cards, to identify each card by commanality,
we finding using testId. add one resCard as id in RestaurantCardComponent.js
*/
  const cardsAfterSearch= screen.getAllByTestId("resCard");
expect(cardsAfterSearch.length).toBe(2);
});
```

O/P of testcases=>

```
File    | % Stmts | % Branch | % Funcs | % Lines | Uncovered Line #s
--------|---------|----------|---------|---------|-------------------
...les  |   91.07 |       70 |   84.21 |   92.59 |
...ts   |    90.9 |       70 |   84.21 |   92.45 |
 ...s   |   83.33 |    83.33 |      70 |    86.2 | 63,96-104
 ...s   |     100 |      100 |     100 |     100 |
 ...s   |     100 |       50 |     100 |     100 | 42-57
 ...s   |     100 |      100 |     100 |     100 |
 ...s   |     100 |      100 |     100 |     100 |
utils   |     100 |      100 |     100 |     100 |
 ...s   |     100 |      100 |     100 |     100 |
--------|---------|----------|---------|---------|-------------------

Test Suites: 5 passed, 5 total
Tests:       12 passed, 12 total
Snapshots:   0 total
Time:        3.079 s
Ran all test suites related to changed files.
```

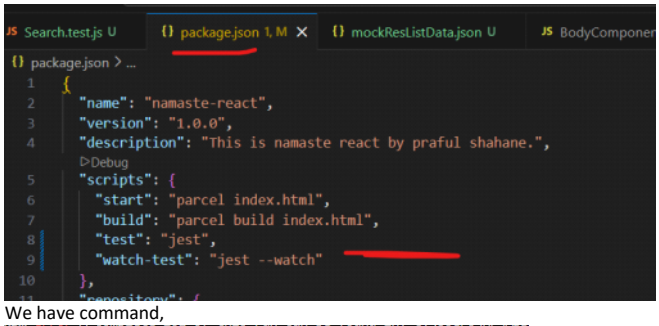
->

2)Now we test top rated restaurant feature.

```
it("Should filter top rated restaurant",async ()=>{

    /* act function return a promise, so we need to add await for act().
    we need to add async for it()'s callback function.
    act takes callback function this again async function.
    then async function render our component
    */
    await act( async ()=> render(
        <BrowserRouter>
            <BodyComponent/>
            </BrowserRouter>
));

    //Before searching, my rescard must be 20.
    const cardsBeforeFilter= screen.getAllByTestId("resCard");
    expect(cardsBeforeFilter.length).toBe(20);

  //find top rated restaurant button.
const topRatedResBtn = screen.getByRole("button",{name:"Top Rated Restaurants"});

//click on that topRatedResBtn button
fireEvent.click(topRatedResBtn);

  const cardsAfterFilter= screen.getAllByTestId("resCard");
  console.log(cardsAfterFilter.length);


expect(cardsAfterFilter.length).toBe(3);

});
```

Note:-

->To make our test run automatically, once we did anychanges,
Configure some code in package.json



We have command,

```
PS C:\Users\HP\OneDrive\Desktop\namaste-react> npm run watch-test
```

Now, our testcases will run automatically if we make any changes.

Episode 13 Part 09

==========

->Now, In Describe () we have a lot of testcases, it can contain below methods also.

1)Before all:-
Before starting all Testcases inside the describe()
Before all code will be executed.

2)Before Each:-
Before starting Each Testcases individually inside the describe()
Before Each code will be executed.

3)After all:-
After ending all Testcases inside the describe()
After all code will be executed.

4)After Each:-
After ending Each Testcases individually inside the describe()
After Each code will be executed.

```
describe("Contact us Page test case",()=>{

    //Before starting all Testcases inside the describe()
    //Before all code will be executed.
  beforeAll(()=>{
    console.log("Before All");

  })
   //Before starting Each Testcases individually inside the describe()
    //Before Each code will be executed.
  beforeEach(()=>{
        console.log("Before Each");
  })

    //After ending all Testcases inside the describe()
    //After all code will be executed.
  afterAll(()=>{
    console.log("after All");
  })

  //After ending Each Testcases individually inside the describe()
    //After Each code will be executed.
  afterEach(()=>{
    console.log("after Each");
  })
```

Episode 13 Part 10

==========

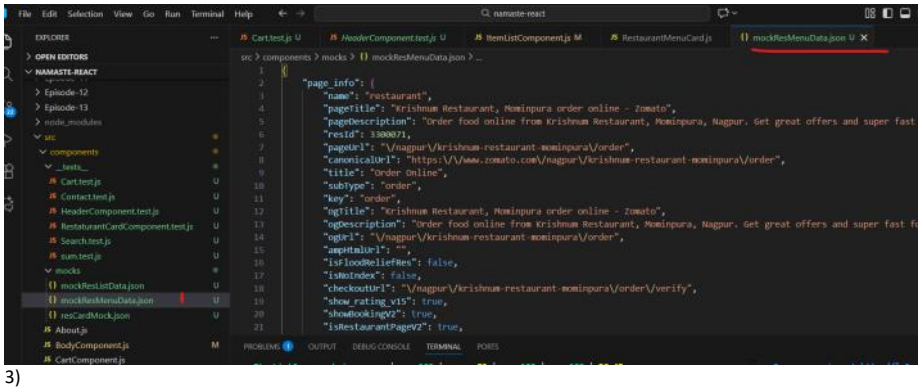
->Now, we will test ***add to Cart*** feature.

->if we click on add button of menu card, it updates our Cart on Header & our Cart Page(CartComponent) updates.
This flow we need to test.

Add one id in ItemListComponent.js

```
Cart.test.js U   HeaderComponent.test.js U   ItemListComponent.js M   RestaurantMenuCard.js   mockResMenuData.jso
components > ItemListComponent.js > ItemListComponent > items.map() callback
const ItemListComponent = ({items,dummy})=>{

    return(
          <div>
           {items.map((e)=>(

              <div  data-testid="foodItems"
              key={e.item.id}  className="p-2 m-2  border-gray-200 border-b-2 text-left flex justify-betv
                  <div  className="w-9/12" >
               <div className="py-2">
                  {/* show item name */}
                     <span >{e.item.name}</span>
                     {/* to show item price, if price is not there show default price */}
                     <span> - ₹ {e.item.price!==0 ? e.item.price : e.item.default_price }</span>

                     </div>
                     {/* to show item description */}
                  <p className="text-xs" >{e.item.desc}</p>
                  </div>
```

2)create one mock json
https://www.zomato.com/webroutes/getPage?page_url=/nagpur/krishnum-restaurant-mominpura/order&location=&isMobile=0



3)
Cart.test.js

```
Cart.test.js U   HeaderComponent.test.js U   ItemListComponent.js M   RestaurantMenuCa
src > components > __tests__ > Cart.test.js > it("Should Load Restaurant menu Component") callback
import {act, fireEvent, render ,screen} from "@testing-library/react";
import RestaurantMenuCard from "../RestaurantMenuCard";
import HeaderComponent from "../HeaderComponent";
import MOCK_DATA_NAME from "../mocks/mockResMenuData.json"
import { Provider } from "react-redux";
import appStore from "../../utils/appStore";
import { BrowserRouter } from "react-router-dom";
import "@testing-library/jest-dom";

global.fetch =  jest.fn(()=>
     Promise.resolve({
          json:  () => Promise.resolve(MOCK_DATA_NAME)
})
)

it("Should Load Restaurant menu Component",async ()=>{

    await act( async ()=> render(
          <BrowserRouter>
          <Provider store={appStore} >
  <HeaderComponent/>
<RestaurantMenuCard  />
          </Provider>
          </BrowserRouter>
 ))

  const accordianHeader=screen.getByText("Dosa (11)");
  fireEvent.click(accordianHeader);

 const foodItemsLength = screen.getAllByTestId("foodItems").length;
 expect(foodItemsLength).toBe(11);

   expect(screen.getByText("Cart(0 items)")).toBeInTheDocument();

    //click on add btn on menu Item
  const addbtn= screen.getAllByRole("button", {name : "Add +"});
  console.log(addbtn.length);
```

```
  const addbtn= screen.getAllByRole("button", {name : "Add +"});
  console.log(addbtn.length);

   fireEvent.click(addbtn[0]); //click on first button.
   //My headerComponent's cart should chnahge to one items.
    //so, we need Header Component to be render with RestaurantMneucard component.'

    //now total cart item will be 1.
  expect(screen.getByText("Cart(1 items)")).toBeInTheDocument();

  //add one more item into cart
  fireEvent.click(addbtn[1]);

    //now total cart item will be 2.
    expect(screen.getByText("Cart(2 items)")).toBeInTheDocument();

});
```

o/p=>

```
PROBLEMS 1  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

 ItemListComponent.js        |     100 |      50 |     100 |     100 | 36-45
 RestaturantCardComponent.js |     100 |     100 |     100 |     100 |
 RestaurantMenuCard.js       |     100 |     100 |     100 |     100 |
 Sum.js                      |     100 |     100 |     100 |     100 |
utils                        |     100 |     100 |     100 |     100 |
 appStore.js                 |     100 |     100 |     100 |     100 |
-----------------------------|---------|---------|---------|---------|-------------------

Test Suites: 6 passed, 6 total
Tests:       14 passed, 14 total
Snapshots:   0 total
Time:        4.833 s
Ran all test suites related to changed files.

Watch Usage: Press w to show more.
```

->Now, also I want to test the 2 item which we added my menuCard, that 2 items must be present in CartComponent as well.
Add cartComponent here,

```
it("Should Load Restaurant menu Component",async ()=>{

    await act( async ()=> render(
        <BrowserRouter>
        <Provider store={appStore} >
  <HeaderComponent/>
<RestaurantMenuCard  />
<CartComponent/>
        </Provider>
        </BrowserRouter>
 ))
```

Cart.test.js U | CartComponent.js | ItemListComponent.js M | RestaurantMenuCard.js | {} mockRest

src > components > __tests__ > Cart.test.js > it("Should Load Restaurant menu Component") callback

```
import {act, fireEvent, render ,screen} from "@testing-library/react";
import RestaurantMenuCard from "../RestaurantMenuCard";
import HeaderComponent from "../HeaderComponent";
import CartComponent from "../CartComponent";
import MOCK_DATA_NAME from "../mocks/mockResMenuData.json"
import { Provider } from "react-redux";
import appStore from "../../utils/appStore";
import { BrowserRouter } from "react-router-dom";
import "@testing-library/jest-dom";

global.fetch =  jest.fn(()=>
    Promise.resolve({
        json:  () => Promise.resolve(MOCK_DATA_NAME)
})
)

it("Should Load Restaurant menu Component",async ()=>{

    await act( async ()=> render(
        <BrowserRouter>
        <Provider store={appStore} >
  <HeaderComponent/>
<RestaurantMenuCard  />
<CartComponent/>
        </Provider>
        </BrowserRouter>
 ))

  const accordianHeader=screen.getByText("Dosa (11)");
  fireEvent.click(accordianHeader);

 const foodItemsLength = screen.getAllByTestId("foodItems").length;
 expect(foodItemsLength).toBe(11);

   expect(screen.getByText("Cart(0 items)")).toBeInTheDocument();

   //click on add btn on menu Item
```

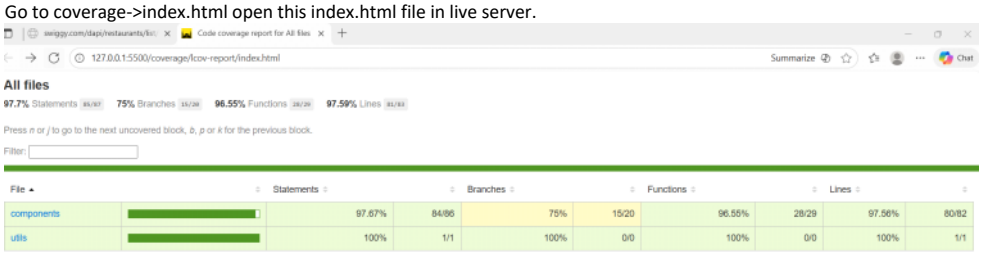
```
36
37      //click on add btn on menu Item
38    const addbtn= screen.getAllByRole("button", {name : "Add +"});
39    console.log(addbtn.length);
40
41     fireEvent.click(addbtn[0]); //click on first button.
42     //My headerComponent's cart should chnahge to one items.
43      //so, we need Header Component to be render with RestaurantMneucard component.'
44
45      //now total cart item will be 1.
46    expect(screen.getByText("Cart(1 items)")).toBeInTheDocument();
47
48    //add one more item into cart
49    fireEvent.click(addbtn[1]);
50
51      //now total cart item will be 2.
52      expect(screen.getByText("Cart(2 items)")).toBeInTheDocument();
53
54       //Checking cart contain 2 items or not(total fooditem will be 11(from menu items)+ 2(from cart))
55      expect(screen.getAllByTestId("foodItems").length).toBe(13);
56
57      //click on clear cart
58      fireEvent.click(screen.getByRole("button",{name:"Clear Cart"}));
59
60      //Now total =>fooditem id will be 11
61      expect(screen.getAllByTestId("foodItems").length).toBe(11);
62
63      //once cart empty we display below message.
64      expect(screen.getByText("Your Cart is Empty add ietms to teh cart")).toBeInTheDocument();
65
66  });
```

Note::-
Our CartComponent internally uses ItemListComponent to show cart Items. Due to this at line no.55 we get answer as 13.

->Our all testcase is passed.
->we did below things in testing episode.

```
# Setting up Testing in our App
-install React testing library
-Install JEST
-Install Babel dependencies bcz our parcel uses babel.
-Configure babel by creating one file at root level.
-Configure  parcel config file to disable default babel transpilation.
-JEST configuration.  npm init jest@latest
-install jsdom library.
-Install @babel/preset-react - to make JSX work in test cases.
 -Include @babel/preset-react inside my babel config.
-Install @testing-library/jest-dom
```

->To Check code coverage in UI,
Go to coverage->index.html open this index.html file in live server.



We can see our code coverage for each file.

==========

++

EPISODE 14: NetFlix GPT - The Beginning

++

In this, episode we build the project using whatever we learn until this project.
Also watch YouTube clone project as well.
In This, we built the NetFlix project.
Features we build.
1)we see how authentication works.
->we will not using nodejs. We use googlefirebase as our backend & then we will connect our react application with google firebase backend.
->we see how we can protect routes from authentication.
Ex:- if user is logged in, then go to partilcuar page, we will build that flow.

2)Form Handling
->How to handle form, how to handle a sign up form,.

NetFlixGPT=> NetFlix + GPT.
We use chatGPT API's to search for movies.
Ex:- we provide input, we type I want to watch some funny movies,
So netflixGPT will recommended us some funny movies.
We can use GPT concept in food ordering app also.

Episode 14 Part 01

==========

->Whenever

//3.32.58

Note::
->Whenever we are testing UI Component inside React, we need to render that component onto the JS DOM first of all.