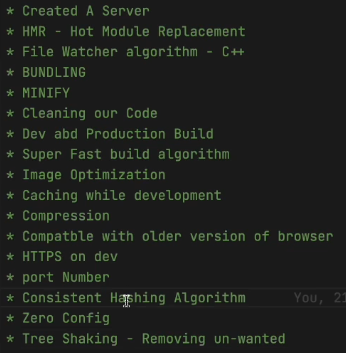
**Akshay Sainis React**

**Day 3:**

Parcel is a bundler

This below all are transitive dependiecies for such below operatrions parcel is depended on some other packages



React cant make performance app alone,it requires diff thngs like parcel,

* Dev dependencies and nrml depeendienecies

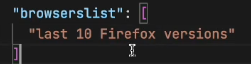
What we have learned:  
-> npm init it gave package.json

* For installing parcel,
* Npm I –D (for dev dependencies) parcel
* Cmd to executr our project: npx parcel index.html
* It created parcel.cache and a server for us, it enables many things like in the above img
* Installing react: npm i react
* Then, import, for that, we have to give type=”module” in script tag in html file,
* Diff b/w packagelock and pck.json
* Should I push parcel cahce on gitignor ? YES
* You should everyting in gitigonre that u can regenerate on server

**HomeWork:**

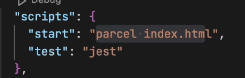
Should have the curiosity

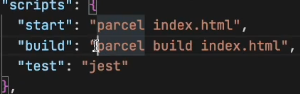
**DAY 4: React-Laying the foundation**

**broweserlist:** it means it definetly work on this, but other browser also it does, but few features may not be supported

**Polyfil:** a code which is replication for a newer version of code

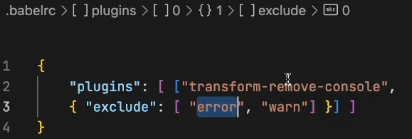
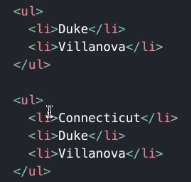
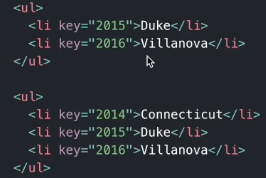
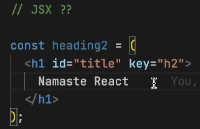
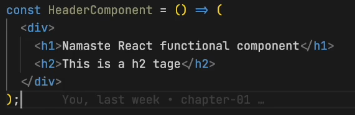
we do not do this, babel does for us(new version to older version).

**For building our app**, we type the cmd **npx parcel index.html** Rather we can add these cmds in script and jst have **npm run start**

Today we are going to talk **about jsx, babel**, **Npm init configuration for managing dependiencies,** Babil is a node pckg, a lib.

Same can be happened with build also, we can write build script, now we can type **npm run build**

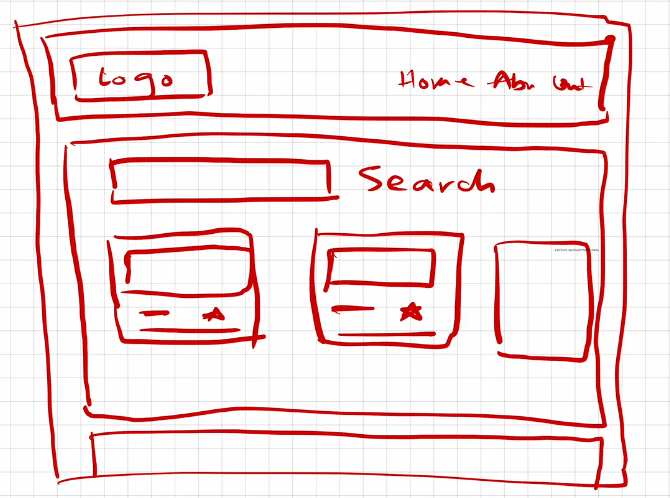
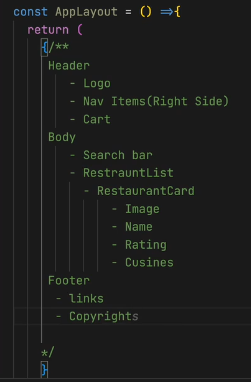
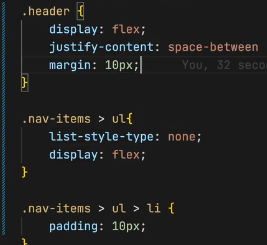
**Npm start and npm run start are same**

* In dist folder, we can see, no **conslole logs been removed, this can be done using pckg called**, **babel plugin transform** remove console, for this we are using a plugin we are configuring it
* **npm i babel-plugin-transform-remove-console --save-dev**
* now after installing it wont work we need to configure it, will be creating .babelrc file, writing this in file
* Then npm run build which will create files in dist fldr, it will not contains console.log but it contains .error
* ** Check react reconsilation key, Read about** Diffing algo, i.e., we uses key, consider the li tag, if we want to add one more li tag on top of prev Ii tags **then it is time consuming, as need to rerender the whole dom**, which **effects changes in dom tree,** **Therefore** it makes easy to **use key in such scenario**
* **How does creatElement woks? .**
* Well it is creating an object, React.CreateElement is creating an object, which then conv into html code, then put upon the DOM.
* **Creating huge html structures with createElement will mess things up, instead of this we use jsx**
* **Is jsx html inside** JavaScript ? T/F -> it false
* It is a html like syntax but not html
* Diff b/w html and jsx
* In jsx we use camelCasing i.e, tabIndex, not tab-index **and className not class**
* How does jsx executes the code, if u type in browser it doesn’t supports it, **babel understands it**,
* Img tag in jsx ?
* Babel comes along with parcel , html in () Is known as jsx expression
* Now let’s learn **React** **Components**
* Everything is a component in react, **we got 2 types of component, functional(NEW) and class(OLD)** in this course will woking with functional components
* ****
* **Functional Component**  is nothing but a function, A function returing a react ele is known as react functional comp
* For any Compoent **the name starts from a Capital letter**, for good practices
* No need of () for single line code, for multiple lines need () can also be written as this, without any return or arrow function thing.
* It is returning jsx
* When we have to render the functional component then will render this way **<function1/>,** and react ele, nrmlly
* We can also use react element inside function using **{heading}**
* Any peace of js code can be written in {}
* Now lets say the api, is returning some malicious code, such attacks are known as cross site scripting attack (xss), by injecting some js code, if it is able to run in our lap then it takes data, **BUT JSX takes care of such attack, i.e, it sanitizes the code and protects from such attack**
* **Component Composition:** such that, we’ve to use a component inside a component

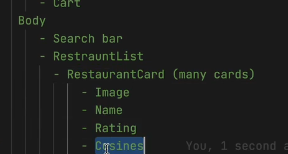
**Revision:**

* Browserlist
* Babel, and its config
* Why do we need keys and diff algo
* What is jsx? It is using React.CreateElement behind the scenes
* Functional component, and class its using, {} </>

**Day 5: Talk is cheap, show me the code**

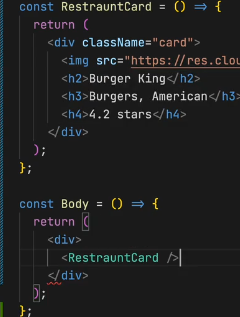
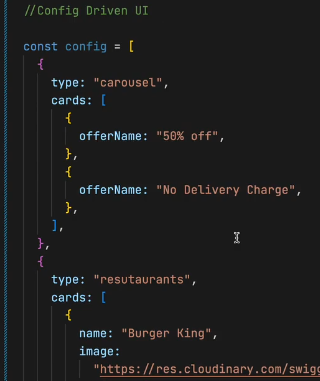
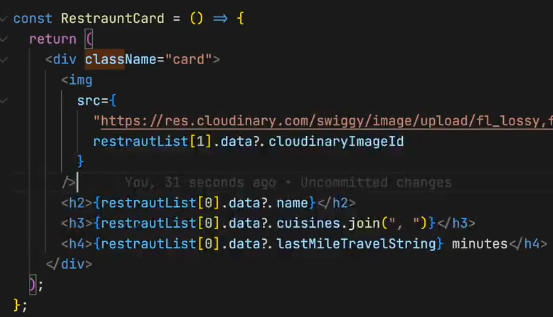
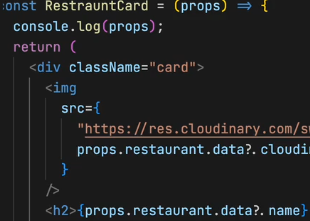
* We can be able to write javascript code inside these {}
* We can call our functional component like {Title()} or <Title/> or <Title> </Title>
* React is a proper library
* Is jsx / es6 is mandatory for react-> NO
* **Now Let build our APP**
* Will be building **Food Villa App**
* Before writing the code, do plan about our App
* ****
* 1st let’s build header
* 1st will be building our layout->structure for that build a component name APPLayout
* 
* Start coding for header component, **add home, about, contact**, cart put **some CSS to align it**
* 
* Add an image/ logo in title component Give a class and resize it also try to add anchor tag
* **NOTE: Any piece of jsx component, there can only be 1 parent**
* i.e., const jsx = <h1>hii</h1> is Correct but
* const jsx = <h1> hii </h1> <h1> hello </h1> is Wrong
* Therefore, we can wrap this thing inside div
* <div></div>
* Therefore, we can use React.Fragment

**Now will be building our resturent card**

****

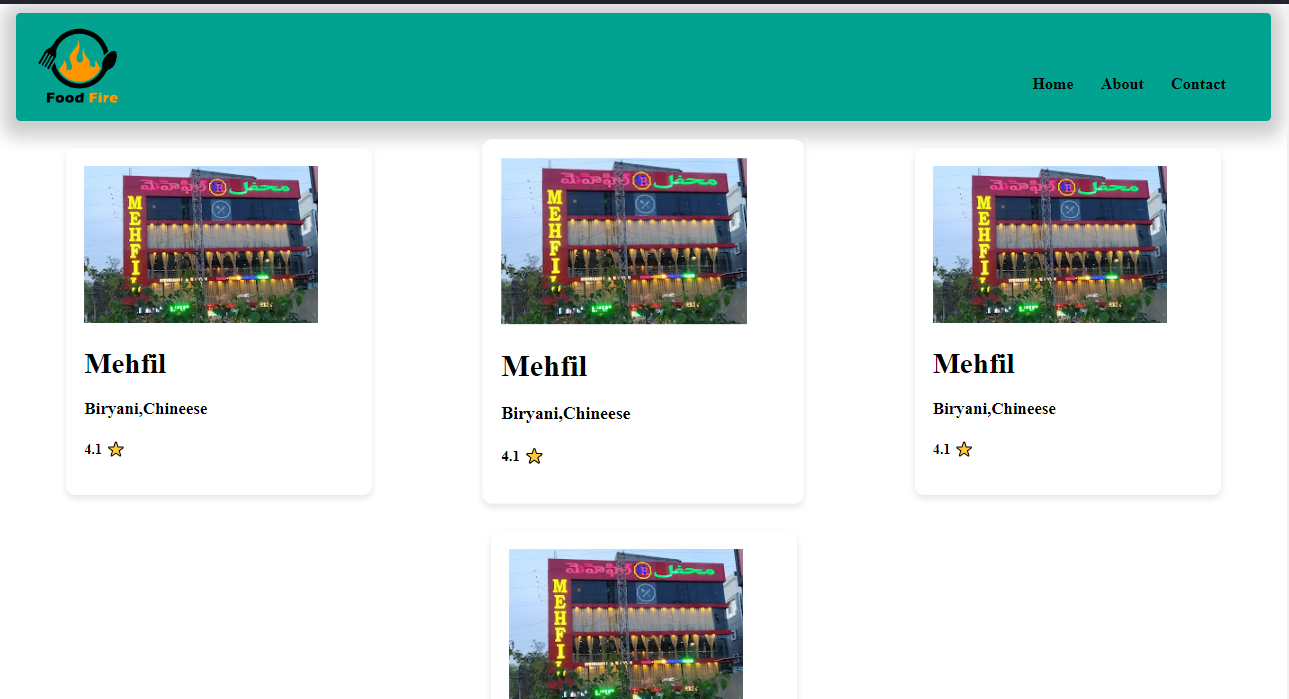
**We’ve** to work on our body part

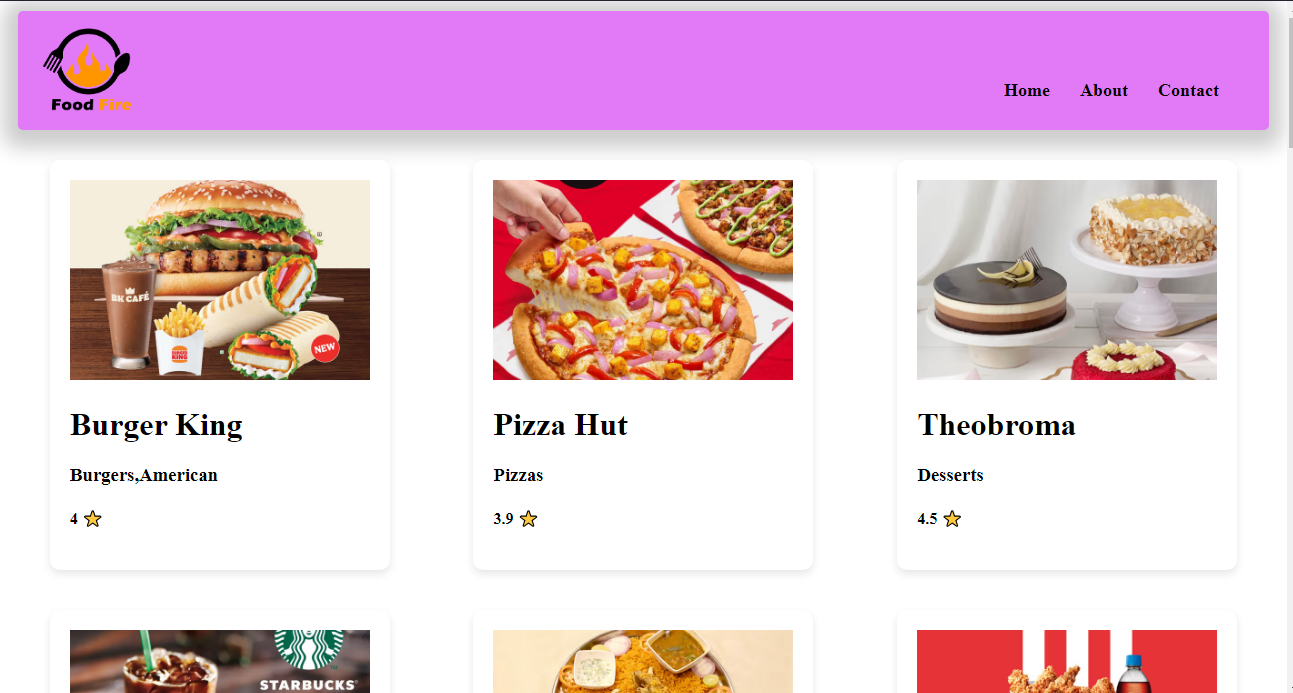
Whenever I’m having this restaurant cards then I need to pull data from some where, So remember this thing, Whenever I’m using UI I’will all way’s be conserned about where will my data come from ?

* For now we will be using hard coded data, then will also learn how to use it through API’s as well
* Will be constructing a functional component with name restaurant card, which will be **returning a jsx element**
* After creating the card, put this in Body, make the returning stmt as div and add **the <Restraurcard/>**
* **Now, give some styling in css to card**
* **It will appear on browser.**
* Now, as we’ve hard coded our restaurant card, but the img, name, lable, rating and all is diff for all, Therefore, we need to make it dynamic
* For that, let’s utilize the javascript, will be creating a js obj, to do dynamic access of data we can do this way.
* To have a (,) in cusines as it a array, we can use join, i.e., **burkerking.cusines.join(“,”)** this how you join items in an array.
* Now, we made our data dynamic, but in the real world there will be so many restaurant, if function call them again and again not good,
* **Display flex, is useful to view the div comp, in horizontal format, also flex-wrap : wrap, can also use grid do ur css**
* **Now lets try to make our all cards dynamic**,
  + In real world the data comes in list formate, such that there will be so many of it.
  + **Config driven UI:** dynamic ui is called as config driven UI, such that, the whole ui is driven by a config which is sent by a backend.
  + **It is Very much useful if you built your Project, using config driven UI**
  + How, will be able to control our UI with backend.
  + As shown in img, if at some place there are no offers and there is no need of any carousel, then backend wont send us such kind of cards, will be directly showing the resutaruants.
  + **In your system desingn round you have to tell this to your, interviewer.**
  + **Now,** Will be building such kind of dynamic UI
  + **Let’s use real data, of swiggy not** the mog data
  + For this, go to inspect, then FetchXR then click on preview it shows the data, or also try reloading the data. Now copy the restaurants data into your code, now how to use that data? Use it normally as shown in img.
  + After getting the data from swiggy, now how to use it, will be using as restuarentList[0].data.name suppose if the name doesnot exist then u’ve to use this **restuarentList[0].data?.name this called optional chaining** (JS thing), also the image can be used as shown
  + Now let’s try **to make this dynamic**, such **that the 1st, 2nd** **and 3rd card should** be showing the info as of list ordering
  + **1 way is to use props**
  + **Props->properties,** i.e, I’m passing my data into my component, functional comp is just a function like in js
  + In functions we’ve concept of arguments and parameter, we pass in arguments we receive parameters
  + In react to use, it as we are passing the props, to use in function we’ve take in as parameter, and can name it anything (props).
  + **Props is like jst a nrml function call as we do in our codes**
  + We can pass any no of args
  + It is received as a para here
  + Now, here’s the thing come up by cool developers, what they do is **destructuring** the object, instead of props use,
  + Now we don’t need props. I can use my restaurant
  + Again you can destructure it like this, with this we can directly use the tags.
  + ****
  + For this will call like: 
  + **If lets say you have destructured the whol at params u got name, cusines, ratings and all so can you pass from u args this way ? well the answer is NO.**
  + **What u do is, u cann pass each with separate tags from the args, or u can use (…) this operator to jst travers all**
  + ****
  + It will work, It is your **spread operator, this is JS es6, in {} can write any** js
  + So**, what if there are 50 cards, then** we cant go on writing this way, therefore, we **uses for loop, but** in the industry we don’t use for loops instead e use **MAPS,** but also we can have for loop,
  + **For maps, as the restaurentList is an array, restaurentList.map(give a callback fun), the fun takes each obj, and for each obj I want my fun to return jsx, which is my <restuarentCard/>**
  + **Everything we’ve builded is like a config driven UI**
  + **Today, we’ve made many things, we made up our header, nav bar, logo, cards (coming from swiggy),**
  + **Now lets dive into virtual DOM**
  + Virtual dom is not jst the concept of react it is a software engineering concept.
  + **We keep a representation of the dom, with us this is known as virtual DOM,**
  + **Why do we need it ?**
  + **A: we need it for reconsilation, reconsilation is an algorithm that react uses, to diff one tree from other and it determines what need to be changed and what not in the ui.**
  + Here the key concept comes into the picture, i.e., lets say we got multiple divs and one dive is been added, as we know the virtual dom only changes the updated div, Therefore to identify the divs we use **keys.**
  + **VirutalDOM is the representation of the dom, and react uses something known as reconsilation, it will find out the different between the tress and only the portion that is required,**
  + Hotmodule reconsilation, this is different, this is on file, which is been down by parcel.
  + ****Also read about **React Fibre**, which came in react16 its new reconsilation engine and it is responsible for div
  + **Now, you should never forget to give key**
  + **Why don’t we use index as the key ? **
  + It will not give error, but u should never have to use index as your key.

**Now lets revise:**

* **1st we**  did planning, then learn about <> </> this
* **Then** we also learn about functional component
* Then we started building our card.
* Then we builded our UI
* Then we had our hard coded data
* Then we made our card dynamic, Then we wanted multiple cards to be dynamic
* Then went to swiggy api, then also learn about carousal of swiggy, i.e., wants to hide and all, such thing are known as **config driven UI, such that u give me a config, my ui will render automatically**
* **Then** we started using props.
* **Such that, pass**ing args
* **Then we** studied about map function and also spread operator
* **Then**  we learned about Virtual dom, reaconsilation algo
* **Then** we learned can I have index as a key

**Practice session:  
**

****

**Day 6: Let’s get hooked! 01-02-2024**

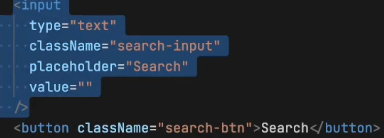
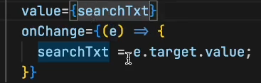
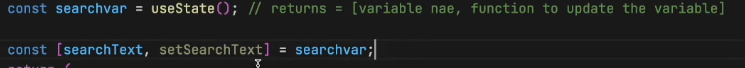
Created a folder named src, do we need to create folder well not, but maintaining proper folder structures this gives us a better modular structure and findable, src is a comman convention been using used in the industry, Now will keep all our files related to our app inside src folder.

* React folder structure, read about it. We can distribute our files depending on the feature of the app .
* Create a src folder then a comp folder, then a file name Title.js add the title code in it, then export the Title component from the file, then import it in app.js and use it
* There are 2 ways of exporting it 1. Export default Title, it will export Title as default i.e., Learn about it, also when u export by using default u use it like
* Now have header and title components in same file, now can I export both of them using default, U can’t I can export only one thing, so if I export this way such that**, exporting by name** then u’ve to use like

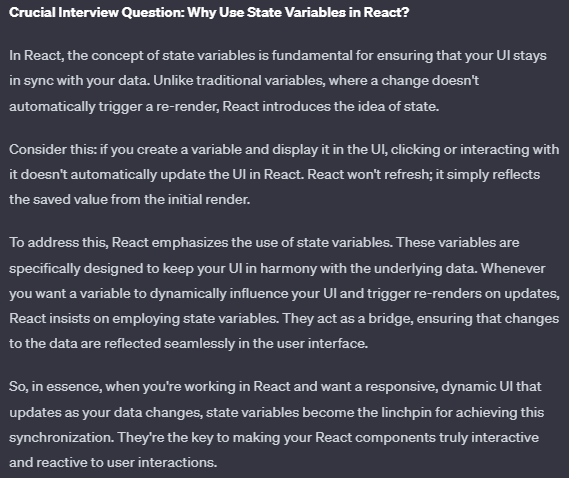
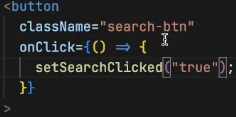
In default import u don’t put {} for named import u have to u {}, when to u use what??

* If you named exported both the comp then you can import this way
* Now if my Header is default export then we can import this way  , does this name have to match the export one, well NO for the default export there is no need, you can change the name, but also try to keep the same name it is good practice.
* And I can also import as ./components/Header.js it will also works fine. And in the react dev community some people are try to name there file as .jsx cause it contains react code in it. And you can also import as .jsx
* Sometimes this can break when you use external library.
* **Now if I’m exporting all components** and named comp, then I can import like this  and I can use like , Now you must be getting why we used  in the code.
* **Now,** What should I be following, Well I try to **export default Header,** Cause I do not export what is not needed, cause title in not needed in App.js right we need it only in our Header component, so I don’t export it, So I jst import things without {}, I only use it when there is real need.
* **HomeWork:** named import, diff ways to export, export default, how do u export a default exp, a named export, what happens when u write \*, also what happens when u name ur files.
* **Now** let’s try to create components for other files as well,
* Let’s create for body comp, create a body.js also do it for footer, so u can destructure you files as much as u can
* **Now Listen very important thing** **we also’ve to create a config** file in our project i.e., **config.js, Well** I put all the **hard codded things into my config file, n**ow I’m using a url for image, suppose I want it in other file as well, so I need to again copy paste it there, SO I just wan to import this from one place kind a global var, **in some company it named as constant.js** files as well.
* Here’s how you can’ve your url, better to export as named also try to have the restrauent list in the constant.js file and export it, and import it in body file. Now also create a new file for the RestaurantCard comp as well and use it in body file and we need that img url in Card comp file.
* Now our code looks cleaner.
* So, you’ve to always keep your code cleaner and maintainable this was 1st part of this session.
* Where to keep our css file, generally will be using tailwind css later onn, for that u don’t need css file, for now keep the css in one file only not in sub folder.

**Now will be building the Search bar functionality**

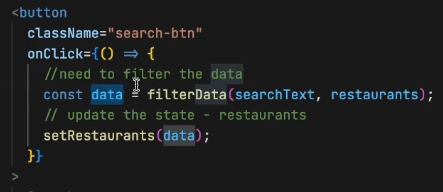
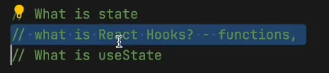
* Now, where do I need search I need it in my body, jst above my res List , give a className “search-container”
* Search container should I’ve input tag in it, also the input should have the value, let it be empty for now, also build a search button. Now in browser when try to write in the input, you wont be able to see **any thing WHY?**
* It will work fine in our html, Cause that input is not same as this, Cause this is controlled by react.
* **How to make this WORK?**
* Even if I hard code the value, and assign it to the value tag in input but, you again you wont be able to edit it.
* **React uses One way DATA BINDING-> i.e.,** I may give the data in the code, but when I try to change from the app, it wont effect the value in the code.
* Now somehow I want to change it thing, How?
* **onChange tag can be used**, will write onChangeInput which takes a function, this function is basically a callback function which e event here  so whenever input is changed this function get called, you can see it , u can see in console the value that is being written.
* But why is it not getting printed over the here, in the box, very Imp question.
* SO, the thing is, whenever you write it the react rerender this component but the value is hardcoded right
* Now, will this thing works ? It wont work like this as well, So how to make this work, So I’m trying to say is Local variable like (searchTxt) is not preferred in react.
* **But if I need to maintain a variable that changes itself, then you need to maintain a variable that is a react kind of a variable.**
* **Now what is a react var? It is a kind of STATE variable, So every component in react maintains a STATE, and you can put your all the variable into the STATE, and everytime you’ve to create a local variable you create a STATE inside react.**
* **Now, Let’s Know what is STATE in react?**
* **Now,** Suppose I’ve to create a local variable like searchTxt, then we will create a with something known as **useState hook, Where** does it comes from ?  , so such varaible are been created using useState hook, Here hook are nothing but kind of like normal funtions, at the end of the day hook is a nrml function. **Where do I get this function from?** Well I get it **from react , who wrote this, FACEBOOK developers. So, useState is use to create state variables.**
* **Now, How do I use my state variable ? SO, ** this function returns a array, and the 1st variable of you array is the variable name  and the searchTExt is a local state variable.
* **These hooks gives** some functionalites, there are different types of hookes that we use across our live course, one of the such important hook is useState hook, this is used to create a local state variable.
* **How** to use this var, we can use it like a normal variable.
* **How** to give a default value for our var, you give it like  this. Also if you want to modify your varaibel then directly you can do that, how to modify it then? Well useState gives us that function to modify the var, good practice to write setVarname , after this u don’t do searchINput = value; u do  or i.e., . Now go and see it will work fine. What ever I write it is getting updated in my local variable
* In angular js we got 2 way data binding
* So, in react it isn’t why it is a good thing, cause your app **becomes unpredictable**. i.e., you may use this var in so many place, so for what we are been changing it its unpredictable. And it is not good for optimization also
* 

It is normal JS destructureing

* **React primarily supports one-way data binding, meaning data flows in one direction from parent components to child components. However, you can achieve a form of two-way binding in React using state and hooks**
* **Very Important Interview Question -> Why do we need state variable?** Generaly if you’ve created a var, and presented in the ui, and with some click you want to update the var, but such thing doesn’t happen in react, react wont rerender it will just print the code saved value, To do that, react says every time **you want your variable in sync with the UI, you need to use your state variable.**
* ****
* **Try** creating a variable, i.e., when clicked it updated to true value,  

With this the UI gets changed, and get updated. Now let’s understand how it is being done. Now when u create the variable state, the react is keeping track of it, now when ever my var gets updated my whole component get rerendered, react is jst destroying the body component and creating the body once again, and it is happening once again.

**Reconsilation is happening here. React is very smart it will just rerender the h1 tag,** React just rerender that portion of your dom that is why react is fast, it using diff alaogrithm.

* **Let the search work,** when type it, let the rest filter out, the thing is when **clicked on the search buttion need to filter the data, the restaurant list**
* Let’s create a function and filter data, after filtering we’ve to update the list for that, we require STATE’s. create a state variable by default the data should be assigned to dumy data right, yes  , now use your restaurants it will work, with map as well , it worked. Now if I pass with filtered data, will the list changes? YES So what I will do is modify the local state variable.
* I’ve to filter the restaurant list using the input I get, and send it to the function then I should be implementing the function which does such work, which give me filtered data.
* 
* Now try to run it, it will work.
* **HomeWork:** Do, the toggle thing that true/false.and answer this  then also restructure your folder then play with export and import, also clear your understanding in onChange(e) on ‘e’ . Also find out empty search why it did not work make it work.

**HomeWOrk ReadOUTS:**

**What is the Virtual DOM?**

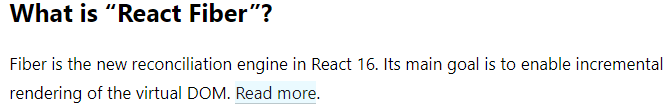
The virtual DOM (VDOM) is a programming concept where an ideal, or “virtual”, representation of a UI is kept in memory and synced with the “real” DOM by a library such as ReactDOM. This **process is called**[**reconciliation**](https://legacy.reactjs.org/docs/reconciliation.html)**.**

This approach enables **the declarative API of React**: You tell React what state you want the UI to be in, and it makes sure the DOM matches that state. This abstracts out the attribute manipulation, event handling, and manual DOM updating that you would otherwise have to use to build your app.

React, however, also uses internal **objects called “fibers”** to hold additional information about the component tree. They may also be considered a part of “virtual DOM” implementation in React.

**Is the Shadow DOM the same as the Virtual DOM?**

No, they are different. The Shadow DOM is a browser technology designed primarily for scoping variables and CSS in web components. The virtual DOM is a concept implemented by libraries in JavaScript on top of browser APIs



React Fiber is a fundamental rewrite of the core algorithm used by React to update the user interface (UI). It was introduced in React version 16.0 as a new reconciliation engine. The term "Fiber" refers to the internal data structure used by React to represent the components in the virtual DOM.

The main goal of React Fiber is to enable incremental rendering of the virtual DOM. In the context of React, reconciliation is the process of determining what changes need to be made to the DOM to reflect the updated state or props of a component. Incremental rendering means breaking down the rendering work into smaller chunks and spreading it over multiple frames. This allows React to better prioritize and manage the rendering process, making UI updates more efficient and responsive.

Key points about React Fiber:

1. **Incremental Rendering:** Fiber allows React to work on rendering and updating the UI in smaller, prioritized units. This means that the rendering work can be interrupted and resumed, allowing for better responsiveness and perceived performance.
2. **Prioritization:** Fiber introduces a priority-based scheduling system that enables React to prioritize different types of updates. This is crucial for ensuring that high-priority updates, such as user interactions, are processed quickly while less critical updates may be deferred or canceled if necessary.
3. **Concurrency:** React Fiber introduces the concept of concurrent rendering, which means that React can work on multiple tasks concurrently without blocking the main thread. This is particularly beneficial for applications with complex UIs and interactions.
4. **Better User Experience:** The improvements brought by React Fiber lead to a more responsive user interface, especially in applications with dynamic and frequently changing content.

In summary, React Fiber is the internal engine of React that powers the reconciliation process. Its incremental rendering approach, along with prioritization and concurrency, enhances the efficiency and performance of React applications, providing a smoother user experience.

**Guide on Reconsilialtion** <https://legacy.reactjs.org/docs/reconciliation.html>

process of reconciliation in React, which is the mechanism by which React updates the user interface efficiently in response to changes in state or props. The generic solutions for transforming one tree into another have a complexity of O(n^3), making them too expensive for practical use.

React relies on a heuristic O(n) algorithm based on two key assumptions:**Elements of different types will produce different trees. Developers can use the key prop to hint at stable child elements across renders.**

**Diffing Algorithm:**

* When diffing two trees, React compares the root elements.
* Elements of different types lead to a full rebuild of the tree.
* DOM elements of the same type are updated by modifying only the changed attributes.
* Component elements of the same type update the props of the underlying component instance

**Stability, predictability, and uniqueness are crucial for keys. Unstable keys, like those from Math.random(), can lead to unnecessary recreations and performance issues.**